The Journal of Systems and Software 94 (2014) 161–185

Contents lists available at ScienceDirect

The Journal of Systems and Software

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

A systematic review of software architecture visualization techniques

Mojtaba Shahin[a,b], Peng Liang[a,c,*], Muhammad Ali Babar[d]

[a] State Key Lab of Software Engineering, School of Computer, Wuhan University, China
[b] Department of Computer Engineering, Neyriz Branch, Islamic Azad University, Iran
[c] Department of Computer Science, VU University Amsterdam, Netherlands
[d] CREST – The Centre for Research on Engineering Software Technologies, The University of Adelaide, Australia

ARTICLE INFO

Article history:
Received 31 August 2013
Received in revised form 4 February 2014
Accepted 22 March 2014
Available online 30 March 2014

Keywords:
Software architecture
Software architecture visualization
Visualization techniques

ABSTRACT

Context: Given the increased interest in using visualization techniques (VTs) to help communicate and understand software architecture (SA) of large scale complex systems, several VTs and tools have been reported to represent architectural elements (such as architecture design, architectural patterns, and architectural design decisions). However, there is no attempt to systematically review and classify the VTs and associated tools reported for SA, and how they have been assessed and applied.
Objective: This work aimed at systematically reviewing the literature on software architecture visualization to develop a classification of VTs in SA, analyze the level of reported evidence and the use of different VTs for representing SA in different application domains, and identify the gaps for future research in the area.
Method: We used systematic literature review (SLR) method of the evidence-based software engineering (EBSE) for reviewing the literature on VTs for SA. We used both manual and automatic search strategies for searching the relevant papers published between 1 February 1999 and 1 July 2011.
Results: We selected 53 papers from the initially retrieved 23,056 articles for data extraction, analysis, and synthesis based on pre-defined inclusion and exclusion criteria. The results from the data analysis enabled us to classify the identified VTs into four types based on the usage popularity: graph-based, notation-based, matrix-based, and metaphor-based VTs. The VTs in SA are mostly used for architecture recovery and architectural evolution activities. We have also identified ten purposes of using VTs in SA. Our results also revealed that VTs in SA have been applied to a wide range of application domains, among which "graphics software" and "distributed system" have received the most attention.
Conclusion: SA visualization has gained significant importance in understanding and evolving software-intensive systems. However, only a few VTs have been employed in industrial practice. This review has enabled us to identify the following areas for further research and improvement: (i) it is necessary to perform more research on applying visualization techniques in architectural analysis, architectural synthesis, architectural implementation, and architecture reuse activities; (ii) it is essential to pay more attention to use more objective evaluation methods (e.g., controlled experiment) for providing more convincing evidence to support the promised benefits of using VTs in SA; (iii) it is important to conduct industrial surveys for investigating how software architecture practitioners actually employ VTs in architecting process and what are the issues that hinder and prevent them from adopting VTs in SA.

1. Introduction

With increasing size and complexity of software-intensive systems, the role of software architecture (SA) as a means of understanding and managing large-scale software intensive systems has been increasingly becoming important. The high level design description of a large system can help a system's stakeholders to understand and reason about the designed architecture with regards to architecturally significant requirements (ASRs) of a software-intensive system (Bass et al., 2012). SA community has been developing various approaches, techniques, and tools for improving software architecture communication and understanding among all the key stakeholders of large-scale software-intensive systems. One of the increasingly popular ways of making software architecture design decisions and their

* Corresponding author at: State Key Lab of Software Engineering, School of Computer, Wuhan University, China. Tel.: +86 27 68776137; fax: +86 27 68776027.
E-mail address: liangp@whu.edu.cn (P. Liang).

http://dx.doi.org/10.1016/j.jss.2014.03.071
0164-1212/

rationale easily understandable is visualizing SA (Shahin and Liang, 2010; de Boer et al., 2009; López et al., 2009; Lee and Kruchten, 2008). Visualization in computer graphics is a technique for creating images, diagrams, or animations to communicate information, which may not be easy to describe and understand in other formats, such as textual (Spence, 2000). Visualization transfers information into visual forms and enhances information understanding in software development (Diehl, 2007).

Software visualization is defined as visual representation of artifacts (such as requirements, design, and program code) related to software and its development process (Diehl, 2007). The main motivation for using software visualization is to help stakeholders to understand and comprehend different aspects of software systems during software development process and reduce the cost of software evolution (Diehl, 2007; Gallagher et al., 2008). SA visualization is defined as a visual representation of architectural models and some or all of the architectural design decisions about the models (Taylor et al., 2009). The importance of visualizing SA has been extensively investigated as SA visualization can be of interest to various stakeholders such as architects, developers, testers, and project managers (Gallagher et al., 2008; Sharafi, 2011; Telea et al., 2010; Shahin and Liang, 2010). SA visualization, e.g., decomposition of a software system's architecture into layers, components, or slices in a structural viewpoint, is critical in understanding and communicating the architecture to a variety of project stakeholders (Cleland-Huang et al., 2013). Due to the recognition of the importance of visualizing SA, an increasing amount of literature describing SA visualization approaches and tools has been published through diverse venues (Gallagher et al., 2008; Sharafi, 2011; Telea et al., 2010). However, there has been no dedicated effort to systematically identify and select, and rigorously analyze and synthesize the SA visualization literature. In order to fill this gap, we decided to carry out a systematic literature review (SLR) (Kitchenham and Charters, 2007) of the SA visualization.

This article reports the design, execution, and findings of the SLR that aimed at systematically identifying, selecting, and summarizing a comprehensive set of SA visualization techniques, associated tools, and the supporting evidence published in the peer-reviewed literature. This SLR enabled us to enumerate a comprehensive set of papers describing SA visualization techniques and tools in order to reveal the key motivators for their development, their evolutionary paths, foundational principles, and assessment mechanisms. For this review, we have systematically identified and rigorously reviewed 53 relevant papers and synthesized the data extracted from those papers in order to answer a set of research questions that had motivated this review. We assert that the results from this SLR can provide important benefits to researchers and practitioners from both software architecture as well as software visualization communities. This review can enable them to gain a better understanding of the available SA visualization techniques, their suitability for different architecting activities, the level of evidence reported for each of them, and the gaps that need further research in this area. The two significant contributions of this paper to the software architecture visualization body of knowledge are:

1. It reports the design, execution, and results of a review aimed at systematically identifying a comprehensive set of relevant papers on SA visualization techniques based on pre-defined selection criteria and rigorously analyzing and synthesizing the reported techniques, associate tools, and reported evidence in an easily accessible format.
2. It structures and classifies the reviewed SA visualization techniques and tools, and the available evidence using different formats that are expected to be useful for practitioners interested in using visualization for communicating and understanding SA design and design decisions. The findings can be used as an evidence-based guide to select appropriate SA visualization techniques and tools based on the required suitability for different activities of the software architecting process. The findings also identify the issues relevant to researchers who are interested in knowing the state-of-the-art of and the areas of future research in SA visualization.

1.1. Background and related work

Software architecture has emerged as an important area of software engineering research and practice over the last two decades. The increasing size of, complexity of, and demand for quality in software systems are some of the most important factors that have resulted in sustained interests in SA research and practice. It is widely recognized that a high level design description can play an important role in successfully understanding and managing large and complex software systems (Bass et al., 2012). SA community has developed several methods, approaches, and tools to help understand and reason about high level architecture design. Software architecture can be described and viewed from multiple perspectives. Two of the most commonly used perspectives of SA are architectural viewpoint and architecting process perspective.

There are two distinct viewpoints on SA, structural and decisional (Poort and van Vliet, 2012): the structural viewpoint expresses SA with components and connectors and considers it as a high-level software structure of a system (Bass et al., 2012). This viewpoint mainly focuses on the end products (e.g., components and connectors) of software architecting process. The decisional viewpoint considers decisions made during architecting as the first class entities and defines SA as a set of design decisions, including their rationale (Jansen and Bosch, 2005). In this SLR, both viewpoints of SA have been considered as visualization techniques can be used to support both kinds of viewpoints of SA. The structural elements (e.g., components and connectors) and decisional elements (e.g., decisions) are generally termed as "architectural elements" or "architectural entities" that are interchangeably used in this paper.

Architecting is a process of conceiving, defining, expressing, documenting, communicating, certifying proper implementation of, maintaining and improving an architecture throughout a system's life cycle (ISo, 2011). From an architecting process perspective, software architecting is composed of a set of general and specific activities (Li et al., 2013; Hofmeister et al., 2007), which can be supported by various visualization techniques and tools. The specific architecting activities cover the entire architecture lifecycle and the general architecting activities provide support to achieve the goals of the specific activities of software architecting. For example, architectural evolution, as a specific architecting activity, copes with correcting faults, responding to new changes, and implementing new requirements in architecture. Architecture recovery, as a general architecting activity, examines existing available sources of a system (such as implementation and documentation of a system) to uncover and extract architecture design and design decisions. Architecture recovery can support architecture evolution by recovering the architecture design and design decisions when architecture documentation is not well documented or unavailable, or architectural design decisions have been lost.

Through this review, we are interested to know how various visualization techniques can facilitate these general and specific architecting activities. It is generally considered that SA visualization techniques can be used to support any stage of the software architecting process, i.e., analyzing, synthesizing, evaluating, implementing, and evolving architecture (Telea et al., 2010). In a decisional viewpoint, visualization of architectural design decisions (ADDs) can improve the understanding of ADDs and their rationale, and this kind of understanding becomes more