Assessing the Effect of Aspect Refactoring on Multi-Agent Applications: A Dynamic Analysis

Mohamed Sedik Chebout, Department of Mathematics and Computer Science, University of Ouargla, Ouargla, Algeria

Farid Mokhati, Department of Mathematics and Computer Science, University of Oum el Bouaghi, Oum el Bouaghi, Algeria

Mourad Badri, Department of Mathematics and Computer Science, University of Quebec, Trois-Rivières, Canada

ABSTRACT

Multi Agent Systems (MAS) are increasingly gaining importance as a powerful paradigm to designing and implementing distributed applications. However, existing multi-agent applications are developed without considering the separation of non-functional concerns from the functional ones. This makes the implementation, comprehension and maintenance of multi-agent applications hard tasks. Aspect-Oriented Refactoring (AOR) is a promising technique for improving modularity and reducing complexity of existing object oriented software systems by encapsulating crosscutting concerns. The authors present, in this paper, a new dynamic approach for investigating empirically the effect of AOR on MAS applications. They focus, particularly, on the effect of AOR on agent behavior in terms of communication. The proposed approach is supported by a multi-agent profiling tool working on AgentFactory platform.

Keywords: Aspect Refactoring, Assessment, Dynamic Analysis, Effect, Multi-Agent Systems

1. INTRODUCTION

Aspect-Oriented Programming (AOP) is the most popular paradigm that provides concern oriented decomposition (Kiczales et al., 1997). It provides, also, explicit constructs to develop software systems whose crosscutting concerns are better modularized because they are no longer tangled together and they are clearly separated from the principal decomposition.

Separating crosscutting concerns in existing code requires two main phases: aspect mining (identification of aspect candidates in code) and refactoring (semantic-preserving transformation to migrate the aspect candidate code to aspect-oriented code). Refactoring (Fowler, 1999)
refers to the restructuring of software. It can be defined as a change made to software in order to improve its structure (without changing its external behavior). The main aim of refactoring transformations is basically to improve software comprehension and to facilitate software maintenance by reversing the effect of software ageing. Studies have been conducted (on both qualitative and quantitative levels) in order to evaluate the benefits and drawbacks of using AOP to modularize crosscutting concerns (Hannemann & Kiczales, 2002; Garcia et al., 2005b; Greenwood et al., 2007).

Static analysis concerns techniques for obtaining information about the possible states that a program passes through during execution (Reps et al., 2004). In other words, Static program analysis is the art of reasoning about the behavior of computer programs without actually running them (Møller & Schwartzbach, 2015). However, dynamic analysis (e.g. software profiling) allows supporting a performance analysis process that consists in collecting, processing and analyzing run-time data. The most common use of profiling information is (Dufour et al., 2004):

• To support program optimization;
• To obtain more accurate results than a static analysis for concrete executions;
• To obtain temporal information nature;
• To obtain information on the frequency of certain events: JIT (Just in Time) optimization, performance debugging. Etc.;
• To support resources usage Analysis: memory usage, CPU, etc.

In the literature, the most recent works dealing with aspect-oriented assessment use static code metrics. These latter may be coupling metrics (Burrows et al., 2010), code understandability metrics (Thongmak & Muenchaisri, 2011), software modularity metrics (Przybylek, 2010; Przybylek, 2011). In contrast, few studies focused on dynamic analysis. We cite among others (Soria et al., 2007; Suganthi & Nadarajan, 2013). Consequently, the dynamic analysis field should be more explored, especially, with the advent of aspect-oriented programming.

In MAS context, dynamic analysis based performance assessment suffers from deficiencies in terms of agent specific performance measurement. The common factor between traditional profilers (Erik & Jim, 2004; Dufour et al., 2004; Haupt & Mezini, 2004) is, generally, performance program analysis such as: CPU usage rate, memory footprint, program input/output, etc. The specific performances inherent to MAS like communications issues (communication cost, communication overloading and impact of received message on agent behavior) are not taken into account. In this paper, we present how these concepts can be mapped to MAS.

Note that the choice of dynamic analysis as an evaluation technique is not fortuitous. Also, we found that Alegandro Garcia’s works and his team (Garcia et al., 2002; Garcia et al., 2003; Garcia, 2004a; Garcia at al., 2004b; Garcia et al., 2005a; Garcia et al., 2005b), are focusing primarily on the effect of aspect oriented programming on modeling (Garcia & Lucena, 2001; Garcia et al., 2006; Silva et al., 2006), design (Kulesza et al, 2005) and implementation (Sant’Anna et al., 2004) (i.e. Static code analysis) levels. The authors used in their studies two MAS-specific case studies: Portalware2 (Garcia et al., 2001) and ExpertComitte (Garcia et al., 2004b) 3. Therefore, the combination of the two techniques, AOR in Agent Oriented Software Development (AOSD) and dynamic analysis, offers an open research area that deserves to be explored with respect to the significant lack of research in this context.

AgentSpotter (Doan Van Bien et al., 2009a; Doan Van Bien et al., 2009b) is a profiling tool designed specifically for gathering and displaying profiling information on MAS. It is implemented under Agent Factory framework (Collier, 2008). Also, AgentSpotter uses two tools:
To Implement an Open-MAS Architecture for Semantic Web Services Discovery: What Kind of P2P Protocol Do We Need?
www.igi-global.com/article/to-implement-an-open-mas-architecture-for-semantic-web-services-discovery/122472?camid=4v1a