Effective Estimation Software cost using Test Generations

Sattarova Feruza¹, Farkhod Alisherov²

Abstract

In software engineering, we have many problems particularly with software cost estimation and the assessment of the evaluation model. Software cost estimation mainly depends on two factors which are the estimation consistency and estimation accuracy. These are very much important to the perfect development of the software without any drawbacks to it. In the past there have been many models and techniques that have failed to give the correct software estimation that really challenges the all factors of perfect software in this paper we use software requirement specification but we were not getting the accurate results that are helpful for developing of perfect software that is having perfect cost estimation. So we introduce the Automated Test-Data Generation Techniques method that gives us the desired results which will be helping us to know the software cost estimation very much accurately and also estimate consistently.

Keywords: Cost Estimation, SRS document, automated test data generation, optimization techniques.

1. Introduction

Software Engineering is the study of design, development and maintenance of software. In other words it is the application of a systematic, disciplined, quantifiable approach to the development, operation, and maintenance of software and an engineering discipline that is concerned with all aspects of software production. Communication skills, team dynamics, working with a "customer," and creativity are also important factors in the software engineering. It is important because of the large expensive software systems [1-6].

Software development process:

A set of activities that leads to the production of a software product is known as a software process. Computer-aided software engineering (CASE) tools are being used to support the software process activities. However, due to the vast diversity of software processes for different types of products, the effectiveness of CASE tools is limited. There is no ideal approach to software process that has yet been developed. Some fundamental activities, like software specification, design, validation and maintenance are common to all the process activities.

Received(March 29, 2011), Review request(March 30, 2011), Review Result(1st: April 20, 2011)
Accepted(June 10, 2011)

¹306-791 Dept. of Multimedia, Hannam University, Daejeon, Korea.
email: myma6585@gmail.com

²(Corresponding Author) 306-791 Dept. of Multimedia, Hannam University, Daejeon, Korea.
email: sntdv@yahoo.com

ISSN: 2383-5281 AJMAHS
Copyright © 2011 SERSC
A software development is also known as software development life cycle (SDLC). It is a term used to describe a process of analysis, planning, design, maintenance, deployment and implementation of an application.

**Risk Management in Software Engineering:**

The management of a risk is the important future in throughout the software development life cycle. A risk is a potential future harm that may arise from some present action, such as, a schedule slip or a cost overrun.

“Risk in itself is not bad; risk is essential to progress, and failure is often a key part of learning. But we must learn to balance the possible negative consequences of risk against the potential benefits of its associated opportunity.”

Risk management is a series of steps whose objectives are to identify, address, and eliminate software risk items before they become either threats to successful software operation or a major source of expensive rework.

**The Risk Management Process:**

The risk management process can be divided into two phases. Those are risk assessment and risk control. The risk assessment further broken down into risk identification, risk analysis, and risk prioritization. Like that risk control also divided into risk planning, risk mitigation, and risk monitoring.

![The Risk Management Cycle](image)

**Software Risk Management:**

There could be risk associated with the every software project, the main goal is to identify and manage those risks. The most important risk management tasks are risk index, risk analysis, and risk assessment.
1) **Risk Index**: Risk index is the multiplication of impact and probability of occurrence. Risk index can be characterized as high, medium, or low depending upon the product of impact and occurrence. Risk index is very important and necessary for prioritization of risk.

2) **Risk Analysis**: The risk analysis is used to identify the high risk elements of a project. The main purpose of risk analysis is to understand risks in better ways and to verify and correct attributes. A successful risk analysis includes important elements like problem definition, problem formulation, data collection.

3) **Risk Assessment**: It integrates risk management and risk analysis. Risk assessment requires correct explanations of the target system and all security features. It is important that risk deferent levels like performance, cost, support and schedule must be defined properly for risk assessment to be useful.

2. Related Work

Jeff Tain and Marvin V. Zelkowitz stated that “A formal model of program complexity developed earlier by the authors used to derive evaluation criteria for program complexity measures. This is then used to determine which measures are appropriate within a particular application domain. A set of rules determining feasible measures for a particular application domain are given, and an evaluation model for choosing among alternative feasible measures is presented. This model is select measure from the classification tree produced by empirically guided software development environment of Selby and portar, and early experiments show it to be an effective process” [7].

Sheng Yu, Shijie Zhou stated that “With the evolution of the software development, the scale of the software is increasingly growing to the extent that we cannot hand it easily. Some metrics are proposed to measure the complexity of software in last a few years. This article aims at a comprehensive survey of the metric of software complexity. Some classic and efficient software complexity metrics, such as Lines of Codes (LOC), Halstead Complexity Metric (HCM) and Cyclomatic Complexity Metric (CCM), are discussed and analyzed first. Then, some other approaches driven from above classic metrics are also discussed. The comparison and the relationship of these metrics of software complexity are also presented” [8].

Qu Yi, Zhou Bo, Zhu Xiaochun stated that “Software quality becomes an increasingly important factor in software marketing. It is well known that software testing is an important activity to ensure software quality. Despite the important role that software testing plays, little is known about the prediction of test suites size. Estimation of testing size is a crucial activity among the tasks of testing management. Work plan and subsequent estimations of the effort required are made based on the estimation of test suites size. The earlier test suites size estimation we do, the more benefit we will get in the process of testing” [9].

HyeYeon Kim stated that “Software development starts from specifying the requirements. A Software
Requirements Specification (SRS) describes what the software must do. Naturally, the SRS takes the core role as the descriptive documentation at every phase of the development cycle. To avoid problems in the latter development phases and reduce life-cycle costs, it is crucial to ensure that the specification be reliable. In this paper, we describe how to model and test (i.e., check, examine, verify and prove) the SRS using two formalism (Z and State charts).

Moreover, these formalism were used to determine strategies for avoiding design defects and system failures. We introduce a case study performed to validate the integrity of a Guidance Control Software SRS in terms of completeness, consistency, and fault-tolerance” [10].

Bogdan Korel stated that “Test &ta generation in program testing is the process of identifying a set of test data that satisfies a selected testing criterion, such as, statement coverage or branch coverage. The existing methods of test data generation are lifted to unit testing and may not efficiently generate test data for programs with procedures. In think paper we present an approach for automated test &ta generation for programs with procedures. This approach builds on the current theory of execution-oriented test data generation. In the approach, test data are derived based on the actual execution of the program under test. For many programs, the execution of the selected statement may require prior execution of some other statements that may be part of some procedures. The existing methods use only control flow information of a program during the search process and may not efficiently generate test data for these types of programs because they are not able to identify statements that affect execution of the selected statement. Our approach uses data dependence analysis to guide the process of test data generation. Data dependence analysis automatically identities statements (or procedures) that affect the execution of the selected statement and take information is used to guide the search process. The initial experiments have shown that this approach may improve the chances of finding test data” [11].

LORI A. CLARKE stated that “This paper describes a system that attempts to generate test data for programs written in ANSI Fortran. Given a path, the system symbolically executes the path and creates a set of constraints on the program's input variables. If the set of constraints is linear, linear programming techniques are employed to obtain a solution. A solution to the set of constraints is test data that will drive execution down the given path. If it can be determined that the set of constraints is inconsistent, then the given path is shown to be non-executable. To increase the chance of detecting some of the more common programming errors, artificial constraints are temporarily created that simulate error conditions and then an attempt is made to solve each augmented set of constraints. A symbolic representation of the program's output variables in terms of the program's input variables is also created. The symbolic representation is in a human readable form that facilitates error detection as well as being a possible aid in assertion generation and automatic program documentation [12]."
3. Existing System

A software requirement specification in its most basic form is a formal document used in communicating the software requirements between the customer and the developer. With this in mind then the minimum amount of information that the software requirement specification should contain is a list of requirements which has been agreed by both parties. The types of requirements are defined in section 3.4. The requirements, to fully satisfy the user should have the characteristics as defined in section 3.5. However the requirements will only give a narrow view of the system, so more information is required to place the system into a context which defines the purpose of the system, an overview of the systems functions and the type of user that the system will have. This additional information will aid the developer in creating a software system which will be aimed at the user’s ability and the clients function [13].

A software requirements specification has a number of purposes and contexts in which it is used. This can range from a company publishing a software requirement specification to companies for competitive tendering, or a company writing their own software requirement specification in response to a user requirement document. In the first case, the author of the document has to write the document in such a way that it is general enough as to allow a number of different suppliers to propose solutions, but at the same time containing any constraints which must be applied. In the second instance, the software requirement specification is used to capture the user’s requirements and if any, highlight any inconsistencies and conflicting requirements and define system and acceptance testing activities [13].

![Diagram of Software Requirement Specification](Fig. 2)
Estimation of software complexity using SRS:

This aims at proposing a measure for early estimation of software complexity on the basis of the SRS of the proposed software. SRS is an unambiguous and complete software specification document for software. Elaborates the recommended practices for documenting the software requirements to generate a Software Requirement Specification (SRS) [14]. The software development starts only after documenting the software requirements using SRS. The contemplation is that, if the entire software is implemented based on this document, can we not estimate the various software development activities too from this SRS? Since complexity analysis has an extremely high payoff for the investment, hence it is necessary to carry out a precise and accurate estimation of software complexity for software to be developed [1].

The process starts by extracting the attributes from SRS of the proposed software and later the extracted attributes are arranged in order to obtain various contributing complexity measures for the estimation of requirement based software complexity. For more precise and accurate estimation, external references are also considered [15][5]. Further, these contributing complexity measures are consolidated to build an IRBC measure [1].

4. Proposed System

Automated Test-Data Generation Techniques:

Several approaches to generating test-data exist. Very few of these approaches are automated, exceptions to this include: goal-oriented, path-oriented, analysis-oriented and random. Due to the complexity of software programs developed in industrial settings, these test-data generation techniques have only been demonstrated to be effective for simple programs. Generally, software programs developed in industry usually exhibit common characteristics: (a) they are very large in size, (b) highly complex, and (c) contain a wide variety of structural features such as arrays and pointers. The success of automated test-data generators with industry software has been very limited due to such characteristics, inhibiting the widespread use of automated test-data generators [2].

Optimization techniques:

In this technique, combination of Generic Algorithm (GA) and Stimulated Annealing (SA) is considered. Genetic Algorithms are based on an abstract model of natural genetic evolutionary process. Simulated Annealing originates from the analogy between the annealing process of solids and the problem of solving combinatorial optimization problems [2]. In GAs, the possible solutions of the given optimization problem are represented as the bit strings called chromosomes (Schaffer 1987). The chromosomes can be represented in a number of
different ways. Traditionally, chromosomes are represented as binary strings. Simulated Annealing is a search technique where a single trial solution is modified at random. In condensed matter physics, annealing is a process of cooling a solid to reach a minimal energy state (ground state). At initial high temperatures, all molecules of the solid randomly arrange themselves in a liquid state, as the temperature descends gradually; the crystal structure becomes more ordered and reaches a frozen state when the temperature drops to zero. Energy represents the quality of the proposed solution. In order to find the best solution, the energy needs to be at its minimum. The smaller the energy gets, the better a solution becomes. Therefore, changes that lead to a “lower energy” are automatically accepted. Meanwhile, changes that cause a “higher energy” are accepted based on a probability given by the Boltzmann factor-acceptance rate. This probability is defined as \( \exp(-\Delta E / kT) \). Where \( \Delta E \) is the change in energy, \( k \) is a constant and \( T \) is the Temperature. When applying simulated annealing, the temperature \( T \) is initially set to a high value. This temperature is repeatedly lowered slight according to a cooling schedule. The probability of accepting a lesser quality solution that will lead to a “higher energy” allows the SA algorithm to frequently escape the local minima [2].

**Objective function:**

The main objective of the GA & SA algorithm is to find a schedule of operations that can minimize the maximum completion time that is the completed time of carrying total operations out in the schedule for \( n \) jobs and \( m \) machines. The Objective or fitness function takes the input as the number of jobs, number of Operations, Chromosome, Operation time Sequence and Machine Sequence of the corresponding operation. Each chromosome genes are assigned by an integer number \( nk \) by ranking the genes (real numbers) in ascending order. And then perform \( (nk \mod No. \text{ of Jobs}) +1 \) operation to each \( nk \) to get the corresponding operation sequence of a chromosome. The fitness function produces the output as a makes pan value for the corresponding operation sequence [13].

**Algorithm:**

1. Initialize temperature \( T \) to a particular value.
2. Initialize the \( N \) number of chromosomes by generating \( n \times m \) real numbers for each chromosome.
3. Find the Operation time sequence and Machine sequence for \( N \) chromosomes.
4. Find the makes span value for each and every chromosome using the objective function and also find the minimum makes pan value (best) among \( N \) makes span values.
5. Select \( N/2 \) chromosomes using the Roulette - Wheel selection from \( N \) chromosomes
6. Crossover the selected chromosomes with the probability as 0.9 and Mutate the new chromosomes with the probability as 0.3 to get new chromosomes.
7. Find the makespan values for newly generated chromosomes using the objective function.
8. Choose the N best chromosomes which have the minimum makespan values from the newly generated and also from old chromosomes.
9. Find the minimum makespan value (best) among the N best chromosomes.
10. If best chromosome is not changed over a period of time then find a new chromosome using temperature.
11. Accept the new chromosome as best with probability as exp(∆E/T), even though current position is worse. Here ∆E is the difference between current best chromosomes makespan and new chromosome’s makespan value.
12. Reduce T.
13. Terminate if the maximum number of iterations is reached or optimal value is obtained.
14. Go to step 3.

5. Experimental Results

The experimental results show that the result achieved authenticates the claim that the proposed measures are comprehensive one and compares well with various conventional measures. So this is a time taking process, hence we propose the automated test generation approaches based on optimization techniques, Which are used to decrease the total communication and computation in the form of input and output cost and time of software testing. A wide range of optimization techniques can be used within these test-data generators, and their relevant characteristics, when applied to these situations stay relatively unknown. The main objective of the GA&SA algorithm is to find a schedule of operations that can minimize the maximum completion time that is the completed time of carrying total operations out in the schedule for n jobs and m machines.

6. Conclusion

The results show that the combination of GA and SA has the best overall performance. In fact, the GA technique consistently out-performs the other approaches. GA achieves complete condition-decision coverage with the Time Shuttle, Perfect Number, and Rescue programs. GA was not capable of achieving complete coverage with the other SA; however, no other optimization technique was able to perform better. GA has the ability to retain the good gene inherited from ancestors and contribute it to successive generations. This helps the GA generate quality test cases quickly. The SA was not able to reach the coverage levels achieved by the GA in many occasions. So the combination of GA and SA did generally well with both input spaces, reaching
average coverage levels of 85% and above. This indicates their potential to be suitable to perform with industrial software. In conclusion, based upon the results from this study, we would recommend that researchers using an optimization technique as the basis of a goal-oriented test data generation system should use a GA and SA based approach.
References


