Science of Computer Programming 75 (2010) 276–287

Contents lists available at ScienceDirect

Science of Computer Programming

journal homepage: www.elsevier.com/locate/scico

Distributed and Collaborative Software Evolution Analysis with Churrasco

Marco D'Ambros *, Michele Lanza

REVEAL, Faculty of Informatics, University of Lugano, Switzerland

ARTICLE INFO

Article history:
Received 27 November 2008
Received in revised form 20 May 2009
Accepted 22 July 2009
Available online 18 August 2009

Keywords:
Software evolution analysis
Collaboration
Visualization

ABSTRACT

Analyzing the evolution of large and long-lived software systems is a complex problem that requires extensive tool support due to the amount and complexity of the data that needs to be processed. In this paper, we present *Churrasco*, a tool to support *collaborative software evolution analysis* through a web interface. After describing the tool and its architecture, we provide a usage scenario of Churrasco on a large open source software system, and we present two collaboration experiments performed with, respectively, 8 and 4 participants.

1. Introduction

Software evolution analysis is concerned with the causes and the effects of software change. There is a large number of approaches, which all use different types of information about the history and the (evolving) structure of a system. The overall goal is, on the one hand, to perform retrospective analysis, useful for a number of maintenance activities, and, on the other hand, to predict the future evolution of a system. Such analyses are intrinsically complex, because modeling the evolution of complex systems implies:

1. The retrieval of data from software repositories, managed by software configuration management systems such as CVS or SVN,
2. The parsing of the obtained raw data to extract relevant facts and to minimize the noise that such large data sets exhibit, and
3. The population of models that are then the basis for any analysis. Tools supporting software evolution analysis should hide these tasks from the users, to let them focus on the actual analysis.

Moreover, such tools should provide means to break down information complexity, typical for large and long-lived software systems. We argue that any software evolution analysis tool should possess the following characteristics:

> **Flexible Meta-model**. Several, and largely similar, approaches have been proposed to create and populate a model of an evolving software system, considering a variety of information sources, such as the histories of software artifacts (as recorded by a versioning system), the problem reports stored by systems such as Bugzilla [1], e-mail archives, user documentation [2], etc. Even if such models are appropriate for modeling the evolution, they are "hard-coded" in the sense that their creators took deliberate design choices in accordance with their research goals. We postulate that *software evolution tools should be flexible with respect to the underlying meta-model*: If the meta-model is changed or extended because some new type of information is at hand, or because some new analysis is required, the tool should adapt itself to the new meta-model.

* Corresponding author. Tel.: +41 58 666 4758; fax: +41 58 666 4536.
E-mail addresses: marco.dambros@usi.ch (M. D'Ambros), michele.lanza@usi.ch (M. Lanza).

0167-6423/$ – see front matter © 2009 Elsevier B.V. All rights reserved.
doi:10.1016/j.scico.2009.07.005

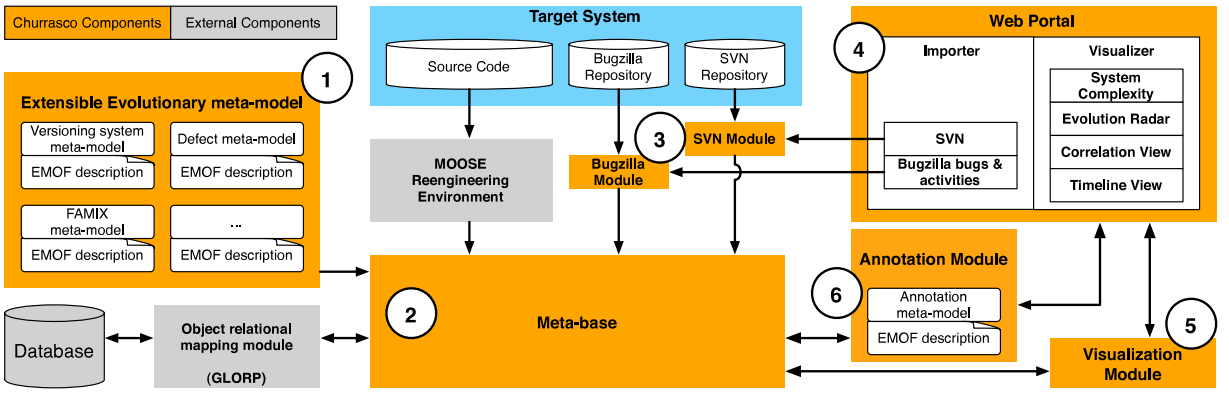

Fig. 1. The architecture of Churrasco.

Accessibility. Researchers have developed a plethora of evolution analysis tools and environments. One commonality among many prototypes is their limited usability, *i.e.*, often only the developers themselves know how to use them, thus hindering the development and/or cross-fertilization of novel analysis techniques. There are some notable exceptions, such as Moose [3], which have been used by a large number of researchers over the years. Researchers also investigated ways to exchange information about software systems [4,5], approaches which, however, are seldom followed up because of lack of time or manpower. We argue that *software evolution tools should be easily accessible*: They should be usable from any machine running any operating system, without any strings attached.

Incremental Storage of Results. Results of analyses and findings on software systems produced by tools are often written into files and/or manually crafted reports, and are therefore of limited use. We claim that *analysis results should be incrementally and consistently stored back into the analyzed models*: This allows researchers to develop novel analyses that exploit the results of a previous analysis (cross-fertilization of ideas/results). It can also serve as a basis for a benchmark for analyses targeting the same problem, and ultimately would also allow one to combine techniques targeting different problems.

Support for Collaboration. The need of collaboration in software development is getting more and more attention. Tools which support collaboration, such as Jazz for Eclipse [6], were only recently introduced, but hint at a larger current trend. Just as the software development teams are geographically distributed, consultants and analysts are too. Specialists in different domains of expertise should be allowed to collaborate without the need of being physically present together. Because of these reasons, we argue that software evolution analysis should be a collaborative activity. As a consequence, *software evolution analysis tools should support collaboration*, by allowing different users, with different expertise, from different locations, to collaboratively analyze a system.

We present *Churrasco* [7], a tool for collaborative software analysis, which is available at http://churrasco.inf.unisi.ch. Churrasco has the following characteristics:

- It hides all data retrieval and processing tasks from the users, to let them focus on the actual analysis, and provides an easily accessible interface over a web browser to model the data sources to be analyzed.
- It copes with modeling and populating problems by providing a flexible and extensible object-relational persistence mechanism. Any data meta-model can be dynamically changed and extended, and all the data is stored in a central database.
- It provides a set of collaborative visual analyses and supports collaborative analysis by allowing users to annotate the analyzed data.
- It stores the findings into a central database to create an incrementally enriched body of knowledge about a system, which can be exploited by subsequent users.

Structure of the paper. In Section 2 we describe the Churrasco framework, its architecture, and its main components. We then provide an example of a collaborative session and describe two collaboration experiments performed with Churrasco (Section 3). We discuss our approach in Section 4 and examine tool building issues in Section 5. We survey related work in Section 6, and conclude in Section 7 with a summary of our contributions and directions of future work.

2. Churrasco

Fig. 1 depicts Churrasco's architecture, consisting of:

1. *The Extensible Evolutionary meta-model* describes the internal representation of software systems' evolution, which can be extended using the facilities provided by the Meta-base module.