Transactions of the Korean Nuclear Society Spring Meeting
Jeju, Korea, May 10-11, 2007

A Study of Dependability Enhancement for Safety-Critical Software in NPP

Y. M. KIM and C. H. Jeong
Korea Institute of Nuclear Safety
Ymkim@kins.re.kr

1. Introduction

Recently, with the rapid development of digital computer and information processing technologies, nuclear I&C (Instrument & Control) system which needs safety-critical function has adopted digital technologies.

For software dependability of the safety-critical system, we are using many software engineering standards as well as software testing. The time needed to obtain a statistically significant number of failures makes software testing impractical. Also, when we use software engineering standards, we focus primarily on development processes. But the relationship between meeting the standards and obtaining the high dependability software is not well established.

In this paper, we approach fault-injection techniques with safety analysis of safety-critical software in Nuclear Power Plants. Fault-injection techniques can help developers move beyond the practical limitations of testing. Fault-injection techniques focus on software behavior, not structure; process-oriented techniques cannot measure behavior as precisely.

This paper is structured as follows. In Section 2, we described dependability of the software, testing and process-oriented assessment and fault-injection techniques. In Section 3, we present our approach about dependability enhancement using fault-injection methods. Section 4 shows examples of the approach and we conclude the paper in Section 4.

2. Related Work

2.1 Dependability of the Software

Dependability is defined as the trustworthiness of a computer system such that reliance can justifiably be placed on the service it delivers. Dependability may be viewed according to different, but complementary, properties, which enable the attributes of dependability to be defined safety, security, and reliability. Dependability is a qualitative system attribute that is quantified through specific measures. Dependability is often evaluated empirically through life testing. However, the time needed to obtain a statistically significant number of failures makes life testing impractical for most fault-tolerant computers. Instead, analytical modeling is typically used to predict dependability [4].

2.2 Software Testing and Process-Oriented Assessment

Testing is the most popular methods of assessing software quality which concentrate on product not process before the current emphasis on process oriented software engineering standards. The direct measurement of software quality via testing is impractical. Normally, we can use random testing to make statistically reasonable predictions about future behavior of software. But, the number of test required for getting high reliability is impractical because the testing effort and cost required to establish a high confidence about the dependability are too high. Process oriented assessment techniques are currently popular because of a reaction to the intractability of adequate software testing [1].

2.3 Fault Injection Techniques

Fault-injection techniques are used on physical systems to test their robustness. Fault-injection technique is practical for measuring software quality. Fault injection can be used to observe the behavior of the error or failure associated with each type of fault component. Fault injection is important to evaluating the dependability of computer systems and can directly evaluate dependability metrics. We can categorize software injection methods on the basis of fault injection time. The faults can be injected during compile-time or during run-time [2].

2.3.1. Compile-time injection

This method injects fault/errors into the source code or assembly code of the target program to emulate the effect of software and transient faults. The source code which is added faulty code alters the target program images, causing erroneous behavior. But, the compile time injection techniques can only be used when the target system code is available, which is a major weak point as the source code is not available for most COTS components.

2.3.2. Run-time injection

During run-time, run-time injection methods are commonly using triggering mechanisms such as time-out, exception/trap, and code insertion. This method can be used even when the source code is not available. But, because the very low level fault injection is executed the faults are injected, the relation between the caused

errors and the software faults defined at high levels is not definite [5].

2.3.2. Consideration of fault-injection

Usually source codes of target systems are not available. In [5], they compared the cost and accuracy of compile-time software implemented fault-injection techniques (CTSWIFI) and run-time software implemented fault-injection techniques (RTSWIFI). They show that CTSWIFI is up to 4 times more expensive in terms of execution time while the accuracy of them is almost same.

3. Enhancing Dependability by Fault-Injection

According to [3], the dependability estimated from the critical set (critical and rarely executed code) failures would thus be close to the actual dependability (based on all failures). They said that critical operations constitute only a small part of the operational profile and failures of critical operations are the major threat to system dependability. In that experiment, critical set included redundancy management, exception handling, initialization and calibration routines. Also, as suggested in [8], fault injection could have played a major role in avoiding hazard.

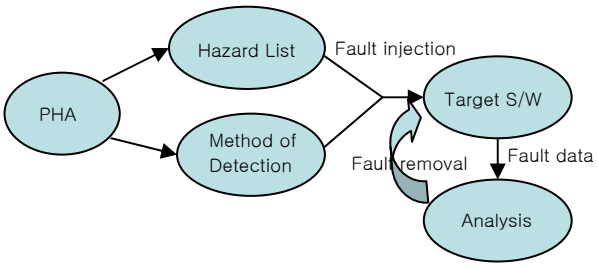

Figure 1. Overall approach

In our study, we use fault-injection method with Software Preliminary Hazard Analysis (PHA) which is executed in the early stage of the software life cycle. Usually, PHA contains hazardous system states, sequences of actions that can cause the system to enter a hazardous state, sequence of actions intended to return the system from a hazardous state to a nonhazardous state, and actions intended to mitigate the consequences of accidents [7].

We considered hazard lists of PHA as critical set. If we concentrate our test effort to critical set using fault-injection methods, we can enhance dependability of the target software. Figure 1 shows the overall approach about that. We can achieve dependability enhancement through fault removal, also find more effective fault-detection methods through information about fault detection coverage of used fault-detection methods. For each hazard list, we can examine that the fault coverage of the selected fault detection methods through fault-injection tests.

4. Examples of the Approach

In the PHA report of the Ulchin 5&6, they used fault detection methods such as inter-channel comparison, periodic test, trip indication, loss of heartbeat, trouble alarm and range limit check [6]. Also, they used safety hazard control verification methods such as code inspection, S/W testing, document review and system validation testing.

In case of Ulchin 5&6, we can find that the methods for the hazard control are mainly software testing and document review. Before, we showed about the limit of the software testing and process oriented standards which recommend V&V during software life cycle. We can put fault-injection methods with the safety hazard control verification methods. Fault-injection techniques could be exploited for dealing limits of testing and process oriented standards.

5. Summary

In this paper, we described an approach for enhancing software dependability of digital I&C systems using PHA and fault injection technique. Fault-injection techniques with PHA could have played a primary part in enhancing safety. Future work will be made to show that the proposed method is a useful for enhancing software dependability of the digital I&C system through dependability modeling. And, we will perform the case study for this research. We expect that this approach might be applied to evaluation of safety-critical software for new plants such as Shin-Kori 3 & 4 NPP and operating plants.

REFERENCES

[1] Jeffrey M. Voas and Keith W. Miller, Using Fault Injection to Assess Software Engineering Standards, Software Engineering Standard Symposium, 1995.

[2]Mei-Chen Hsueh, Timothy K. Tsai, and Ravishankar K. Iyer, Fault Injection Techniques and Tools, IEEE Computer, 1997

[3]Dong Tang and Herbert Hecht, An Approach to Measuring and Assessing Dependability for Critical Software Systems, 8th IEEE International Symposium on Software Reliability Engineering, 1997.

[4]Jean Arlat, Martine Aguera, Fault Injection for Dependability Validation: A Methodology and Dome Applications, IEEE Transaction on Software Engineering, 1990

[5] Diamantino Costa and Tiago Rilho, ESFFI – A novel technique for the emulation of software faults in COTS components, IEEE, 2001

[6]Software Safety Plan for DPPS & DESPAS-AC for Ulchin NPP U 5&6, rev 02.

[7] IEEE Standard for Software Safety Plans, 1994

[8]J. Voas, Software Fault Injection: Growing 'Safer' Systems, Proceedings of the IEEE Aerospace Conference, 1997