EvolTrack: A Plug-in-Based Infrastructure for Visualizing Software Evolution

Cláudia Werner[1], Leonardo Murta[2], Marcelo Schots[1], Andréa M. Magdaleno[1,3], Marlon Silva[1], Rafael Cepêda[1], Caio Vahia[1]

[1] Programa de Engenharia de Sistemas e Computação (PESC) – COPPE/UFRJ
Caixa Postal 68.511 – 21945-970 – Rio de Janeiro, RJ – Brasil

[2] Instituto de Computação – Universidade Federal Fluminense
Rua Passo da Pátria 156 – 24210-240 – Niterói, RJ – Brasil

[3] Núcleo de Pesquisa e Prática em Tecnologia (NP2Tec) — Universidade Federal do Estado do Rio de Janeiro (UNIRIO) – 22290-240 – Rio de Janeiro, RJ – Brasil

{werner,schots,andrea,marlon,rcepeda}@cos.ufrj.br,
leomurta@ic.uff.br, caiovahia@poli.ufrj.br

Abstract. *Researchers and practitioners have looked for technologies and methodologies to help monitoring and controlling software development. As software evolves and becomes more complex, it needs to deal with more complex and abundant data. This work provides an overview of EvolTrack, an infrastructure that exploits the Software Visualization discipline for supporting software evolution control and monitoring activities.*

1. Introduction

Software maintenance and evolution control activities emerge as two major areas of a computer system lifecycle. The former concerns day-to-day changes implemented into a software system; the latter refers to what happens to software in long-term, during its entire life span [Jarzabek 2007]. In short, the main focus of both is on changes that occur during the system lifecycle. However, as stated by Parnas (2001), managing the inherent and continuous software changes becomes difficult in the course of time due to several reasons, including the comprehension of those changes.

For a proper monitoring of these activities, a significant amount of data must be collected, processed, and stored over time. Nevertheless, the value of such data for an organization depends, amongst others, on the possibilities to extract and understand the underlying information from these data, so that it can be used to control and improve the development process. To achieve this goal, data must be presented in an intuitive way, and unnecessary information overloading must be avoided.

In order to understand changes that occur during software maintenance and evolution, software visualization techniques have been pointed out as a promising solution for supporting better comprehension of complex systems. This field seeks to investigate and develop abstractions and computational methods for representing various software aspects, such as its structure, behavior, and evolution [Diehl 2007].

Visualization tools provide ways to convert data into visualizations, focusing on

what the data represent [Sogeler 2006], in order to improve the perception and cognitive capacity of human beings for reducing the complexity of software. Such tools become even more useful when they operate in a development environment that can provide integration with other tools and resources.

In this sense, this paper provides an overview of the EvolTrack suite, a plug-in-based infrastructure for analyzing and visualizing software from multiple viewpoints, aiming at bringing a better understanding of software evolution. This paper is organized as follows: Section 2 describes the EvolTrack architecture; Section 3 presents some of the plug-ins developed on top of the EvolTrack infrastructure; Section 4 discusses some related work; and Section 5 shows the final remarks, some limitations, and future work.

2. EvolTrack Infrastructure Overview

According to Telea *et al*. (2010), a visualization tool should be able to support several data types and provide means to compare, correlate, and examine data and views, besides allowing integration with existing tools and being flexible, customizable, and scalable. The authors affirm that some of the key requirements to a software visualization tool include: (i) automatic extraction of data from repositories, (ii) automatic generation of visualization, and (iii) scalability of data to be processed.

Despite these multipurpose recommendations, the evolutionary aspect has its own restrictions. Visualizing the evolution of software is not an easy task, since the addition of the time dimension implies the inclusion of extra data and representations. Caserta & Zendra (2010) state that it is important to keep an overview of the evolution, as it allows the team to understand the current status of a software project.

In this context, the EvolTrack infrastructure[1] aims at tracing and visualizing the evolution of a software project under different perspectives, that is, according to the task viewpoint. It was developed by the Software Reuse[2] team at COPPE/UFRJ, and its architecture was designed to offer flexibility and extensibility, allowing the creation and customization of data sources, transformers, and views. EvolTrack's architecture is composed by four main components and is schematically represented in Figure 1:

- **Datasource Connector**, which is responsible for extracting historical project information from a specific kind of data source and generate a model out of it. A typical example of these data sources is a version control system (VCS);
- **Model Transformer (optional)**, which adds information to the model, enriching this model with project details (e.g., the number of committers of a given model element, the cyclomatic complexity of a given method etc.);
- **Kernel**, whose main purpose is to manage the information extracted from the Datasource Connector, keeping its traceability and orchestrating the flow of information to be presented. It also contains a persistence mechanism and specifies required interfaces for Datasource Connectors and View Connectors;
- **View Connector**, which transforms the information gathered over time into visual abstractions in order to facilitate its understanding.

[1] Site: http://reuse.cos.ufrj.br/evoltrack/
[2] Site: http://reuse.cos.ufrj.br/

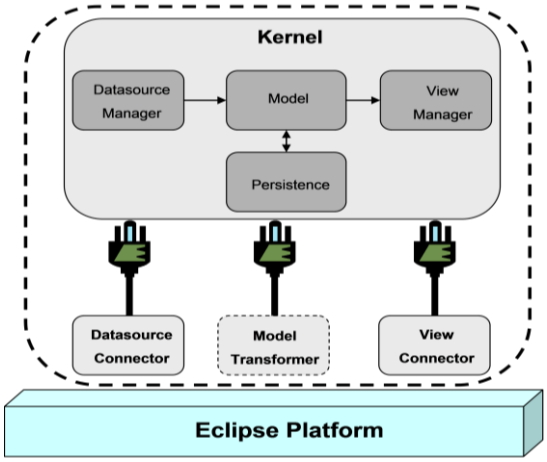

Figure 1. EvolTrack's Architecture

Summarizing, the process is as follows: the Datasource Connector transforms the project information into a model. The Kernel may delegate the model to Transformers, if there are any available. Finally, the model is mapped into visual abstractions by the View Connector, which displays the results to the user.

3. EvolTrack Plug-ins

EvolTrack's extensible architecture enables the construction of plug-ins to be used as Datasource Connectors, Transformers, or View Connectors. Currently, EvolTrack has four active plug-ins that offer three complementary perspectives (i.e., metrics, software architectures, and social networks) regarding software evolution analysis. They are summarized in Table 1 and detailed in the following subsections.

Table 1. EvolTrack Plug-ins

Plug-in name	Type	Features
EvolTrack-VCS [Silva 2010]	Data source connector	Collects data from VCS repositories and transforms source code into UML models
EvolTrack-PREViA [Oliveira 2011]	Transformer	Compares models and gathers metrics of precision and recall between them
EvolTrack-MetricEView [Silva 2010]	Visualization connector	Presents the evolution of software metrics as speedometers, historical graphs etc.
EvolTrack-SocialNetwork [Vahia *et al*. 2011]	Visualization connector	Allows social network visualization and analysis

3.1. EvolTrack-VCS

In order to gather historical information from software configuration management repositories (which stores the history of projects), a version control system (VCS) is needed. However, each VCS implementation would require a specific data source connector. This motivated the development of a new connector, called EvolTrack-VCS [Silva 2010]. It makes use of the Maven SCM API[3], which provides mechanisms for accessing configuration management repositories via generic interfaces.

[3] Site: http://maven.apache.org/scm/

EvolTrack-VCS aims to provide extensibility, so that other VCSs can be added over time, requiring only the implementation of some interfaces. Currently, the connector communicates with 12 popular (commercial and open-source) VCSs.

This component can operate in two modes of extraction: (i) real-time mode, in which the connector searches for a new version (if any) in the repository to add it immediately to the evolution flow, and (ii) traditional mode (real-time option off), only working with previously selected versions. Figure 2 shows the connector's configuration screen, with the data required for its operation, including the selection of the VCS type (a), the option of real-time extracting (b), and the revision browser (c).

After capturing this information, the connector performs reverse engineering of the obtained source code and builds, by default, a class diagram that represents the project's version, including its packages, classes, interfaces and relationships. Note that the EvolTrack-VCS connector has an extensible architecture that allows easy adoption of additional diagrams and visual representations. The current version is able to reverse engineer Java code, but it can be extended to other languages or notations via the substitution of the reverse engineering parser.

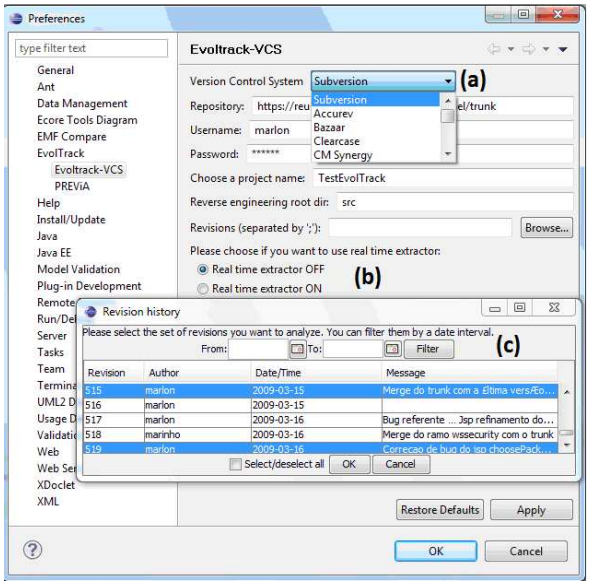

Figure 2. Connector's configuration (a, b, c)

3.2. EvolTrack-PREViA

According to Caserta & Zendra (2010), the visualization of software architecture evolution is an important topic in evolution visualization. In this sense, the PREViA approach [Oliveira 2011] uses mechanisms for model comparison, metric extraction, and software visualization for analyzing differences between software models.

This approach aims to: (i) provide a better perception of the adherence between what is being implemented (emerging model) with respect to what has been designed (conceptual model), (ii) provide a better perception of the evolution of the implementation (emerging model), and (iii) provide a better perception of the evolution of software design (conceptual model). From the adherence perspective, potential problems in sketching conceptual models and/or in the implementation of such models

can be identified. Precision and recall metrics were adapted for dealing with this scenario, representing exactness and completeness of models, respectively.

The EvolTrack-PREViA plug-in implements the two before mentioned visualization perspectives: the adherence of the emerging models to a conceptual model over time, and the evolution process by means of differences found between versions of the same level of abstraction. Note that both perspectives seek to obtain the perception of differences between models, but these are presented to the user under different points of view. An example of the adherence perspective can be seen in Figure 3, where divergent elements are highlighted (center panel) and can be marked as implementation-specific by the user (lower panel). UML Profiles are used for including additional information to the models (e.g., precision and recall values).

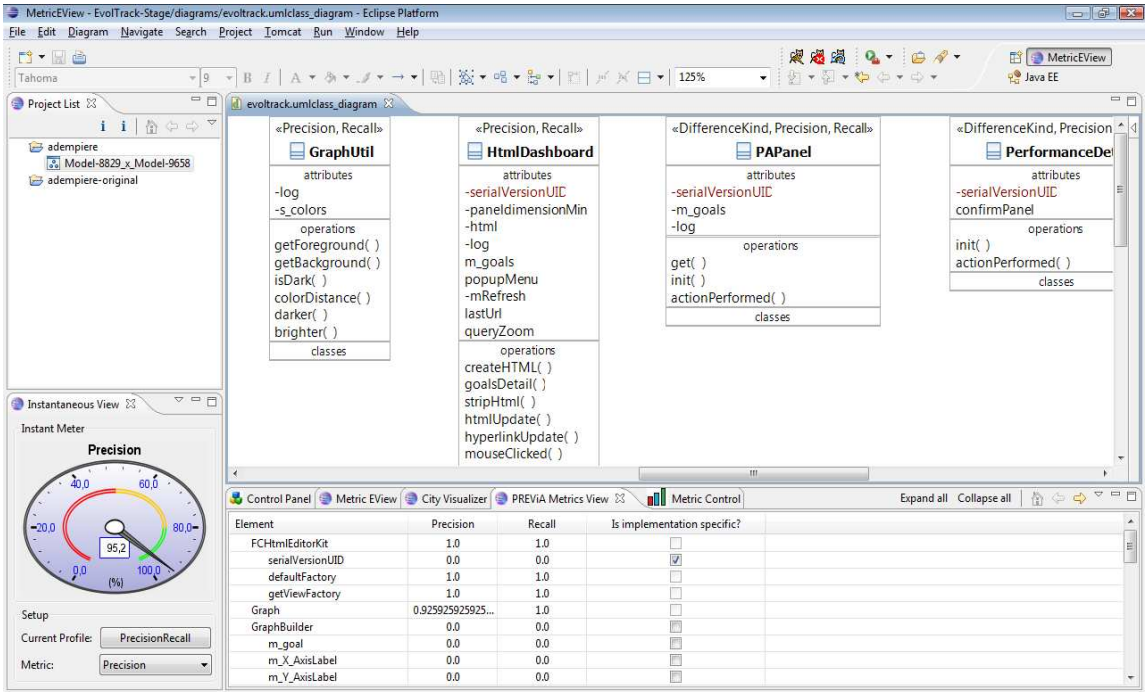

Figure 3. EvolTrack-PREViA and MetricEView (with speedometer)

3.3. EvolTrack-MetricEView

The MetricEView connector [Silva 2010] is a set of views that can represent different data formats, regardless of the data source. By using the data (i.e., metrics values) obtained by a Datasource Connector (optionally handled by a Transformer), the information about the metrics' evolution is presented in three view modules that make use of visualization techniques for showing software evolution through metrics. Such metrics can be presented both individually, to focus on an instantaneous software property (see the speedometer in Figure 3), and together, to historically analyze the relation between different software attributes.

3.4. EvolTrack-SocialNetwork

EvolTrack-SocialNetwork[4] [Vahia *et al*. 2011] aims to provide awareness of how

[4] Site: http://reuse.cos.ufrj.br/evoltrack/socialnetwork

collaboration happens among members of a software development team. It provides features to enhance visualization and analysis of these networks. Figure 4 shows an example of the approach, using a network extracted from a real project (Floggy[5]).

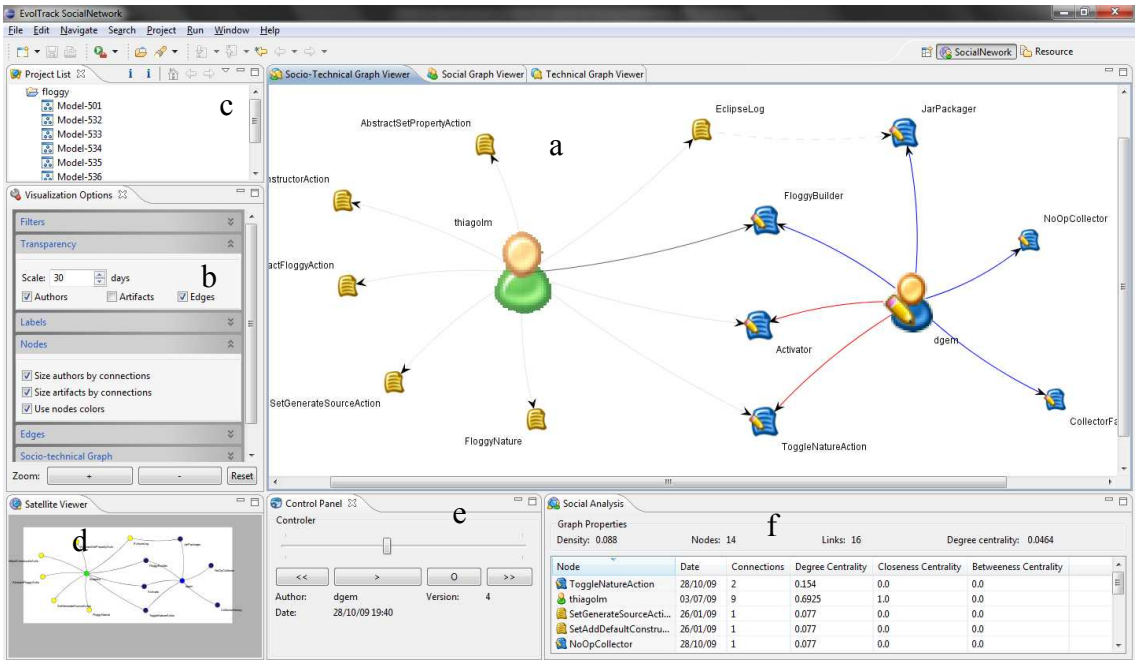

Figure 4. EvolTrack-SocialNetwork

This plug-in provides the visualization of technical, socio-technical, and social networks. In these networks, the authors are represented by a person icon and, for the artifacts, a document icon is adopted. These icons are modified (regarding colors and graphics) to illustrate when a node is added to the network (the icon with + sign) or when it has undergone some modification (pencil icon to edit) (Figure 4.a).

Each tab in the main view (Figure 4.a) corresponds to a network (technical, socio-technical, and social). In the options panel (Figure 4.b), it is possible to apply filters, transparency, and so on. Above this panel (Figure 4.c), the user can choose which model to view in the project list. At the bottom left, there is a satellite view (Figure 4.d), where a thumbnail of the graph is shown for easier navigation. Finally, at the bottom central (Figure 4.e) and right (Figure 4.f), there is a control panel for navigating between versions of the model, along with an analysis panel to monitor network metrics.

4. Related Work

Aiming at uncovering project structure and understanding how it changes over time, Froehlich & Dourish (2004) developed the Augur tool that helps exploring lines of code to create a visualization of the software structure. Therefore, the tool associates each line of code with a set of information of interest, such as to what activity it relates, to which method or class it belongs, and who was responsible for its creation.

EvoLens [Ratzinger *et al*. 2005] enables a user to view the evolution of object-

[5] Site: http://floggy.sourceforge.net/

oriented systems from multiple dimensions. It supports data from software developed in Java and kept in a CVS repository. Evolution happens when a check-in happens whilst the hierarchy of the software is represented by its own structure. The visualization of the structure of the software is based on graphs and colors. Additionally, a lens-based visualization model is used to reduce the amount of information presented to the user.

Wettel & Lanza (2008) use a 3D city metaphor to explore the evolution of object-oriented software systems. Classes are depicted as buildings located in districts (packages). Metrics are used to define the visual properties of the artifacts displayed.

SourceMiner [Carneiro *et al.* 2010] is a tool based on multiple views and interactive techniques for improving software comprehension activities. It presents instantaneous data, but does not allow visualizing the project history over time.

These related works use different software visualization metaphors, such as graphs and 3D views. Additionally, they do not have the high flexibility provided by EvolTrack, using models as common communication language annotated with metrics as arbitrary visual properties. Another distinguished feature of EvolTrack is the ability for real-time monitoring of project development, allowing the playback of any period in the development history of the software.

5. Final Remarks

This paper presented an overview of EvolTrack, a software evolution infrastructure embedded with mechanisms that support the capture of software projects' data from multiple data sources. It also provides visual representations of different aspects of software, increasing the awareness of evolution in software projects. One of the main benefits of such infrastructure is its architectural flexibility, which allows other analysis and visualization approaches to be built on top of it, providing a high level of reuse.

EvolTrack was evaluated by analyzing its feasibility, performance, and usability in open source projects [Cepeda *et al*. 2010], the perception of differences [Oliveira 2011], and support for modeling education [Schots *et al.* 2010]. Results provided significant improvements for the infrastructure and its plug-ins, as well as opportunities for future work.

The current architecture allows the execution of only one data source at a time, i.e., the combination of data sources is not possible yet. Another constraint is that the kernel uses the UML metamodel, i.e., a mapping for UML model elements is required when a data source outputs another model representation. There is work in progress for migrating the kernel to the Ecore representation, which is a more general one.

There is a wide range of possibilities to enhance EvolTrack mechanisms. Regarding the source of information, other types of data source can be implemented to retrieve data from other repositories, such as bug trackers, emails, and forums. Also, the implementation of other visualization techniques can enable the handling of larger data sets. Finally, we intend to conduct more studies on the feasibility and applicability of the infrastructure. These studies can be performed in the context of open source projects and other real analysis scenarios.

Acknowledgments

This work was financially supported by CNPq, CAPES, and FAPERJ.

References

Carneiro, G., Roberto Júnior, P., Nunes, A., Mendonça, M. (2010), "An Eclipse-Based Multi-Perspective Environment to Visualize Software Coupling". In: *Congresso Brasileiro de Software (CBSoft) – Sessão de Ferramentas*, pp. 1-6, Salvador, Brasil.

Caserta, P., Zendra, O. (2010), "Visualization of the Static Aspects of Software: A Survey", *IEEE Transactions on Visualization and Computer Graphics*, pp. 1-20.

Cepeda, R. D. S. V., Magdaleno, A. M., Murta, L. G. P., Werner, C. M. L. (2010), "EvolTrack: Improving Design Evolution Awareness in Software Development", *Journal of the Brazilian Computer Society (JBCS)*, 16, 2, pp. 117-131.

Diehl, S. (2007), *Software Visualization: Visualizing the Structure, Behaviour, and Evolution of Software*, 1 ed., Springer.

Froehlich, J., Dourish, P. (2004), "Unifying Artifacts and Activities in a Visual Tool for Distributed Software Development Teams". In: *26th International Conference on Software Engineering*, pp. 387-396, Scotland, UK.

Jarzabek, S. (2007), *Effective Software Maintenance and Evolution: A Reuse-Based Approach*. Auerbach, CRC Press Taylor and Francis.

Oliveira, M. S. (2011), "*PREViA: An Approach for Visualizing the Evolution of Software Models*" (in Portuguese), Master Thesis, COPPE/UFRJ, 185p.

Parnas, D. L. (2001), *Software Fundamentals*, Addison-Wesley Longman Publishing.

Ratzinger, J., Fischer, M., Gall, H. (2005), "EvoLens: Lens-View Visualizations of Evolution Data". In: *8th International Workshop on Principles of Software Evolution (IWPSE)*, pp. 103-112, Lisbon, Portugal.

Schots, M., Rodrigues, C. S., Werner, C., Murta, L. (2010), "A Study on the Application of the PREViA Approach in Modeling Education". In: *XXIX International Conference of the Chilean Computer Society*, pp. 96-101, Antofagasta, Chile.

Silva, M. A. (2010), "*IAVEMS: Infrastructure for Supporting Evolution Visualization of Software Metrics*" (in Portuguese), Undergraduate Project, UFRJ, 98p.

Sogeler, D. (2006), "*Analysis and Improvement of Data Handling Performance of a Visualization Tool*", Master Thesis, Technische Universiteit Eindhoven, 79p.

Telea, A., Voinea, L., Sassenburg, H. (2010), "Visual Tools for Software Architecture Understanding: A Stakeholder Perspective", *IEEE Software*, 27, 6, pp. 46-53.

Vahia, C. M., Magdaleno, A. M., Werner, C. M. L. (2011), "EvolTrack-SocialNetwork: Uma Ferramenta de Apoio à Visualização de Redes Sociais". In: *Congresso Brasileiro de Software (CBSoft) – Sessão de Ferramentas*, São Paulo, Brasil (in Portuguese) (to appear).

Wettel, R., Lanza, M. (2008), "Visual Exploration of Large-Scale System Evolution". In: *Working Conference on Reverse Engineering (WCRE)*, pp. 219-228, Antwerp, Belgium.