Available online at www.sciencedirect.com

SciVerse ScienceDirect

Procedia Computer Science 18 (2013) 250 – 259

Procedia Computer Science

International Conference on Computational Science, ICCS 2013

An Empirical Evaluation of the Cost and Effectiveness of Structural Testing Criteria for Concurrent Programs

Maria A. S. Brito[a], Simone R. S. Souza[a], Paulo S. L. Souza[a,*]

[a]*Universidade de São Paulo, Instituto de Ciências Matemáticas e de Computação, ICMC/USP, P.O. 668, São Carlos (SP), Brazil, 13560-970*

Abstract

Concurrent program testing is not a trivial task. Features like nondeterminism, communication and synchronization impose new challenges that must be considered during the testing activity. Some initiatives have proposed testing approaches for concurrent programs, in which different paradigms and programming languages are considered. However, in general, these contributions do not present a well-formed experimental study to validate their ideas. The problem is that the data used and generated during the validation is not always available, hampering the replication of studies in the context of other testing approaches. This paper presents an experimental study, taking into account the concepts of the Experimental Software Engineering to evaluate the cost, effectiveness and strength of the structural testing criteria for message-passing programs. The evaluation was conducted considering a benchmark composed of eight MPI programs. A set of eight structural testing criteria defined for message-passing programs was evaluated with the ValiMPI testing tool, which provides the support required to apply the investigated testing criteria. The results indicate the complementary aspect of the criteria and the information about cost and effectiveness has contributed to the establishment of an incremental testing strategy to apply the criteria. All material generated during the experimental study is available for further comparisons.

Keywords: Software testing; Concurrent programs; MPI programs; Experimental study

1. Introduction

The demand for distributed and parallel applications has been growing due to the advanced hardware technology, which provides more efficient machines and allows to process large volumes of data. However, these applications are inevitably more complex than sequential ones. Every concurrent software contains features, such as nondeterminism, synchronization and inter-process communication, which significantly hamper their validation and their testing. Approaches to test concurrent programs efficiently and effectively have been proposed and are an important factor for the success and quality of the programs built in this domain.

Several studies have been conducted to define approaches to test concurrent programs [1, 2, 3, 4, 5]. In general, these studies present some evaluation to demonstrate the applicability of their contribution. The problem is that the data used is not always available, hampering the replication of the studies and the fair comparison among different testing techniques for concurrent programs. In the context of sequential program testing, Weyuker [6] pointed out the importance of comparative studies to evaluate different testing techniques, allowing the replication

*Corresponding author. Tel.: +55-16-3373-9700 ; fax: +55-16-3373-9700
E-mail address: masbrit@icmc.usp.br; srocio@icmc.usp.br; pssouza@icmc.usp.br.

Selection and peer review under responsibility of the organizers of the 2013 International Conference on Computational Science
doi:10.1016/j.procs.2013.05.188

of the experimental studies. Thus, demonstrating the applicability and effectiveness of testing techniques is as important as proposing testing techniques for new application domain.

The empirical evaluation of techniques, criteria and testing tools has been intensified in recent years, mainly in the context of traditional programs. These empirical evaluations, in general, consider three basic factors for a comparison: cost, effectiveness and strength [7]. Cost refers to the effort required to satisfy a testing criterion and can be measured by the number of test cases necessary to cover it. Effectiveness refers to the ability of a test set to reveal defects. Strength refers to the probability to satisfy a testing criterion using a test set adequate to another testing criterion. These comparison factors are important for the proposition of an efficient testing strategy taking into account the benefits of each testing criterion.

This paper has contributed in this direction, presenting an experimental study that evaluates structural testing criteria defined for message-passing programs, analyzing their cost, effectiveness and strength. Experimental study takes into account the process defined by Wholin [8], which includes activities for the definition, planning, conduction, analysis and packing of experimental studies. A benchmark composed of eight MPI (Message Passing Interface) programs is defined and used in our study. MPI is a message-passing library interface specification for the development of portable message-passing concurrent programs using sequential languages, such as C and Fortran [9]. Our work considers MPI programs written in C language.

The analyzed testing criteria were proposed by Souza et al. [10] and include structural criteria to test concurrent and sequential aspects of message-passing programs. Information about control, data and communication flows is extracted from a program under test and used to guide the generation of a test case set. Eight different testing criteria were analyzed using the ValiMPI support tool [11]. This tool provides the required resources to apply test cases and evaluate their coverage in programs considering the structural testing criteria for MPI programs.

The material generated during the experimental study, including programs, results of testing activity, ValiMPI tool and results of experimental study has been organized and is available for public access, providing relevant information to further studies and comparisons. According to our knowledge, it is the first study which uses concepts of the Experimental Software Engineering for the definition, conduction and analysis of the testing criteria in the context of concurrent programs.

The remaining of the paper is organized as follows: Section 2 presents the test model and the structural testing criteria for message-passing programs, as well as the ValiMPI testing tool. Section 3 describes the experimental study, including the definition, planning, results and analysis. Section 4 reports the related works and Section 5 presents the final considerations and future research directions.

2. Structural Testing Criteria for Message-Passing Programs

In message-passing programs, communication is made by *send* and *receive* basic primitives, in which a process can send a message to another process or to a group of processes. The first one is called point-to-point communication and the second is called collective communication. In both cases, the test activity must establish an association between the variables sent and the location where these variables are used in the receiver process(es). Besides, it is important to define a strategy to derive all possible synchronizations among the processes of the concurrent software processes. Executing these distinct synchronizations allows the verification of different pairs of definition and use of variables in different processes. Associated to these synchronizations there are important questions that must be considered, such as non-determinism, controlled execution and race conditions.

Souza et al. [10] defined a test model and a set of structural testing criteria that represent the main features of message-passing programs, including control, data and communication aspects. The test model considers that a concurrent program is a set $Prog = p^0, p^1, ..., p^{n-1}$ composed of its n parallel processes. A Control Flow Graph (CFG) of each process p is generated using the same concepts of sequential programs. Each CFG^p represents the control flow of a process p. A Parallel Control Flow Graph (PCFG) generated for $Prog$ is composed of CFG^p (to $p = 0 \ldots n-1$) and edges of communication among processes. N represents the set of nodes and E represents the set of edges in PCFG. E has two subsets: E_p, which are edges of a same process and E_s, composed of edges that represent the communication between processes, called interprocess edges. A node i in a process p is represented by notation n_i^p. Two subsets of N are defined: N_s, which are the nodes composed of primitives *send* and N_r, which are nodes composed of primitives *receive*. A set R_i^p is associated with each $n_i^p \in N_s$ containing possible nodes that can receive the message sent by node n_i^p. This set is important to establish all possible communication pairs.