Using Evowave for Logical Coupling Analysis of a Long-lived Software System

Rodrigo Magnavita[1], **Renato Novais**[1,2], **Bruno Silva**[3], **Manoel Mendoncça**[1]

[1]Fraunhofer Project Center at UFBA
Salvador, BA.

[2]Instituto Federal da Bahia
Salvador, BA.

[3]Universidade Salvador - UNIFACS
Salvador, BA.

{rodrigo.magnavita, manoel.mendonca}@ufba.br,

renato@ifba.edu.br, bruno.carreiro@pro.unifacs.br

Abstract. *Logical coupling reveals implicit dependencies between program entities, by measuring how often they changed together during development. The comprehension of how the logical coupling property manifests itself in the project is a key activity. Software Evolution Visualization (SEV) has been a promising approach to this end. However, this is not trivial, since SEV has to handle different software entities and attributes, and still deals with the temporal dimension of evolution. In this sense, we have been working on the specification, development and evaluation of a novel SEV tool, called EVOWAVE. This novel visualization metaphor is able to visualize different types of data generated in software evolution using both overview-based and detail-based approaches. It can be applied to different software engineering tasks and domains. In this paper, we present a logical coupling study we conducted in order to evaluate this tool in this important domain.*

1. Introduction

As a software system is developed and maintained across time there is an increasing amount of effort to understand program entities and carry out changes on them. This phenomenon happens, besides other reasons, due to complex dependencies grown among program entities throughout the software evolution. Moreover, such dependencies may not be explicitly found by analyzing a single version of the source code. In that context, authors have recently investigated a property called Logical coupling, which reveals implicit dependencies between program entities, by measuring how often they changed together during development [Robbes et al. 2008]. Coupling has been proved useful in a variety of software daily activities, such as maintenance effort estimation [Gupta and Rohil 2012], program comprehension [Briand et al. 1999], design flaws detection [Marinescu 2004]. Due to its importance, logical coupling has been studied with different purposes, such as: detection of architectural weaknesses, poorly designed inheritance hierarchies, blurred interfaces of modules [Gall et al. 2003], and change impact analysis [Rolfsnes et al. 2016]. In all of those cases, the comprehension of how the logical coupling property manifests itself in the project is a key activity.

Software visualization (SoftVis) has been effectively used as a way to tackle with software comprehension activities using high amount of data. SoftVis supports engineers in understanding software and building knowledge through visual elements, with less complexity over a large amount of data which is often generated during software evolution [Diehl 2007]. The comprehension of logical coupling is particularly addressed in the field of Software Evolution Visualization (SEV), as the property of logical coupling is measured by analyzing how often software modules change together in their evolution.

In the last two years, we have been working on the specification, development and evaluation of a novel SEV tool, called EVOWAVE [Magnavita 2016]. EVOWAVE realizes a novel visualization metaphor [Magnavita et al. 2015]. It is able to visualize different types of data generated in software evolution using both overview-based and detail-based approaches. EVOWAVE is able to represent a huge number of historical events at a glance. Several mechanisms of interactions allow the user to explore the visualization in detail. EVOWAVE has one perspective, implemented in a circular layout. This open source tool can be applied to different software engineering tasks and domains. To validate its benefits, we conducted three exploratory studies using EVOWAVE in three different software engineering domains: software collaboration [Magnavita et al. 2015], library dependency, and logical coupling.

In this paper, we present the logical coupling study. As in the other two studies, we selected a close related work [D'Ambros et al. 2009], and replicated the study they conducted. Our goal was to show the benefits of EVOWAVE in a new domain (logical coupling) considering an already published work, and also to comparatively highlight the benefits of our tool.

Other works provide a similar visualization layout for software evolution analysis [Kula et al. 2014][D'Ambros et al. 2009][Burch and Diehl 2008]. In [Kula et al. 2014], the authors proposed an evolution-based visualization of the coupling relationship between system modules and their required libraries. Although providing a similar visual representation, they do not focus on logical coupling relations. D'Ambros et al. present a visualization-based approach, called Evolution Radar [D'Ambros et al. 2009], that integrates logical coupling information at different levels of abstraction (package level and file level). In addition, they illustrated a retrospective analysis driven by logical coupling in the context of the ArgoUML project involving its maintenance activities such as restructuring and re-documentation. Despite the fact EVOWAVE has been used in the same domain, we believe our tool has advantages against Evolution Radar since it is a fresh web-based open source tool, highly configurable, and supports visualization of events of any kind across time (besides software evolution properties).

In our context, we use the EVOWAVE, a multiple domain software evolution visualization metaphor, to perform the same exploratory study performed in [D'Ambros et al. 2009] on the analysis of ArgoUML's logical coupling information. Therefore, this paper contributes by showing the study on the ArgoUML's logical coupling history using a single perspective software evolution visualization.

This paper is organized as follow. Section 2 presents a brief description of EVOWAVE. Section 3 presents the study we conducted to evaluate EVOWAVE in the logical coupling domain. Finally, Section 4 concludes this work.

2. EVOWAVE in a Nutshell

EVOWAVE is a new visualization tool that enriches the analysis capabilities of software evolution. It is inspired on concentric waves with the same origin point in a container seen from the top. This section briefly presents the concepts related to the EVOWAVE. Figure 1 shows one example of EVOWAVE. It portrays a logical coupling data history of ArgoUML project. For the sake of understanding, this example is decorated (mockup) with labels pointing to the main concepts of the metaphor.

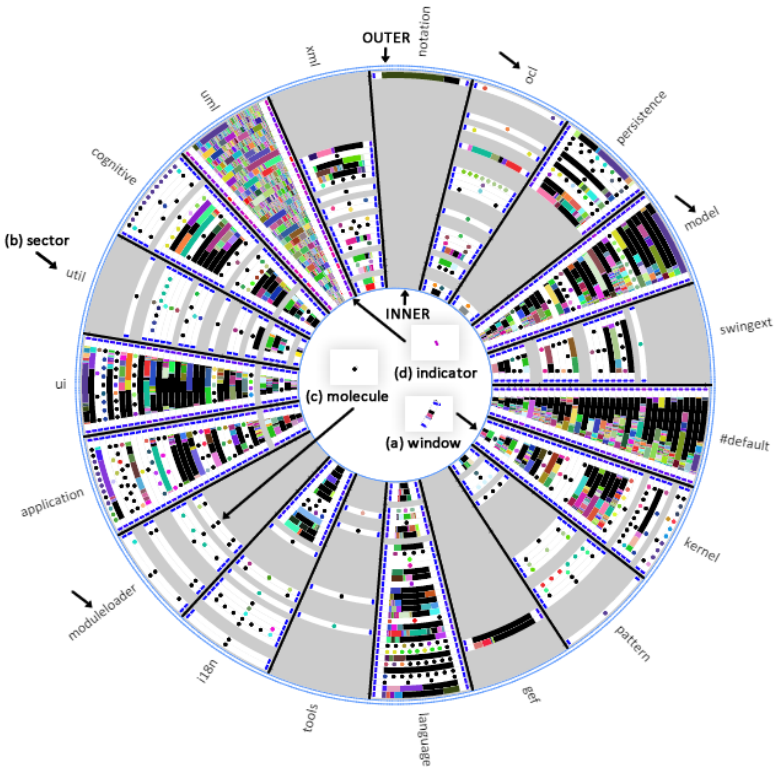

Figure 1. EVOWAVE visualization with all changes separated by modules from 2003 to 2005 with the "uml" package in focus.

EVOWAVE has a circular layout with two circular guidelines (***inner*** and ***outer***). The path between them represents a software life cycle period between two selected dates. This period, named ***timeline***, gives an overview of the software history. It is comprised by a series of short periods with the same periodicity (e.g., ten days, two hours, one month).

A ***window*** is a group of consecutive short periods (Figure 1-a). It is circular in shape and its length depends on the number of grouped periods. It can be used to compare a subset of short periods, making it possible to carry out a detailed analysis regarding the overall context. A ***sector*** is a visual element drawn between the two circular guidelines according to its angle (Figure 1-b). It may have different angles. This concept is used to group events that share some characteristic (e.g., classes of the same packages).

Molecules are depicted as circular elements inside sectors and windows (Figure 1-c). Each molecule has an event associated to it. When we have more molecules than the display size limits, we gathered and drawn them as a quadrilateral polygon that fills the

region where the molecules are. Molecules can represent any change on software history, such as file changes, team changes or bug reports. The molecules' color can be used to map software properties as an event categorization or a numerical property range. The number of molecules ***indicator*** is drawn as a rectangle located in the frontier of the sectors for each window (Figure 1-d). Its color varies from red to blue, where the reddest indicator has the largest number of molecules and the bluest has lowest number of molecules. The indicator can refer to the local sector or to all sectors. If set to local, its color will take in consideration the other windows inside the sector. Otherwise, if set to all sectors (global), its color will take in consideration all windows present in the visualization.

EVOWAVE provides a set of ***mechanisms of interaction***. They allow users, for example, to navigate (zoom in) into sectors of interest or to highlight a window across all sectors. In this last case, it mitigates possible problems that occurs when we have to much data and many used colors.

3. An Exploratory Study on Logical Coupling Domain

This section presents the study we conducted to validate EVOWAVE on the analysis of logical coupling between software modules during its history. Using the GQM (Goal-Question-Metric) paradigm [Basili and Rombach 1988], the goal of this study was: **To analyze** the EVOWAVE metaphor; **With the purpose of** characterize; **Regarding** logical coupling evolution between software modules; **From the point of view** of EVOWAVE researchers; **In the context of** a retrospective analysis made by D'Ambros [D'Ambros et al. 2009] in a real world large open source system called ArgoUML.

3.1. Study Settings

ArgoUML is the most well-known open source UML modeling tool written in Java with more than 17 years of development and at least 200,000 lines of code distributed in 4,222 classes.

Setup. We developed a parser to read Subversion logs and identify the files that were modified together within the "org.argouml.uml" package (which is the root package of the software). For each commit, we extracted the following data: the changed files and their package; when the change occurred; and the commit id. With this data we setup the EVOWAVE as follows: **The period of analysis** involved 173 months, from September 4th, 2000, to January 11th, 2015; **Sectors** are source code packages; **Windows** represent six months; **Molecules** represent changes in a Java file; and **Molecule Colors** represent the commit of the change. When a package is selected the commits in black represent revisions that did not make changes in the source code.

Tasks. We followed the same approach of [D'Ambros et al. 2009], which uses the logical coupling between modules to understand implicit dependencies in the system. The main goal is to make a retrospective analysis of the logical coupling evolution in the ArgoUML project.

3.2. Study Execution

First, we made an overview analysis in order to understand how the changes were spread among the system modules (i.e. Java packages). Figure 2-A helps us to identify packages with most changes. The root package "org.argouml.uml" underwent most of the changes

during fourteen years of development, followed by “#default” and “kernel” respectively. In addition, only one window in “org.argouml.uml” is empty (gray color arc in the package sector), while “#default” has two and ”kernel” has three. This means that, within the period of 173 months analyzed, there was only one period of 6 months without changes in the classes of the root package. Therefore, we decided to analyze the logical coupling of such package (“org.argouml.uml”) due to: 1) the amount of changes in it; 2) it was intensively changed from the beginning to the present date (almost fifteen years); and 3) it is an important package as it contains core classes for the UML tool.

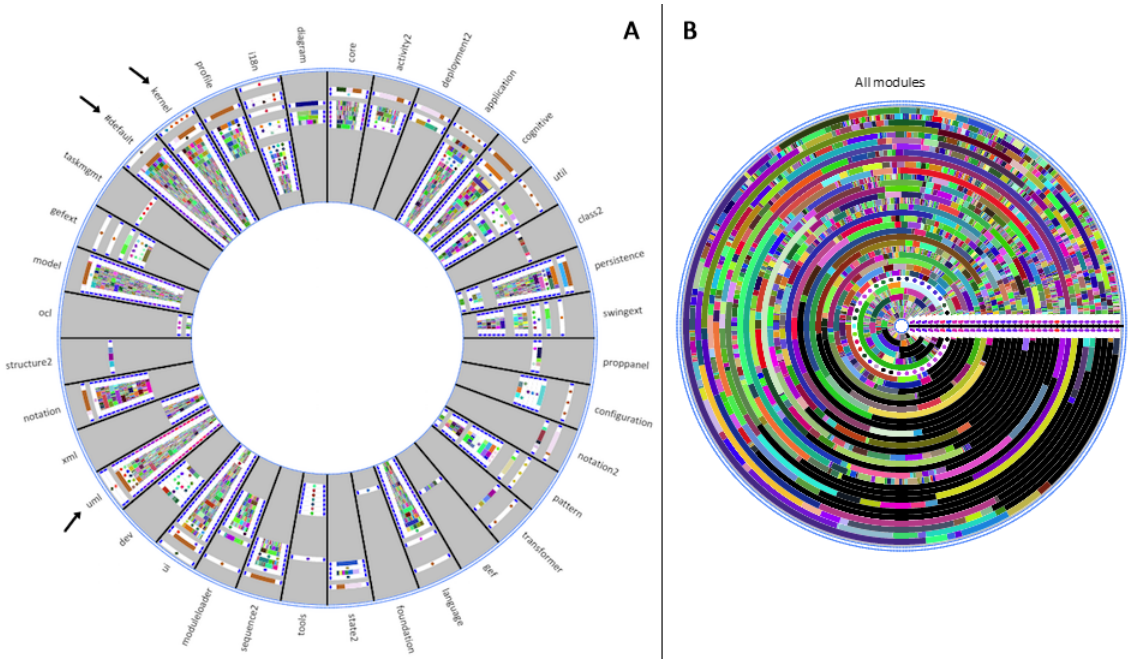

Figure 2. A) EVOWAVE visualization with all module changes during fifteen years of development; B) EVOWAVE visualization with all module changes from 2003 to 2005, focused on the “uml” package.

We filtered the period limiting it from January 2nd, 2003, to December 30th, 2005, in order to analyze the logical coupling of the package “uml” during the period with most changes. Additionally, for more detailed information, we changed the period of the window from six to one month. Figure 2-B illustrates the visualization with this filter and aggregates all modules into one sector to get a global picture of the logical coupling of the “uml” package. It is possible to notice that most commits caused changes in the “uml” package, as the black color has low presence in the visualization. Another worth observing information from this global picture is the number of commits with several files undergoing changes together. This can be noticed by looking at the size of the arcs with the same color. This may be interpreted as a problem to be further investigated in a project because the higher the number of co-changed files, the higher is the logical coupling. Therefore, one possible consequence of such high degree of interdependency among source files is an increasing occurrence of side effects during maintenance activities.

To investigate in more details, we analyzed the logic coupling of the “uml” package with other system modules, supported by the visualization illustrated in Figure 1.

Due to space restriction, we selected four packages (*moduleloader*, *ocl*, *model*, and *uml*), to explain how someone could recover more information using EVOWAVE and thus improving comprehension and discovering knowledge about the project.

The package "moduleloader" has a low logical coupling with the "uml" package. Nevertheless, in the fifth early month in the "moduleloader" package there were three changes in the same commit that changed almost all modules. Looking deeper into the comment of this commit and into the changes, we identified they were related to the copyright style and impacted 1,065 files. In terms of source code those changes had no impact, but let us know that probably every single file must change when there is a new copyright information.

The package "ocl" is highly logical coupled with the "uml" package. This is observed because there is almost no black color in the "ocl" sector. Among the changes, there are two commit highlights in the forth newer window: the green and red colors. The commit related to the green color is the copyright change that we demonstrate while analyzing the "moduleloader" package. The commit represented by the red color seems to impact many packages. While analyzing the comment and the changes from this commit, we identified a major change in the system. The class responsible to be the facade for all communications with the "model" package changed its signature from "ModelFacade" to "Model.getFacade()". This is a big change in the system because it breaks the signature used in many packages (e.g., ocl, persistence, uml). The logical coupling between the "ocl" package and the "uml" package for this commit is a consequence of their coupling with the model package.

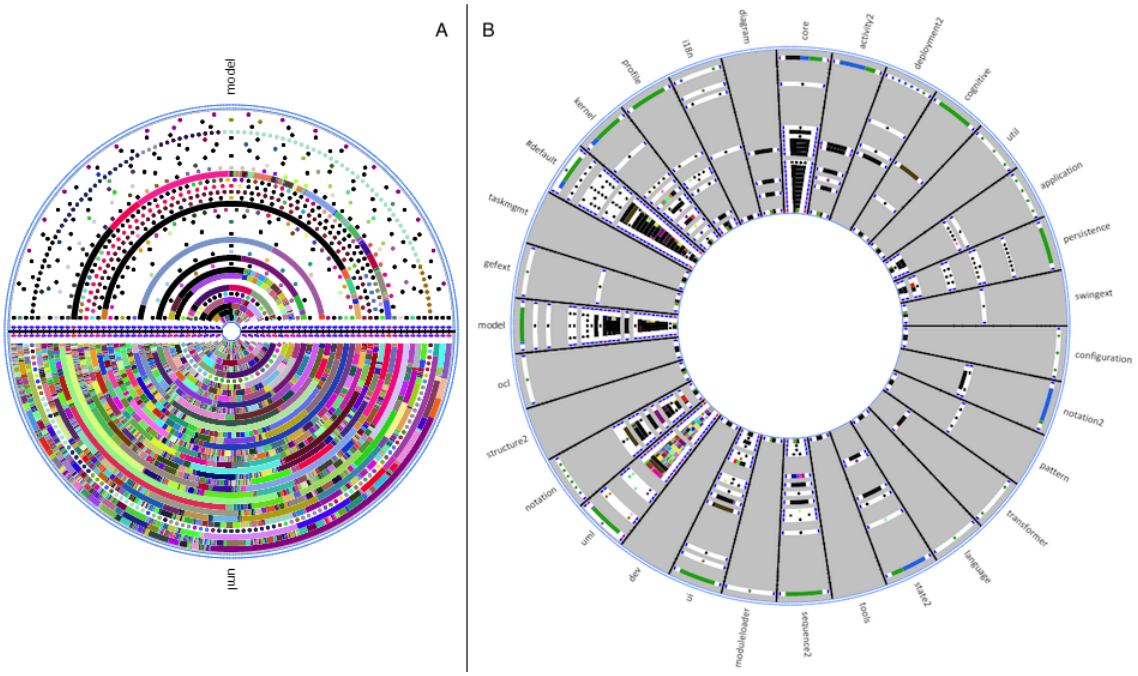

Figure 3. A) EVOWAVE visualization to analyze the logical coupling between the "uml" and "model" packages. B) EVOWAVE visualization from January 2, 2010 to February 8, 2013

The "model" package holds a lot of data because it changed many times during

this period. In this case, we filtered the visualization to show only the "uml" package and the "model" package in order to better understand their logical coupling. Figure 3-A represents the visualization with this filter. The first point to notice is the amount of changes in those packages. Clearly the "uml" package had more changes than the "model" package during this period. Another important point is the frequency of black and other colors. Black molecules are present in practically all months. Nevertheless, there are many colors (i.e. many different commits) present in windows. This leads to the conclusion that many commits were submitted in that month, and they revealed logical coupling with the "uml" package. The eleventh newest window (January, 2005) has the highest number of colors, and was one of the months with most changes. Looking deeper into the commits submitted in that month, we found that those packages are coupled by an architecture decision to have a Facade design pattern to access the "model" package. The Facade pattern can be defined as an entry point to features joined in a module to reduce the need for one module to known another module completely. Having this design pattern as the reason for the coupling of those two packages is a good sign of good architectural decisions throughout the software development.

Additionally, we filtered the period limiting it from January 2nd, 2010 to February 8th, 2013 in order to analyze the logical coupling of the package "uml" in more recent data. Figure 3-B illustrates the visualization for this period. The first thing we noticed was the reduction of the logical coupling among the "uml" package and the rest of the system. The reason for this might have been refactoring activities performed between 2005 and 2010 to reduce the coupling between the "uml" package and the rest of the system. Nevertheless, the green commit at the second newest window stands out in this visualization. This commit impacts practically all active modules in the system. Looking more deeply into the commit and its changes, we identified that the reason was a change in the logging library from log4j to the native Java logging API.

4. Final Remarks

Logical coupling reveals implicit dependencies between program entities. It is useful in a variety of software daily activities. A challenge is to comprehend how the logical coupling property manifests itself in the project. To this end, Software Evolution Visualization is a promising approach.

In this work, we used the EVOWAVE, a multiple domain software evolution visualization metaphor, to perform a exploratory study on the analysis of ArgoUML's logical coupling information. This study showed we were able to evaluate the EVOWAVE in another domain (i.e. logical coupling). We used the same study procedures as in [D'Ambros et al. 2009]. In addition to the fact it is one more domain, we can highlight other outcomes of our tool as it is an fresh open source tool, web-based, and highly configurable.

The exploratory study conducted helped us to see the outcomes and shortcomings of EVOWAVE. As outcomes, it indicated the EVOWAVE usefulness on the logical coupling analysis of a large software system. As shortcomings, it indicated the need of tool improvement. More mechanisms of interaction are necessary to speedup the analysis, specially by end users. The exploratory study itself has limitations, since it is conducted by the authors' perspective. As future work, we plan to conduct other experimental stud-

ies in order to evaluate EVOWAVE in the context of logical coupling domain, however considering end users.

References

[Basili and Rombach 1988] Basili, V. R. and Rombach, H. D. (1988). The TAME project: Towards improvement-oriented software environments. *IEEE Trans. Softw. Eng.*, 14(6):758–773.

[Briand et al. 1999] Briand, L. C., Daly, J. W., and Wüst, J. K. (1999). A unified framework for coupling measurement in object-oriented systems. *IEEE Trans. Softw. Eng.*, 25(1):91–121.

[Burch and Diehl 2008] Burch, M. and Diehl, S. (2008). Timeradartrees: Visualizing dynamic compound digraphs. *Computer Graphics Forum*, 27(3):823–830.

[D'Ambros et al. 2009] D'Ambros, M., Lanza, M., and Lungu, M. (2009). Visualizing co-change information with the Evolution Radar. *IEEE Trans. Softw. Eng.*, 35(5):720–735.

[Diehl 2007] Diehl, S. (2007). *Software Visualization: Visualizing the Structure, Behaviour, and Evolution of Software*. Springer-Verlag New York, Inc., Secaucus, NJ, USA.

[Gall et al. 2003] Gall, H., Jazayeri, M., and Krajewski, J. (2003). CVS release history data for detecting logical couplings. In *Proceedings of the 6th International Workshop on Principles of Software Evolution*, IWPSE '03, pages 13–, Washington, DC, USA. IEEE Computer Society.

[Gupta and Rohil 2012] Gupta, N. K. and Rohil, M. K. (2012). *Object Oriented Software Maintenance in Presence of Indirect Coupling*, pages 442–451. Springer Berlin Heidelberg, Berlin, Heidelberg.

[Kula et al. 2014] Kula, R., De Roover, C., German, D., Ishio, T., and Inoue, K. (2014). Visualizing the evolution of systems and their library dependencies. In *Software Visualization (VISSOFT), 2014 Second IEEE Working Conference on*, pages 127–136.

[Magnavita 2016] Magnavita, R. (2016). *EVOWAVE: A Multiple Domain Metaphor for Software Evolution Visualization*. Dissertation, Universidade Federal da Bahia.

[Magnavita et al. 2015] Magnavita, R., Novais, R., and Mendonça, M. (2015). Using evowave to analyze software evolution. In *Proceedings of the 17th International Conference on Enterprise Information Systems*, pages 126–136.

[Marinescu 2004] Marinescu, R. (2004). Detection strategies: metrics-based rules for detecting design flaws. In *Software Maintenance, 2004. Proceedings. 20th IEEE International Conference on*, pages 350–359.

[Robbes et al. 2008] Robbes, R., Pollet, D., and Lanza, M. (2008). Logical coupling based on fine-grained change information. In *2008 15th Working Conference on Reverse Engineering*, pages 42–46.

[Rolfsnes et al. 2016] Rolfsnes, T., Alesio, S. D., Behjati, R., Moonen, L., and Binkley, D. W. (2016). Generalizing the analysis of evolutionary coupling for software change impact analysis. In *2016 IEEE 23rd International Conference on Software Analysis, Evolution, and Reengineering (SANER)*, volume 1, pages 201–212.