Information Sciences 305 (2015) 234–255

Contents lists available at ScienceDirect

Information Sciences

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

An approach for the evolutionary discovery of software architectures

Aurora Ramírez, José Raúl Romero *, Sebastián Ventura

Department of Computer Science and Numerical Analysis, University of Córdoba, 14071 Córdoba, Spain

ARTICLE INFO

Article history:
Received 30 July 2014
Received in revised form 6 January 2015
Accepted 24 January 2015
Available online 7 February 2015

Keywords:
Search based software engineering
Software architecture discovery
Evolutionary algorithms
Ranking aggregation fitness

ABSTRACT

Software architectures constitute important analysis artefacts in software projects, as they reflect the main functional blocks of the software. They provide high-level analysis artefacts that are useful when architects need to analyse the structure of working systems. Normally, they do this process manually, supported by their prior experiences. Even so, the task can be very tedious when the actual design is unclear due to continuous uncontrolled modifications. Since the recent appearance of search based software engineering, multiple tasks in the area of software engineering have been formulated as complex search and optimisation problems, where evolutionary computation has found a new area of application. This paper explores the design of an evolutionary algorithm (EA) for the discovery of the underlying architecture of software systems. Important efforts have been directed towards the creation of a generic and human-oriented process. Hence, the selection of a comprehensible encoding, a fitness function inspired by accurate software design metrics, and a genetic operator simulating architectural transformations all represent important characteristics of the proposed approach. Finally, a complete parameter study and experimentation have been performed using real software systems, looking for a generic evolutionary approach to help software engineers towards their decision making process.

1. Introduction

Throughout software development, software engineers need to make decisions about the most appropriate structures, platforms and styles of their designs. The automatic inference and evaluation of different design alternatives is a challenging application domain where computational intelligence techniques serve to provide support to software engineers, especially when limited information about the system being developed is still available.

In this context, architectural analysis constitutes an important phase in software projects, as it provides methods and techniques for handling the specification and design of software in the earlier stages [9]. It is considered a human-centered decision process with a great impact on the quality and reusability of the end product. During high level analysis, component identification allows the discovery of system blocks, their functionalities and interactions. For this reason, it is a good practice when dealing with complex systems [35], resulting in more comprehensible software and making its development and maintenance simpler and more affordable.

Frequently, software engineers need to tackle architectural analysis from a working system in order to migrate it or extend its functionality [10]. This could be a difficult task when the underlying system conception has been perverted

* Corresponding author. Tel.: +34 957 21 26 60.
E-mail address: jrromero@uco.es (J.R. Romero).

http://dx.doi.org/10.1016/j.ins.2015.01.017
0020-0255/

due to requirements changes. A more dramatic situation occurs when reverse engineering techniques from source code are the only way to extract system information, leading to inappropriate abstractness because of missing documentation. In these cases, engineers must expend their time and effort, with their own experience as their only guarantee, in the manual discovery of these functional blocks.

Architectural optimisation methods in the field of software engineering (SE) have often proposed guidelines and recommendations to modellers for the identification and improvement of software architectures [5,6]. Hence, semi-automatic tools and intelligent systems might be an efficient solution to support the engineering work in order to obtain quality models.

More specifically, the discovery of the architecture of a software specification can also be formulated as the search of the most appropriate distribution of available software artefacts in more abstract units of construction. Traditionally, proposed approaches are based on the refactoring of source code [21,34], implying that architectural blocks are recovered at the end of the development process without regarding analysis decisions. Besides, it is frequent that source code is evolved without an exhaustive control from the analysis perspective, and it is likely not to be representative of the original conception of the system. Instead, the discovery process can be carried out using earlier available information, like the detailed analysis models in the form of class diagrams. These models offer an intermediate view of the software, between the abstractness of the architecture specification and the specificity of the code.

Recently, the combination of metaheuristic approaches and software engineering as problem domain, denominated search based software engineering (SBSE), has undergone a huge growth [17]. Since the appearance of SBSE, evolutionary computation (EC) has emerged as the most applied metaheuristic [16], demonstrating that it constitutes an interesting and complementary way to help software engineers in the improvement of their object-oriented class designs [33] or user interfaces [36]. In this paper, EC is explored as a search technique to extract the underlying software architecture of a system. It constitutes a novelty in SBSE, where architectural discovery has been viewed as a re-engineering task from source code, which is more oriented towards maintenance and refactoring purposes. The main research questions posed in this work are the following:

RQ1: Can single-objective evolutionary algorithms (EA) help the software engineer to identify an initial candidate architecture of a system at a high level of abstraction? Such an approach should be heavily oriented towards the expert domain, looking for the interoperability with software engineering standards and tools, as well as for the comprehension of the elements involved within the evolutionary model.

RQ2: How does the configuration of the algorithm influence both the evolutionary performance and the quality of the returned solution? In order to answer to this question, an in-depth parameter study is required, aiming to provide useful guidelines on this regard to the software architect.

In the proposed evolutionary approach, class diagrams constitute the source artefacts used to abstract the software architecture, which is encoded using a flexible tree structure. Design alternatives are explored by a specific genetic operator applying domain knowledge. Concepts like cohesion and coupling guide the search, defining a ranking-based fitness function.

The rest of the paper is structured as follows. Section 2 introduces some background in SBSE and architectural modelling. Section 3 details the problem description, whereas the evolutionary model is described in Section 4. Next, experimentation is presented in Section 5, including a detailed parameter study. An illustrative example of the approach is explained in Section 6, and results are discussed in Section 7. Finally, concluding remarks are outlined in Section 8.

2. Background

This section presents the most relevant subjects and background related to our work. More specifically, it introduces evolutionary computation as a technique to solve software engineering tasks, as well as the main terminology related to architectural analysis. Finally, previous works on software architecture optimisation in SBSE are presented.

2.1. Evolutionary computation in software engineering

Evolutionary computation [7] is one of the first population-based and bio-inspired metaheuristics for the resolution of optimisation problems. For this reason, EC has been applied for many years now to a variety of topics and considerable efforts have been applied in order to propose new techniques and operators [38] to solve complex applications.

Applying metaheuristics like EC to any domain requires that the scenario to be solved must be reformulated as an optimisation problem. Software engineering is not an exception [8]. Since the appearance of SBSE, considerable efforts have been devoted to this field. Although the first and most studied area has been the automation of test case generation [12], other tasks related to the rest of phases in the software development, from requirements specification [39] to software verification [27], have already been studied. The advances in the field demonstrate that the application of EC to software enhancement is not only focused in the generation of automated programs, other activities classically performed by humans present new challenges.

Since SE is mainly a human-centered activity, the automation of the expert's reasoning presents a great challenge, especially in the analysis and design phases [29]. Design tasks considered in SBSE encompass problems like the conception of both object-oriented [33] and service-oriented [30] architectures, software module clustering [28] or software refactoring