The Small Project Observatory - A Tool for Reverse Engineering Software Ecosystems

Mircea Lungu, Michele Lanza
REVEAL @ Faculty of Informatics - University of Lugano, Switzerland

ABSTRACT

Software evolution researchers have focused mostly on analyzing single software systems. However, often projects are developed and co-exist within *software ecosystems*, i.e., the larger contexts of companies, research groups or open-source communities. We present The Small Project Observatory, a web-based analysis platform for ecosystem reverse engineering through interactive visualization and exploration.

Categories and Subject Descriptors

D.2.8 [**Software Engineering**]: Software Ecosystems

Keywords

Software Evolution, Software Visualization, Software Ecosystems, Mining Software Repositories

1. INTRODUCTION

Software engineering research has produced many successful tools that support the analysis of the static structure of software systems and their evolution, recorded in the associated software repositories. These tools usually focus on individual software systems, however, software systems do not exist by themselves, isolated from other systems, but rather they exist in larger contexts that we call *software ecosystems*: groups of projects that are developed and co-evolve in the same environment. The environment can be a company, a research group, or an open source community.

Ecosystem analysis has the following goals:

- *Understand the past:* make sense of the software legacy stored in the versioning systems of an ecosystem.
- *Control the present:* monitor the activity of the projects, observe the evolution of quality metrics, discover code duplication, etc.
- *Shape the future:* identify reusable components, extract baselines for product families, etc.

ICSE '10 Capetown, South Africa

We are mostly concerned with understanding the past, and in particular we are interested in reverse engineering software ecosystems, i.e., to recover high-level information by analyzing the low-level facts that exist in the versioning systems of the component projects [2]. To support our research, we have developed The Small Project Observatory (hereafter referred to as SPO).

SPO is a web-based analysis framework dedicated to supporting ecosystem reverse engineering [3]. SPO is free and open source software, written in Smalltalk, hence the name. Figure 1 presents a screenshot of SPO running inside the Opera browser. SPO provides *multiple viewpoints* on software ecosystems supported by *exploratory interaction and navigation* facilities.

Although there are other researchers that study collections of projects, to our knowledge we are the first to propose a tool for analyzing and understanding software ecosystems.

2. ECOSYSTEM VIEWPOINTS

SPO offers a large number of *ecosystem viewpoints*, visual perspectives that capture different aspects of the ecosystem structure and evolution [2]. The viewpoints are generated based on the information extracted from the software repositories of the projects that make up the ecosystem. There are two main aspects of an ecosystem that can be extracted by analyzing the low-level facts in the software repositories:

1. *Project-Centric* aspects regard the projects that make up the ecosystem, their properties, their relationships, and their evolution.
2. *Developer-Centric* aspects regard the activity, collaboration, and interdependencies of the developers that contribute to the ecosystem.

For each of these aspects the ecosystem can play one of two roles in the analysis:

1. *The Focus.* The goal is to understand the ecosystem from a holistic point of view. This type of analysis starts from the premise that the problems that are associated with an ecosystem are different from the ones associated with individual systems.
2. *The Context.* The goal is to understand the individual elements, i.e., projects, of the ecosystem. This type of analysis starts from the premise that one can better understand an element if he studies it in its context.

We illustrate four types of viewpoints with examples taken from three case studies (presented in more detail in [2]).

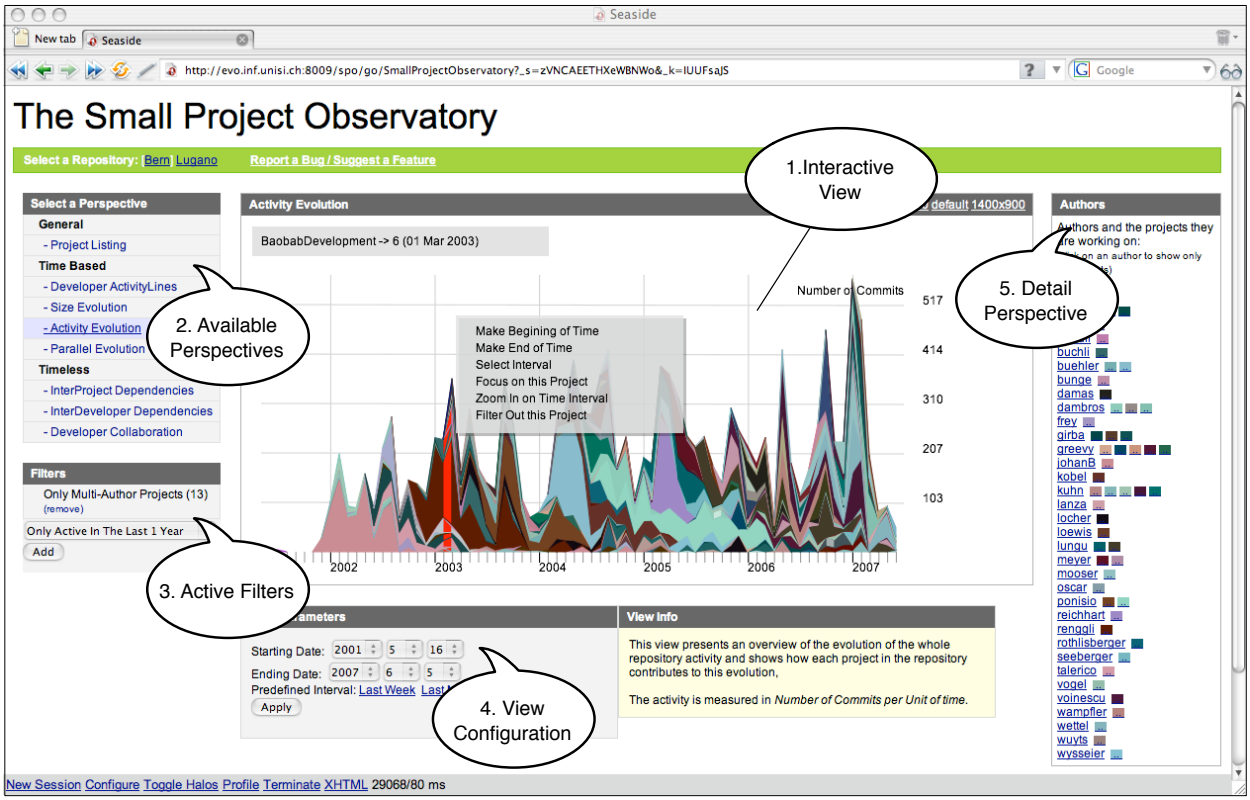

Figure 1: Screenshot of The Small Project Observatory running in the Chromium browser

2.1 Project-Centric - Ecosystem as Focus

Supporting a holistic understanding of the ecosystem in terms of its projects.

There are multiple questions that regard the ecosystem as a whole: How did the activity on these projects progress over time? How did the size of the code in the projects evolve over time? How do projects reuse code in the ecosystem? Are there projects that depend on others? SPO provides various analyses and visualizations that answer these questions. In this section we focus on the *Project Dependency Map*, a viewpoint which presents the static dependency relationships between the projects in an ecosystem.

Visualizing the inter-project relationships highlights the critical projects in an organization because it pinpoints the projects that all the others depend on. As a result, it is mainly a tool for the project manager. However, visualizing the dependencies between projects is also useful for a developer who is trying to understand the way his code fits into the big picture or a developer who is new to the ecosystem.

Figure 2 presents the Project Dependency Map of a subset of the projects in the SCG ecosystem. The nodes represent projects, and their size is proportional to the project size as measured in number of classes. The intensity of a node is proportional to the amount of commits performed to that project. The dependencies between projects imply a static dependency which is either subclassing or method invocation. The figure shows the largest, most active, and one of the most used projects in the ecosystem at the center of the graph. The name of the project is Moose and it represents a framework for reverse engineering [7], on top of which many other tools have been built.

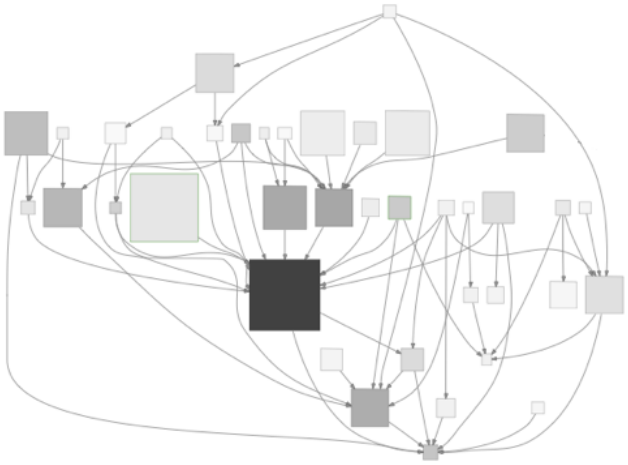

Figure 2: The dependencies between a subset of the projects in the SCG Bern Ecosystem

SPO allows for zooming into the details of the individual projects. By selecting a node in the Project Dependency Map a user can zoom in to views that present details of the corresponding project.

2.2 Project-Centric - Ecosystem as Context

Understanding the individual projects in the context of the ecosystem.

One set of questions that are not answered by the holistic ecosystem viewpoints are the ones that regard the role of a given project in the ecosystem. Why are there dependencies to a given project from the others in the ecosystem? Which parts of the project are used by the other projects in the ecosystem?

In SPO, the user can answer this type of questions, and similar ones, by diving into the details of an individual project.

Figure 3 presents a visual representation of the architecture of Moose (the large, dark project from Figure 2) in the context of the ecosystem.

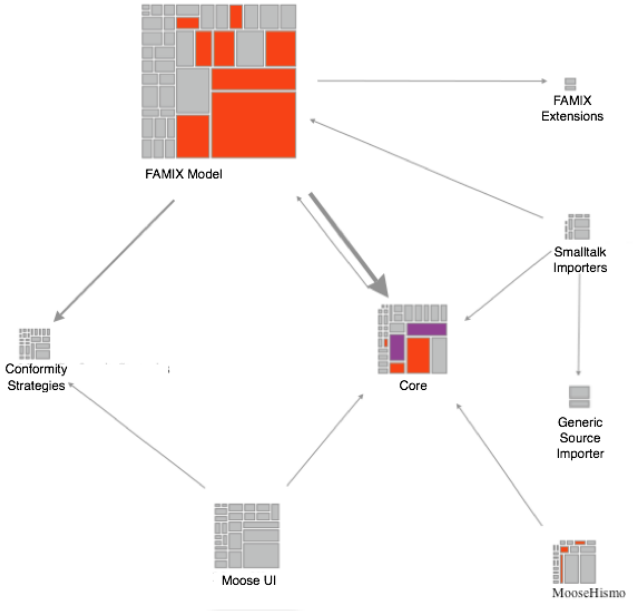

Figure 3: The interaction of the Moose project with the ecosystem (classes with called methods are red, classes that others inherit from are violet)

The modules are represented as nodes in the graph. The area of the node is proportional to the size of the corresponding module. Inside the modules the contained classes and sub-modules are represented as boxes with areas that are proportional to their respective sizes. The dependencies between modules are represented as the edges in the graph. The width of an edge is proportional to the strength of the dependency. The classes that contain methods invoked by other classes in the ecosystem are highlighted with red. The classes that are subclassed by other classes in the ecosystem are highlighted with blue. The classes that are both invoked and subclassed are highlighted in violet.

Analyzing a project in the broader context of the ecosystem can provide useful information that otherwise would not be available. Zooming in to the details of the Moose project that we have encountered earlier allows us to discover the reason for the dependency between the other projects in the ecosystem and it.

2.3 Developer-Centric - Ecosystem as Focus

Supporting a holistic understanding of the ecosystem based on the collaboration structure of its developers.

One of the most interesting and in the same time least explicit aspects of an ecosystem is the social structure that emerges as a result of the collaboration between the contributors to the ecosystem.

SPO provides the *Developer Collaboration Map*, a viewpoint, intended mainly for managers, which shows how developers collaborate with each other within an ecosystem. We consider that two developers collaborate on a project if they both make modifications to the project for number of times greater than a given threshold. Based on this definition of collaboration we construct a *collaboration graph* in which the nodes are developers and the edges represent collaboration relations on specific projects.

Figure 4 presents the visual representation of the collaboration graph in the Soops case study. For more details we refer the reader to [4].

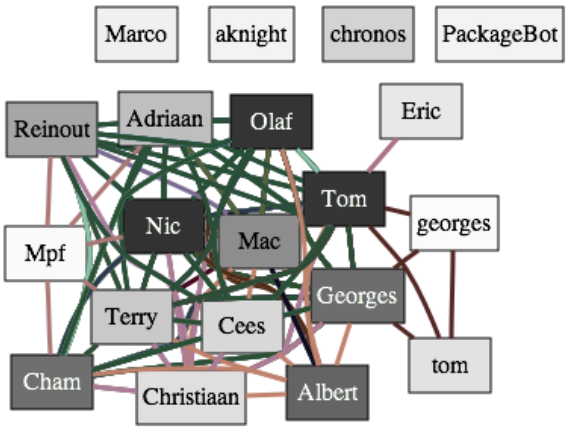

Figure 4: The collaboration structure in the Soops case study shows an ecosystem in which everybody collaborates with everybody on multiple projects

The visual conventions are the following:

- The nodes in the graph are developers; the color intensity of each node is proportional to the amount of commits the corresponding developer has performed to the ecosystem.
- The edges represent collaboration relationships; each edge is color coded to represent the collaboration on a given project.

The figure presents an ecosystem where, with the exception of four contributors (i.e. Marco, aknight, chronos, PackageBot), the developers are tightly connected in the collaboration graph.

SPO allows navigating to detailed views of a given developer by selecting a node in the graph and to the details of a given project by selecting an edge in the graph.

Figure 5: The vocabulary of a developer from one of our case studies

2.4 Developer-Centric - Ecosystem as Context

Understanding each individual developer's contribution to the ecosystem.

Understanding developers in the context of the ecosystem means understanding the type and importance of each developer's contributions. One aspect of this is understanding the expertise of the individual developers.

Each contributor to the ecosystem has a certain type of expertise based on the domain knowledge that he has. When a large number of developers contribute to the projects in the ecosystem it is easy to lose track of each individual developer's level of expertise. This is especially the case in large open-source ecosystems (e.g., RubyForge, SqueakSource, etc.).

SPO can automatically recover the domain of expertise of the individual developers by extracting all the pieces of code to which each developer contributed during his activity in the ecosystem and analyzing the natural language terms that the developer used for the various identifiers. The details of the approach are presented elsewhere [6, 5].

Figure 5 presents the developer expertise tag cloud for one of the developers (Malnati) in our Reveal case study.

One observation that resulted from our work on developer expertise was that many of the terms that appear frequently in the vocabulary of a developer occur in the vocabularies of all the developers in the ecosystem (e.g., controller). By enriching the analysis with information about the entire ecosystem, such terms can be filtered out from the vocabulary of each developer, offering a crisper image of each developer's domain of expertise.

In SPO, the tag cloud is interactive in the sense that selecting a term allows for searching for other developers that have that term in their vocabulary.

3. TOOL INFORMATION

SPO is the first application that supports ecosystem analysis and visualization for reverse engineering. It is a free and open source web application[1] written in Smalltalk. SPO received the annual Technology Innovation Award at the 15th International Smalltalk Conference. SPO was used both in academia and industry to analyze software ecosystems.

[1]We summarized our experiences in writing web-based visualization-intensive reverse engineering tools in [1].

We have evaluated the usability of the tool with graduate students at the University of Lugano and we discovered that they were in general satisfied with the tool, but wanted more responsiveness and more flexibility. We reported more on the usability study elsewhere [3] and we are still improving the tool. In the future, we plan to deploy the tool in industrial contexts to be able to receive feedback on its usefulness.

A deployment of SPO is accessible at `http://inf.unisi.ch/`. It includes three ecosystem case studies that can be freely analyzed as well as other documentation and installation instructions.

Acknowledgments: We gratefully acknowledge the financial support of the Swiss National Science foundation for the project "DiCoSA" (SNF Project No. 118063).

4. REFERENCES

[1] M. D'Ambros, M. Lungu, M. Lanza, and R. Robbes. Promises and perils of porting software visualization tools to the web. In *Proceedings of WSE 2009 (11th IEEE International Symposium on Web Systems Evolution)*, pages 109–118. IEEE CS Press, 2009.

[2] M. Lungu. *Reverse Engineering Software Ecosystems*. PhD thesis, University of Lugano, Switzerland, Oct. 2009.

[3] M. Lungu, T. Gîrba, and M. Lanza. The small project observatory: Visualizing software ecosystems. *EST special issue of the Science of Computer Programming. DOI:10.1016/j.scico.2009.09.004*, 2009.

[4] M. Lungu, M. Lanza, T. Gîrba, and R. Heeck. Reverse engineering super-repositories. In *Proceedings of WCRE 2007 (14th IEEE Working Conference on Reverse Engineering)*, pages 120–129. IEEE CS Press, 2007.

[5] M. Lungu, J. Malnati, and M. Lanza. Visualizing gnome with the small project observatory. In *Proceedings of MSR 2009 (6th IEEE Working Conference on Mining Software Repositories)*, pages 103–106. IEEE CS Press, 2009.

[6] J. Malnati. Developer-centric analysis of svn ecosystems. Master's thesis, University of Lugano, June 2009.

[7] O. Nierstrasz, S. Ducasse, and T. Girba. The story of moose: an agile reengineering environment. *SIGSOFT Softw. Eng. Notes*, 30(5):1–10, 2005.