# A Multi-view Toolkit to Assist Software Integration and Evolution *

Kamran Sartipi
Dept. Computing and Software, McMaster University, Hamilton, ON, L8S 4K1, Canada
*sartipi@mcmaster.ca*

**Abstract**. *Software product line engineering aims at producing functionally similar software systems as a family of products. In this process, the development life cycle has been shifted from traditional activities into reuse-centric and customizable component approaches. In software product line engineering from existing systems many technical, organizational, and user-adoption problems need to be dealt with by the means of proper tool support. In this context, we provide a supporting toolkit that blends the behavior and structure recovery techniques in order to localize the major components of the existing systems as candidates for generic or reusable components. An integration of the reusable and new components will form a domain reference architecture, whose instantiation will produce the products. For each new product the scattering of the added features in the structure of the original components will be determined by the means of two metrics to assess the functional cohesion and feature functional scattering. This allows us to control the structural evolution of the product line engineering process.*

## 1. Introduction

Modern industrial organizations in different application domains are deeply involved in various software engineering tasks such as: developing new systems, integrating legacy assets with modern applications, decentralizing monolithic systems, and performing various maintenance activities. In order to sustain their competitiveness in industry, these organizations require a well maintained high-quality software system to continue their business without any interruption caused by software failure. In this context, a multi-view and interactive assistant-tool would be extremely valuable in order to identify the intended software system components, and to leverage the knowledge of the software experts about the impact of the integrated features on the system structure. As an example of a software product line engineering organization, the Fraunhofer IESE defines its PuLSE (Product Line Software Engineering) process using the following stages [2]: i) study the existing enterprise system and its domain to compose a version of the PuLSE that is suited for that specific domain and also compose a reference architecture; ii) specify and generate a member of the product family by extracting generic components and adding new features using new technologies; and iii) monitor and control the evolution and management of the product family infrastructure.

In this paper, we describe the characteristics of our view-based software analysis and evaluation toolkit (namely *Alborz*) that assists a product line engineering process (e.g., PuLSE) in various ways, such as: i) *identifying generic components*: by executing important task scenarios on the system and applying dynamic analysis, where the core implementation of the reusable components within the source code are identified with minimum effort; ii) *extracting generic components*: by augmenting the core implementations into cohesive components that include the group of highly related functionality; iii) *evaluating the merit of the final products*: by measuring feature functionality scattering throughout the system's structure as a means to assess the impact of the new features on the structure of the system, as well as measuring the cohesion of the system in term of overall association of the components.

## 2. Highlights of the toolkit

Alborz is a multi-view, interactive, and wizard-based software architecture reconstruction and evaluation toolkit that takes advantage of the Eclipse plug-in technology to provide feature extensibility, and uses GXL format to interoperate with other reverse engineering tools. Structure recovery is based on static pattern matching and high-level architectural queries. Behavior recovery is based on execution pattern extraction and identification of software features in the source code using data mining techniques.

**Scalability of the dynamic analysis**. Alborz toolkit provides three different techniques to deal with large execution traces. The first technique is based on eliminating the loop-based repetitions in a trace by representing the initial sequence of function entry/exit pairs as *dynamic call tree* that allows us to perform a top-down program-loop elimination operation [6]. The second technique is based on the

*This work was funded by Natural Sciences and Engineering Research Council of Canada (NSERC).

data mining algorithm *sequential pattern discovery* [1] that extracts frequently occurring traces that allow us to locate redundant traces such as program-loops or common software operations (e.g., mouse movement, user-interface interactions, and utilities). Data mining operations by nature generate a large number of patterns, most of which are substrings of a larger pattern, hence, a further operation for subtrace elimination is required to obtained unique traces. The third technique is based on *string manipulation* algorithms that allow us to identify repetitive patterns in an string of elements.

**Blend of dynamic and static analysis**. The combination of dynamic analysis with static analysis of software system is considered as a very promising approach [3, 7, 4]. These techniques usually produce independent views, not an integration of both views. Alborz toolkit allows us to inject software behavioral information (stemmed from task scenarios or use cases) into static analysis of the system in order to adjust the static analysis to produce more sensible results. This approach uses feature to source code assignment to localize the core functions that implement specific software operations (features) [6] and then uses the core functions as the seeds in a software clustering technique to collect functions into software modules or components that correspond to specific operations of the software system [4].

**Discovery of patterns**. The application of techniques such as data mining and concept lattice analysis reveals patterns of relations among the software entities. Specific data mining techniques such as association rules discovery has direct application in static analysis [5] and sequential pattern discovery has been applied on dynamic analysis for locating the implementation of software features in source code [6]. The application of other data mining techniques on software analysis is yet to be studied. Very few approaches in dynamic analysis utilize the visualization power of concept lattice techniques. These techniques must handle the inherent characteristic of the lattice in the sense that the lattice easily becomes overwhelmed by the number of generated concepts. One remedy for such a problem is to raise the granularity level of the objects and attributes in defining the context table. Alborz uses concept lattice technique to separate the common execution patterns from specific patterns of a targeted operation [4].

**Usability and extensibility**. Inadequate tool support would cause interesting approaches to become obsolete or being used only by their developers. Much effort is needed to make an interesting approach usable by others. Alborz provides: i) short learning curve that is guided by wizards with clear explanation of their tasks; ii) extensible and interoperable open source technologies available for developing platform independent environments (e.g., Java and XML), as well as tool integration platform (e.g., Eclipse) that is based on plug-in technology. The proposed tool can act a key role in such product family generation in various aspects of program understanding, localizing the features in the source code, and evaluating the merit of the final product.

## 3. Limitations and future work

The current version of Alborz only covers procedural software systems and lacks a fact extractor tool; however, the tool interoperates with the existing fact extractor tools and accepts input data in a variety of formats (e.g., GXL, RSF). In dynamic analysis, the trace extraction is performed through execution of a set of task scenarios; hence, familiarity of the user with the application domain and the subject system is required. The future extensions to the Alborz toolkit include the provision of: i) static and dynamic analysis of object oriented systems; ii) growing the core of a component using the pool of execution patterns generated by sequential pattern discovery technique; hence generating cohesive components based on the frequency of usage not static dependencies; this provides a different view of the component as opposed to growing the component by static analysis; and iii) discovery of useful patterns as well as anti-patterns in both static and dynamic views, in order to be used for generating more maintainable components. So far, we have successfully applied our toolkit on isolated systems such as Xfig drawing tool and Pine email client; however we still need to apply our toolkit on real world software product line projects as the next step.

## References


[1] R. Agrawal and R. Srikant. Mining sequential patterns. In *Proceedings of the International Conference on Data Engineering (ICDE)*, pages 3–14, 1995.

[2] J. Bayer et al. Pulse: A methodology to develop software product lines. In *Proceedings of ACM SIGSOFT Symposium on Software Reusability (SSR'99)*, pages 122–131, May 1999.

[3] C. Riva and J. V. Rodriguez. Combining static and dynamic views for architecture reconstruction. In *Proceedings of the IEEE CSMR*, pages 47–55, 2002.

[4] K. Sartipi, N. Dezhkam, and H. Safyallah. An orchestrated multi-view software architecture reconstruction environment. In *Proceedings of IEEE WCRE'06*, pages 61–70, October 2006.

[5] K. Sartipi and K. Kontogiannis. A user-assisted approach to component clustering. *Journal of Software Maintenance: Research and Practice (JSM)*, 15(4):265–295, July/August 2003.

[6] K. Sartipi and H. Safyallah. Application of execution pattern mining and concept lattice analysis on software structure evaluation. In *Proceedings of the SEKE'06*, pages 302–308, June 2006.

[7] A. van Deursen, C. Hofmeister, R. Koschke, L. Moonen, and C. Riva. View-driven software architecture reconstruction. In *Proceedings of the IEEE Working Conference on Software Architecture (WICSA'04)*, pages 122–132, 2004.