Mission Reliability Modeling Methodology for Software Dynamic Evolution

Yumei Wu[1]; and **Yongli Yu[2]**

[1]School of Reliability and Systems Engineering, Beihang University, Beijing, China
[2]Maintenance Engineering Institute, Shijiazhuang, China

Sp. Issue 2013/ # Paper 1
Corresponding author:
Yumei Wu
Mailling address: No. 37, Xueyuan Road, Haidian District, Beijing, 100191, P. R. China
E-mail: wuyumei@buaa.edu.cn
Tel: +86-13903116343
Fax: +86-10-82317663

KEYWORDS

Keywords-Mission reliability; Software evolution; Software quality; Dynamic; Reliability modeling

ABSTRACT

From the concept of software dynamic evolution, the paper makes a detail analysis of the influence of the software evolution on software mission reliability. The basic question of the software mission reliability for the software evolution is presented. The model framework of software mission reliability is built, and the modeling methods of software mission reliability models are given. The research conclusions have importantly theoretical and engineering value for the reliability analysis and evaluation of the reconfigurable software on function.

Introduction

With the informationization of weapons, software plays an irreplaceable role in the weapon operation, as an important part of weapons. Currently, with the changes in mission requirements and constant upgrade in hardware, many large and complex software systems (operational command and control software, a new generation aircraft avionics systems software), begin to add new features, continually modify. For instance, with the different combat objects, the different combat environment, and on weapon equipments upgrades, the software for the operational command and control system needs to modify the existing features; add new functions, in order to meet the operation. The important characteristics that the software system modify constantly during the use, is namely software evolution, which greatly affects the reliability, maintainability and various quality characteristics of the software.

Scholars have made deep and meticulous researches on the fault-tolerant functions of software system based on dynamic configuration (Guo, *et al.*, 2011; Khalgui, *et al.*, 2011, Alkhalid, *et al.*, 2012; Mortensen, *et al.*, 2012; Bavota, 2012), methods to improve software quality based on the dynamic configuration (Hanakawa, 2011; Meanaeatra, *et al.*, 2011; Mens and Tourwe, 2004), dynamic configuration optimization (Murphy-Hill, *et al.*, 2009; Gracioli and Fröhlich, 2010; Burger and Hummel, 2011; Soares, 2011), the dynamic configuration of avionics systems software for the IMA (Koru and El Emam 2009; Watkins, *et al.*, 2006; Strunk and Knight, 2006). However, the current software architecture modeling and software reliability modeling techniques are difficult to adapt to this new problem of software evolution. Therefore, the reliability models and modeling methods for software evolution have extremely urgent requirements in theory and engineering. This paper analyzes the concept of software evolution firstly , then taking the Systems Science as direction, tries to build the software task reliability modeling framework and methodology for the software dynamic evolution, and aims to lay the foundation of the software mission reliability research under the software dynamic evolution.

Basic Problem of Software Evolution and Mission Reliability

(1) Software Evolution

Until now, there is not accurate definition of software evolution (Mens and Tourwe, 2004). In general, the Software Evolution refers to the behaviors of software maintenance and the processes of software updates within the life cycle of the software system. In modern software system's life cycle, the evolution is an active that runs through the whole process. Changing the system needs, achieving enhanced functionality, adding new features, changes in the software architecture, software bug fixes, changes in the operating environment and so on, all require that the software system have a strong ability of the evolution, and quickly the adapt the changes, and reduce the cost of software maintenance.

Software evolution can be divided into static evolution and dynamic evolution according to whether the software is in the operation. Based on the predictability of evolution results, dynamic evolution can be further divided into planed evolution and non-planed evolution. In addition, according to the evolution hierarchy, the dynamic evolution can be divided into the dynamic evolution of the function level, the class/object level, the component level, the structural level, and the workflow level.

(2) The Basic Problem of Mission Reliability for Software Evolution

Software evolution has an important impact on software reliability, especially on the software evolution process and correlative technology for software reliability. From looking at the impact of software evolution process, the basic problems of mission reliability for software evolution are mainly: Software architecture is modeled based on mission decomposition. According to the software development needs, the design requirements and other related documents, the decomposition method of software mission may be researched by dint of the hierarchical thinking. At the same time, by use of the mission-driven, the software function modules are divided, and the correspondence relations between task and function modules are established.

The task path is research based on evolution rules. The main works firstly focus on how to access the method for the evolution rules of the software task execution path. The formal representation method of evolution rules will be studied to provide a stronger theoretical foundation for the task path evolution. According to the problems encountered in the software execution process, the task execution path is usually, adjusted behind by dint of the evolution rules. Therefore, the rule-based certainty and uncertainty evolution will affect the decision of the task path. Then, the enforcement mechanisms of the tasks under different evolutionary rules will be studied, and the affect of task execution path will be researched when the software is actually running.

The mission reliability simulation modeling method based on multi-level simulation. Taking the software mission reliability as a metrics, the integration among the software architecture modeling based on the task decomposition, the reconstruction of functional modules based on the rules and the dynamic configuration mechanism will be done. The simulation processes of software mission reliability will be defined, the simulation algorithm of software mission reliability will be optimized to enhance the simulation efficiency and accuracy.

(3) The Mission Reliability Modeling Framework for Software Evolution

According to the basic problems of mission reliability, the mission reliability model and modeling method framework for software evolution are built, See Figure 1.

Mission Reliability Modeling Method for Software Evolution

(1) Software Architecture Modeling Based on task Decomposition

Software architecture modeling based on task decomposition is shown in Figure 2.

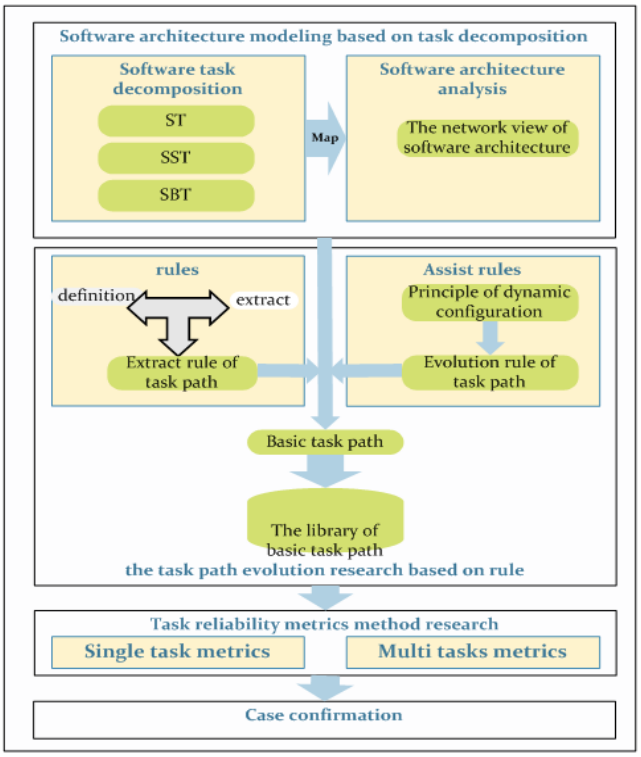

Figure 1: Mission Reliability Model Framework For Software Evolution

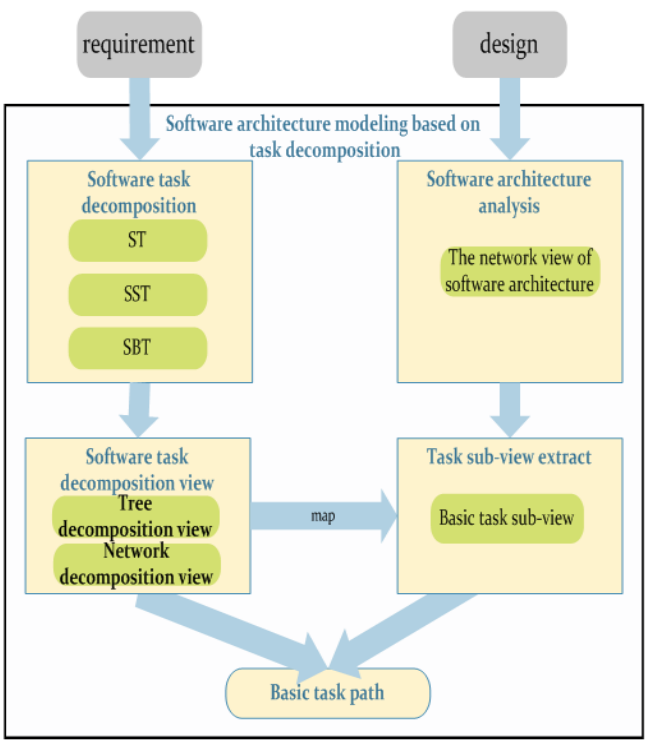

Figure 2: Software Architecture Modeling Technology Based on Task Decomposition

First, by means of the hierarchical method, the software task (ST) is divided into the software sub-tasks (SST) and software basic tasks (SBT), and the tree hierarchy structure of "software task - software sub-task - software basic task" is established, shown in Figure 3.

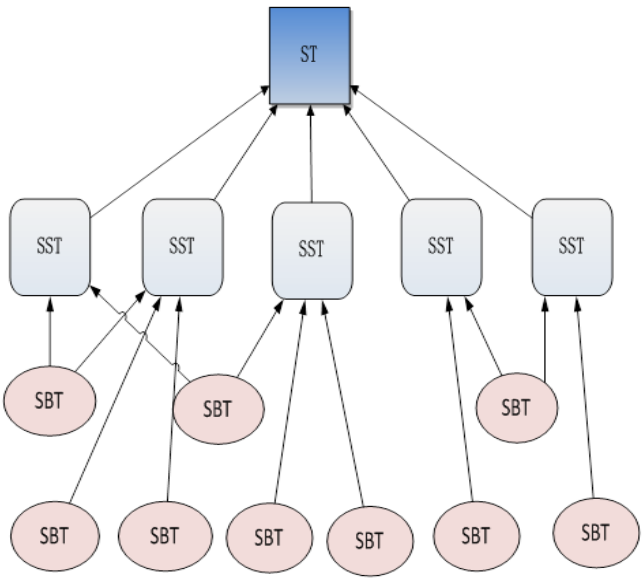

Figure 3: The Tree View of Software Task Decomposition

As different SST can contain the same SBT, the SBT can be divided into two categories: one is for the SBT shared by the plurality of SST (see Figure 3). Another is for the SBT of a unique SST (see Figure 3). The sharing SBT are often likely to come into conflict when multiple SST are in the concurrent execution, and cause unreasonable resources seizing and task scheduling problem. It is a key consideration in the modeling process. Furthermore, when the number of the SST and SBT comes to a certain extent, and the sharing relationship between the SST and SBT is more complex, the tree hierarchy of the software will evolve into a network structure (see Figure 4). The tree hierarchy in Figure 3 is isomorphic relationship with the network topology in Figure 4. The formal methods can be used to describe a view of the software task, and reduce the sharing relationship of the SBT, thus to achieve a dynamic allocation of resources and a reasonable scheduling of tasks.

After the completion of the task decomposition, the SBT is defined as an undivided and intact software task in logic, consisting of a set of software functional modules combining with a certain rule.

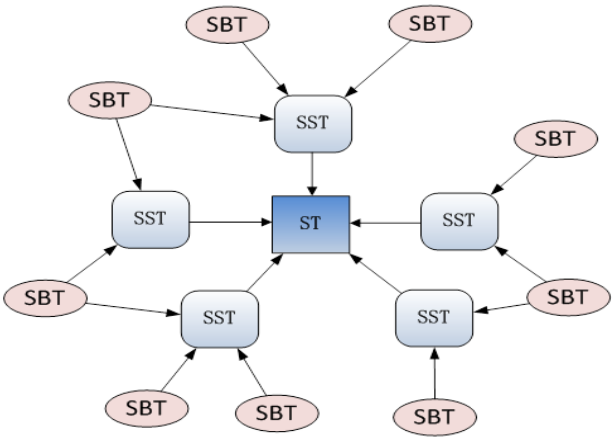

Figure 4: The Network View of Software Task Decomposition

The software function modules are distinguished from the view of task completed. At the same time, the structure view description is provided that the function modules complete the software task. The function module is abstracted as a node, the relationships between functional modules as an edge. The software architecture can be abstracted as a directed graph. For the operational command and control system software, the architecture of complex software, has the network characteristics, shown in Figure 5 (a).

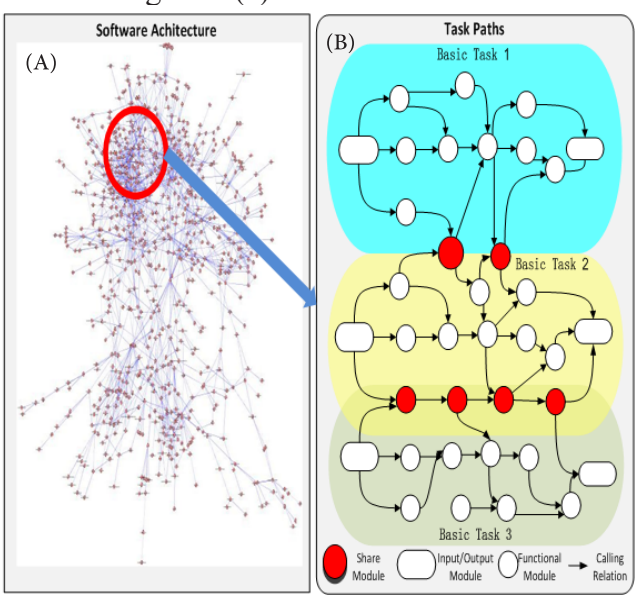

Figure 5 (a & b): The Software Architecture Diagram & The Task Path Found Based on Software Architecture

In software architecture, when extracting a SBT of the software, the whole involved functional modules and relationships can be called, the execution path of the SBT can be built, and directed sub-graph can be formed, shown in Figure 5(b). Since the different SBT would invoke the same functional module (shared module), it the conflict will occur among the execution paths of SBT. When multiple SSTs are concurrent execution, the issues are considered such as resource sharing and task priority. By means of the formal methods the software architecture and its internal task path can be modeled. The same time, formal methods can precisely define the sharing relationship of functional modules among the task paths, to achieve the automated scheduling of resources, and also to facilitate the later analysis.

The static software architecture, task decomposition and dynamic task path finding can be integrated, to form the hierarchical and network software architecture model for task and the static and dynamic software internal structure.

(2) The Evolution Rules of Task Execution Path Modeling of the evolution rules of task execution path shown in Figure 6.

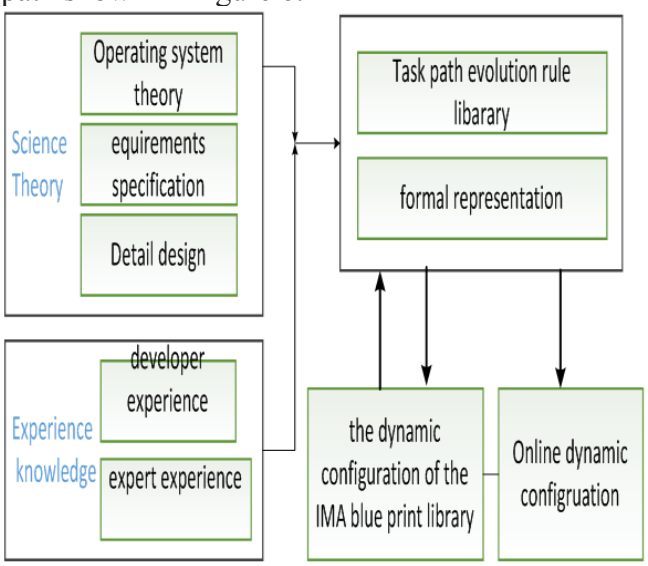

Figure 6: The Task Path Evolution Research Based on Rule

Usually due to task changing or functional module failure, the online dynamic evolution of the new generation software can occur during operation. Therefore, after determining the execution path of SBT by means of the software architecture modeling process, the task execution

path problem must be solved when the software evolution take place.

The evolution of software task path may be caused by a variety of reasons, and the evolution mechanisms are not the same, the reasons described to lead to the evolution, the mechanisms of evolution and the evolution rules are needed to extract and define. For the given evolution rules, the reconstruction of the function module as well as the path changes are found out during the task execution process.

The evolution rules can be obtained as follows:

(2.1) Scientific theories, including requirements specification, design document, and operating system theory. The possible changes of task execution path may have been taken into account in the requirements and design stages of the software development, and alternative paths under different conditions have already been pre-defined. In addition, some of the rules can be extracted from the operating system theory. According to analyzing the relative task scheduling algorithm of real-time embedded operating system, common task abnormalities can be summarized, such as deadlock, shared resource inconsistency, and the corresponding evolution program of the task path for different abnormal task is given.

(2.2) Experience knowledge, including the experiences of the developer and expert. Through the developers' designing, coding, testing experience, and the experience of the experts, the most common problems can be determined in the implementation of the software, and the corresponding evolution program of task execution path can also be given. In addition, we can also consider learning from the dynamic configuration library of the IMA Blue Print.

After extracting the evolution rules of software task execution path, these evolution rules are defined and described by formalized methods, the alternatives of execution path are given when the resource sharing conflicts, task scheduling chaos and the function module fails. Using these formalized descriptions of the rules, on the one hand, these rules are able to be presented on dynamic configuration rule library similar to the Blue Print, and the automated task path evolutes under the rules guidance when the software is running online, to realize the software dynamic reconfiguration online. On the other hand, artificial intervention can be taken, temporarily shielding the fault functional modules and its infection modules in the execution path, and replacing the fault modules online. In actual operations, once some software function module has the problem or multiple concurrent tasks appear, the alternatives must be selected in the rule library, and a new execution path is determined by the pre-developed dynamic configuration rules of the software system, to achieve the online dynamic evolution of software system.

(2.3) The Mission Reliability Modeling Methods

The mission reliability of software system is modeled by the multi-level simulation method. The appropriate simulation tools will be selected to achieve the modeling process based on the above research. At the same time, the description and definition of evolution rules will be completed and then the simulation process and simulation algorithm are studied, to achieve the simulation calculation to the mission reliability of software system, and complete the mission reliability metrics of software system.

Conclusion

The software dynamic evolution has a very important impact on mission reliability, which is extremely difficult to estimate, especially in large and complex multi-task software systems. This paper starts from the concept of software evolution , presents the three basic questions about the software mission reliability, that is software architecture modeling based on task decomposition, task path based on evolution rules and mission reliability simulation modeling based on multi-level simulation, and gives the basic idea to solve the above problems to form a solution orienting modeling methodology for mission reliability modeling problems of software evolution, and lays a better foundation for subsequent mission reliability research of software evolution.

References

Alkhalid A; Alshayeb M; and **Mahmoud SA** (2011) Software Refactoring at The Package Level using Clustering Techniques. *IET Software*, **5** (3): 276-284.

Bavota G (2012) Using Structural and Semantic Information to Support Software Refactoring. *In:* 34th *IEEE International Conference on Software Engineering (*ICSE '12*),* 2-6 June 2012. Zurich, Switzerland, pp1479-1482.

Burger S; and **Hummel O** (2011) Towards Automatic Reconfiguration of Aviation Software Systems, *In: IEEE 35th Annual Computer Software and Applications Conference Workshops (COMPSACW)*, 17-22 July 2011, Munich, Germany. pp200-205.

Gracioli G; and **Fröhlich AA** (2010) A Dynamic Software Reconfiguration Infrastructure for Embedded Systems. *In:* 17th *IEEE International Conference on Telecommunications(ICT)*, 4-7 April 2010, Singapore, pp981-988.

Guo SC; Huang HZ; **Wang ZL;** and **Xie M** (2011) Grid Service Reliability Modeling and Optimal Task Scheduling Considering Fault Recovery. *IEEE Transactions on Reliability*, **60** (1): 263-274.

Hanakawa N (2011) A Process Refactoring for Software Development with Process Complexity and Activity Priority Lists. *In: Software Measurement, 2011 Joint Conference of the 21st Int'l Workshop on and 6th Int'l Conference on Software Process and Product Measurement (IWSM-MENSURA)*, 3-4 Nov. 2011, Nara, Japan. pp 209-214.

Khalgui M; Mosbahi O; Li Z; and **Hanicsh HM** (2011) Reconfiguration of Distributed EmbeddedControl Systems. *IEEE/ASME Transactions on Mechatronics*,**16** (4): 684-694.

Koru AG; and **El Emam K** (2010) Theory of Relative Dependency Higher Coupling Concentration in Smaller Modules and its Implications for Software Refactoring and Quality. *IEEE Software*, **27** (2): 81-89.

Meanaeatra P; Rongviriyapanish S; and **Apiwattanapong T** (2011) Using Software Metrics to Select Refactoring for Long Method Bad Smell, *In: Electrical Engineering/ Elecronics, Computer, 8th International Conference on Telecommunications and Information Technology(ECTI-CON)*, 17-19 May 2011, Khon Kaen, Thailand, pp. 492-495.

Mens T; and **Tourwe T** (2004) A Survey of Software Refactoring, *IEEE Transactions on Software Engineering*, **30** (2): 126-139.

Mortensen M; Ghosh S; and **Bieman JM** (2012) Aspect Oriented Refactoring of Legacy Applications: an Evaluation. *IEEE Transactions on Software Engineering*, **38** (1): 118-140.

Murphy-Hill E; Parnin C; and **Black AP** (2009) How we Refactor and How We Know it. *In: 31st IEEE International Conference on Software Engineering(ICSE)*, 16-24 May 2009, Vancouver, British Columbia, Canada, pp287-297.

Soares G (2010) Making Program Refactoring Safer, *In: ACM/IEEE 32nd Conference on Software Engineering (ICSE) vol. 2*, 2-8 May 2010, Cape Town, South Africa, pp521-522.

Strunk EA; and **Knight JC** (2006) Dependability through Assured Reconfiguration in Embedded System Software. *IEEE Transactions on Dependable and Secure Computing*, **3** (3): 172-187.

Villavicencio G (2012) A New Software Maintenance Scenario Based on Refactoring Techniques. *In: 16th European Conference on Software Maintenance and Reengineering(CSMR)*, 27-30 March 2012, Szeged, Hungary, pp341-346.

Watkins BC (2006) Integrated Modular Avionics: Managing the Allocation of Shared Intersystem Resources. *In:* 25th *Digital Avionics Systems Conference, 2006 IEEE/AIAA, 15-19 Oct. 2006,* Portland OR, pp1-12.