Computer Engineering and Applications Vol. 1, No. 1, June 2012

Techniques for Component-Based Software Architecture Optimization

Adil A abdeaiziz[1], Wan Nasir[2]

Department of Software Engineering
University Technology Malaysia
[1]*aasadil2@live.utm.my*
[2]*adil_sudan@hotmail.com*

ABSTRAKSI

Meskipun Component Base System (CBS) meningkatkan efisiensi pengembangan dan mengurangi kebutuhan untuk dipertahankan, dan komponen kualitas yang lebih baik dapat merusak produk yang baik jika komposisi tidak dikelola dengan tepat. Dalam dunia nyata, seperti domain otomasi industri, probabilitas ini tidak dapat diterima dan ukuran tambahan, waktu, upaya, dan biaya yang diperlukan untuk menguranginya. Banyak pendekatan optimasi umum telah diusulkan dalam literatur untuk mengelola komposisi sistem pada tahap awal pengembangan. Makalah ini mengkaji pendekatan baru untuk mengoptimalkan arsitektur perangkat lunak. Hasil dari penelitian ini akan bermanfaat untuk digunakan dalam mengembangkan optimasi kerangka kerja yang efisien untuk arsitektur perangkat lunak dalam penelitian yang sedang kami laksanakan.

Kata Kunci: Sistem Berbasis Komponen, pendekatan optimisasi, arsitektur perangkat lunak.

ABSTRACT

Although Component-Based System (CBS) increases the efficiency of development and reduces the need for maintenance, but even good quality components could fail to compose good product if the composition is not managed appropriately. In real world, such as industrial automation domain, this probability is unacceptable because additional measures, time, efforts, and costs are required to minimize its impacts. Many general optimization approaches have been proposed in literature to manage the composition of system at early stage of development. This paper investigates recent approach es used to optimize software architecture. The results of this study are important since it will be used to develop an efficient optimization framework to optimize software architecture in next step of our ongoing research.

Keywords: Component-Based System, optimization approach, software architecture.

1. INTRODUCTION

In recent years, there have been increasing interests in using Component-Based System Development (CBSD) approach, particularly COTS (commercial off the shelf) components, to develop large complex applications. Both software consumers and developers share the interest for the CBSD approach because of the clear advantages. Some advantages are but not limited to: The efficiency of development

increased, the product becomes more reliable, need for maintenance is radically decreased, the development time decreases, and the usability of the products increases. Although it promises faster time-to-market and increased productivity [1], many risks has been introduced when developing COTS-based systems such as failure to satisfy the quality attributes. The use of good quality components to develop system does not grantee to obtain system with the satisfied quality. Indeed, bad quality components will not produce high quality product, and even good quality components can damage a good product if the composition is not managed appropriately. In real world, such as industrial automation domain, this probability is unacceptable and additional measures, time, efforts, and costs are required to minimize it. For example it been reported that a large Japanese car manufacture had to recall 160,000 vehicles due to software failure [2]. Consequently, the failure to satisfy the quality attribute such as reliability means a financial loss, increased expenses of hardware, higher cost of software development, and harder than that, the loss of relationships with consumers. Whenever, quality issues are addressed at implementation or integration time, correction of problems impacts the cost, schedule, and quality of the software. For example it been reported that a large Japanese car manufacture had to recall 160,000 vehicles due to software failure [2]. Also, reports confirm that about 25 percent of software problems are related to software architecture and hardware-configuration issues that can be detected very early in the development cycle.

2. NEED FOR ARCHITECTURE OPTIMIZATION

When an architect starts building a new CBS application, he has many options to do this task. Each probable solution is arranging from a mixture of distinctive components. All those possible alternatives are called Design Options. The combination that satisfied the performance requirements is the target of the architect. However, design options are proportional with the degree of freedom. The degrees of freedom are resulted due to the following [3]: Components, the selection of one component from number of components instances with the same functionality but different performance specifications; Resource Allocation, due to the fact that, the selection of hardware does not impact the functional of components, its configuration could be changed during search. Therefore, hardware environment are modeled separately from the common assembly. In fact, manual or/and mismanaging composition lead to undetected problems in the system. Researchers have proposed Software Optimization Architecture Approaches to avoid such problem since it provides early evaluation for architecture.

3. OPTIMIZATION APPROACHES

From literature, solutions can be classified into three groups of optimization approach. Each approach aims to guide the search process towards the optimal solution, these main approaches are: Anti-Patterns based solution, Rule-Based Search, and Meta-heuristic search techniques. The approaches are discussed below.

ISSN: 2252-4274 (Print)
ISSN: 2252-5459 (Online)

4. ANTI-PATTERNS APPROACH

The approach aims to establish feedback generation process based on performance anti-patterns [4] using XML format. It takes as input the XML representation of the software system and gives in output a list of detected performance anti-patterns. No grantees to apply it in complex system. Since, it includes the problem (i.e. model properties that describe the anti-pattern) and the solution (i.e. actions to take for eliminate the problem). However, human experience in several steps is needed. For example, the detection of antipatterns in a subsystem is a task whose complexity heavily depends on the structure of the subsystem and the definition of the anti-patterns itself. Furthermore, there is no offer of new architecture candidates.

4.1. Rule-Based Approach

Rule-based [5] approaches try to identify problems in the model (e.g. bottlenecks) based on predefined rules and rules containing performance knowledge are applied to the detected problems. Rule-based approaches focus on performance analysis without considering other quality criteria. These approaches cannot find solutions for which no rule exists, thus, they cannot cover all possible solutions and might result in locally optimal.

2.4. Metaheuristic-Based Approaches

Meta-heuristics originated and inspired by natural process and creature's behavior to solve complex real world problems. Evolutionary Computing (EC) methods and the Swarm Intelligence (SI) algorithms are the main common groups of methods represent the field[6]. Meta-heuristics EC techniques such as Genetic Algorithms (GAs) methods have proven its usefulness to solve the problem of architecture optimization. Recently, SI techniques such as Particle Swarm Optimization (PSO) [7], [8], [9] an alternative search technique, often performed better than many EC techniques such as GAs when applied to various problems [10, 11]. EC need to handle the population movement; therefore, they are less fast in discovering optimal solutions. Furthermore, EC algorithms may have a memory to store previous status; this may help in minimizing the number of individuals close to positions in candidate solutions that have been visited before, but it may also slow to converge since successive generations may die out. In contrast, SI is easy to symbolize the architecture alternatives as an optimization problem, and less number of parameters required.

5. CONCLUSION

Architect needs to use optimization to avoid problem of quality dissatisfaction cause due to the late evaluation of developed system. Metaheuristic approaches provide efficient techniques to optimize software architecture. In contrast, other approaches such as of rule-based and Anti-Pattern do not cover the design space and no new candidates are suggested. Evolutionary and Intelligent swarm are subdivision of metaheuristics. Both approached used to optimize software architecture. However, the latter one has outperformed the former method.

Furthermore, SI algorithm is easier to manage. Based on this result we recommend using SI algorithm to develop new optimization approach.

REFERENCES

[1]. Voas, J., *COTS software: the economical choice?* Software, IEEE, 1998. **15**(2): p. 16- 19

[2]. D Hoch, et al., *The race to master automotive embedded systems development.* 2006, McKinsey Company, Automotive and assembly sector business technology office: Germany.

[3]. Martens, A. and H. Koziolek, *Automatic, Model-Based Software Performance Improvement for Component-based Software Designs.* Electronic Notes in Theoretical Computer Science, 2009. **253**(1): p. 77-93.

[4]. Cortellessa, V. and L. Frittella, *A framework for automated generation of architectural feedback from software performance analysis*, in *Proceedings of the 4th European performance engineering conference on Formal methods and stochastic models for performance evaluation*. 2007, Springer-Verlag: Berlin, Germany. p. 171-185.

[5]. Xu, J., *Rule-based automatic software performance diagnosis and improvement*, in *Proceedings of the 7th international workshop on Software and performance*. 2008, ACM: Princeton, NJ, USA. p. 1-12.

[6]. website. *Swarm and Evolutionary Computation*. 2011 [cited 2011 13-11-2011]; Available from: http://web.mysites.ntu.edu.sg/epnsugan/PublicSite/Shared%20Documents/Swarm%20and%20Evolutionary%20Computation_2.pdf.

[7]. Gudise, V.G. and G.K. Venayagamoorthy. *Comparison of particle swarm optimization and backpropagation as training algorithms for neural networks*. in *Proceedings of the Swarm Intelligence Symposium. SIS '03. IEEE*. 2003.

[8]. Boeringer, D.W. and D.H. Werner, *Particle swarm optimization versus genetic algorithms for phased array synthesis.* Antennas and Propagation, IEEE Transactions on, 2004. **52**(3): p. 771-779.

[9]. Eberhart, R. and Y. Shi, *Comparison between genetic algorithms and particle swarm optimization Evolutionary Programming VII*, V. Porto, et al., Editors. 1998, Springer Berlin / Heidelberg. p. 611-616.

[10]. Liang, J.J., et al., *Comprehensive learning particle swarm optimizer for global optimization of multimodal functions.* IEEE Transactions on Evolutionary Computation, 2006. **10**(3): p. 281-295.

[11]. Qasem, S.N. and S.M. Shamsuddin, *Radial basis function network based on time variant multi-objective particle swarm optimization for medical diseases diagnosis.* Applied Soft Computing, 2011. **11**(1): p. 1427-1438.

ISSN: 2252-4274 (Print)
ISSN: 2252-5459 (Online)