Development of Software Testing Environment for Safety-critical Software Reliability Quantification

Sang Hun Lee
Rensselaer Polytechnic Institute
Troy, NY

Eun-chan Lee
Korea Hydro & Nuclear Power Co., Ltd.
Gyeongju, Republic of Korea

Seung Jun Lee
Ulsan National Institute of Science and Technology
Ulsan, Republic of Korea

Hyun Gook Kang
Rensselaer Polytechnic Institute
Troy, NY

Sung Min Shin
Korea Atomic Energy Research Institute
Daejeon, Republic of Korea

KEYWORDS

Nuclear Power Plant, Digital I&C System, Software Reliability, Software Testing

ABSTRACT

Software has been used to digitalize many instrumentation and control (I&C) systems in nuclear power plants (NPPs). Since software failure may cause the common cause failure of the processor modules in digital I&C systems, the reliability of the software used in NPP safety-critical I&C systems must be quantified and verified with proper test cases and environment. In this study, a software testing method using the developed simulation-based software test-bed is proposed. In the test-bed, the microprocessor architecture of the programmable logic controller (PLC) used in NPP safety-critical applications is emulated and the execution behavior of the microprocessor at each machine instruction line is captured. The effectiveness of the proposed method is demonstrated with the safety-critical trip logic software of a fully digitalized reactor protection system (IDiPS-RPS). The method provides a practical way to conduct software testing in order to prove the software to be error-free while effectively reducing the software testing effort compared to existing software testing methods.

INTRODUCTION

Recently, existing circuit-based hardware instrumentation and control (I&C) systems in nuclear power plants (NPPs) are being replaced with microprocessor-based digital I&C systems due to analog systems approaching obsolescence and to functional advantages of digital systems [1]. Compared to the analog I&C systems, the digital systems provide advanced performance in terms of accuracy and computational capabilities and have potential for improved capabilities such as fault tolerance and diagnostics [2]. However, the use of microprocessor-based digital systems in NPP safety I&C systems triggered a big challenge in incorporating the risk characteristics of digital systems into the probabilistic risk assessment (PRA) model of a NPP in order to estimate the reliability of software used in digital systems and its risk effect on the NPP safety [3, 4].

The objective of this study is to develop a simulation-based software test-bed for the white-box testing of the NPP safety-critical software. The test-bed is developed by emulating the microprocessor architecture of a programmable logic controller (PLC) used in NPP digital I&C system and capturing its behavior at each machine instruction line while the software executes its dedicated safety function. The effectiveness of the proposed software testing framework is demonstrated with the safety-critical trip logic software of a fully digitalized reactor protection system (IDiPS-RPS), developed under the Korea Nuclear Instrumentation & Control Systems (KNICS) project [5].

The proposed method can effectively reduce the software testing time and effort by emulating the software behavior in a machine-level compared to the existing black-box testing. In addition, the test result of the safety-critical software from the suggested method can be utilized to support the software reliability quantification of the NPP digital I&C systems and applied to the NPP PRA model to analyze the effect of software failure on the digital system availability or the NPP risk.

TARGET SYSTEM

The IDiPS-RPS is a digitalized RPS developed for newly constructed NPPs as well as for upgrading existing analog-based RPS [5]. It has the same function as an analog RPS which automatically generates a reactor trip signal and engineered safety features actuation signals whenever there is a demand. Figure 1 illustrates the architecture of IDiPS-RPS which consists of four redundant channels of bistable processors (BPs) and coincidence processors (CPs) for its dedicated safety functions.

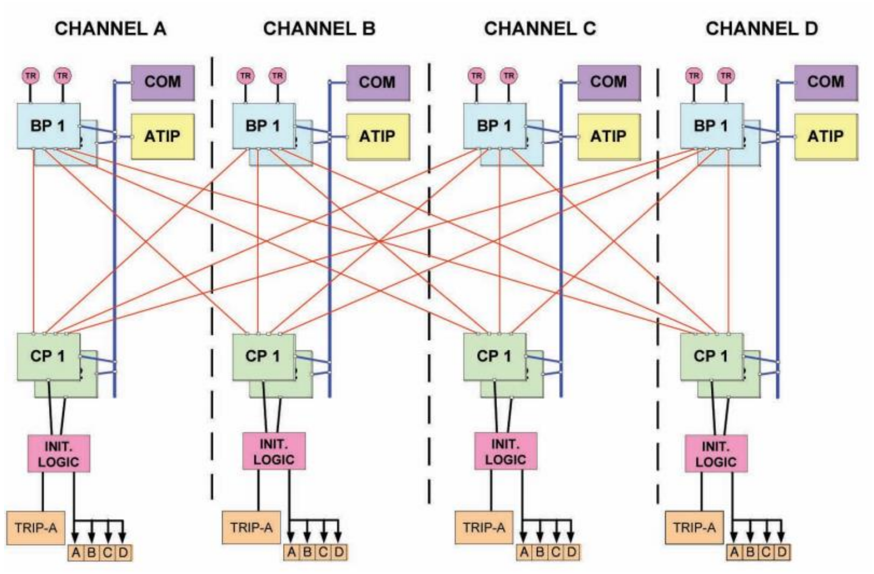
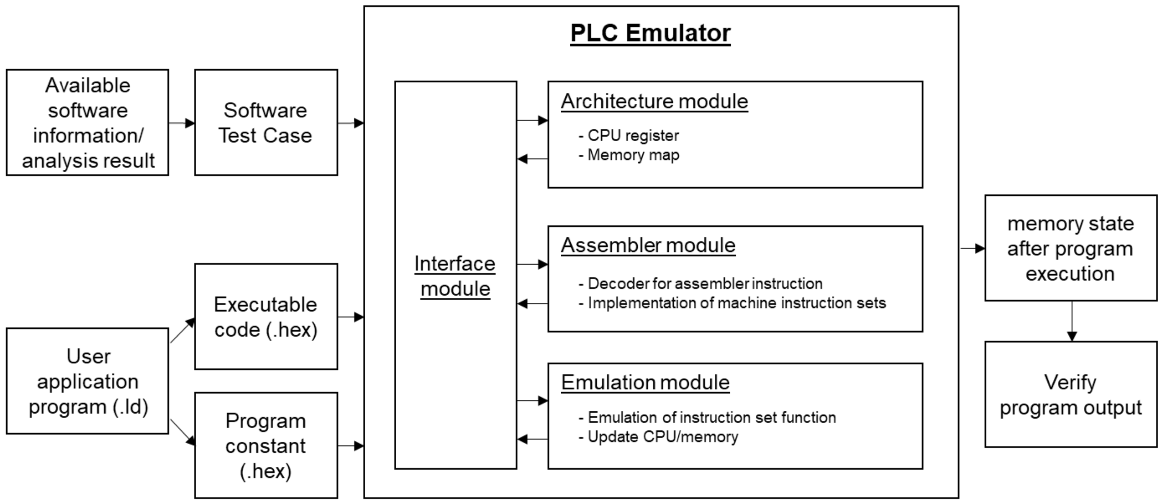


Figure 1. An IDiPS-RPS architecture.

In each IDiPS-RPS channel, the BP determines the reactor trip state by comparing the process variables measured from the plant sensors with the predefined pre-trip or trip set-points. The CP generates a final hardware-actuating pre-trip or trip signal by conducting voting logic with the pre-trip or trip signals transferred from BPs in all channels. The BP and CP processors are configured with the safety grade PLC platform (POSAFE-Q) [6] and the safety signal generation logic in each processor is implemented as a software in the PLC platform. The application software is developed in the form of function block diagram and ladder diagram (FBD/LD) programming language [7]. In the implementation phase of software lifecycle, the FBD/LD programs are compiled to machine instruction codes which can be loaded to the PLC memory area and executed by the PLC microprocessor [8].

DEVELOPMENT OF SOFTWARE TEST-BED

In this study, in order to simulate the software behavior given the states of software input and internal variables, a software test-bed is developed which captures both the internal (CPU and memory architecture) and external (the states of program input and output variables) aspects of the PLC scan cycle [9]. The software test-bed can be used to check whether the correct output is generated by the software program using the test cases provided by the software tester. Figure 2 shows an overview of the developed test-bed structure.



PLC Emulator
Available software information/ analysis result
Software Test Case
User application program (.ld)
Executable code (.hex)
Program constant (.hex)
Interface module
Architecture module
- CPU register
- Memory map
Assembler module
- Decoder for assembler instruction
- Implementation of machine instruction sets
Emulation module
- Emulation of instruction set function
- Update CPU/memory
memory state after program execution
Verify program output

Figure 2. Overview of simulation-based test-bed for NPP safety software testing.

The test-bed is executed based on three basic operation processes of the PLC microprocessor: 1) fetch, 2) decode, and 3) execute [10]. In the fetch phase, the executable code which is uploaded to the memory map is fetched from the *Architecture module*. In the decode phase, the fetched executable code in a binary form is decoded into specific instruction set by the *Assembler module*. In the execute phase, the operation of the decoded instruction set is performed by the *Emulation module* and the operation results are stored in the CPU register or the memory. If necessary, the registers which represents the status of the microprocessor are updated during the execute phase. When the software testing is conducted using the developed test-bed, the executable code of the safety software which was compiled from the FBD/LD language and the constant files which contain the memory map of the input (e.g. pressure, water level in NPP) and the internal variable (e.g. counter, test parameters) used in the application software are loaded to the test-bed. Then, the software test cases are uploaded to the memory area emulated in the *Architecture module*. After all machine instruction lines of safety software are executed, the final status of CPU registers and memory map is saved as an output file for every software test cases. The generated output files can be used to verify whether correct output is generated given the test case by checking the specific memory area that corresponds to the dedicated safety function of the application program such as trip signal.

CASE STUDY

The proposed software testing method was applied to a KNICS IDiPS-RPS BP trip logic software as a case study. In the IDiPS-RPS system, the BP compares the process variables which are transmitted from the measurement sensors in NPP with the pre-defined trip set-points. The function of each module is implemented as a software logic in a binary format in the PLC memory map. Among 19 BP trip logics, the pressurizer-pressure-low trip logic which has a variable trip set-point and operator bypass function was chosen as a case study to demonstrate the effectiveness of the proposed software test method. The pressurizer-pressure-low trip logic is one of the most complicated logics among BP trip logic modules which include various functions such as operator bypass, reset delay timer and the set-point reset by the operator. Figure 3 shows the operation logic of the pressurizer-pressure-low trip [11].

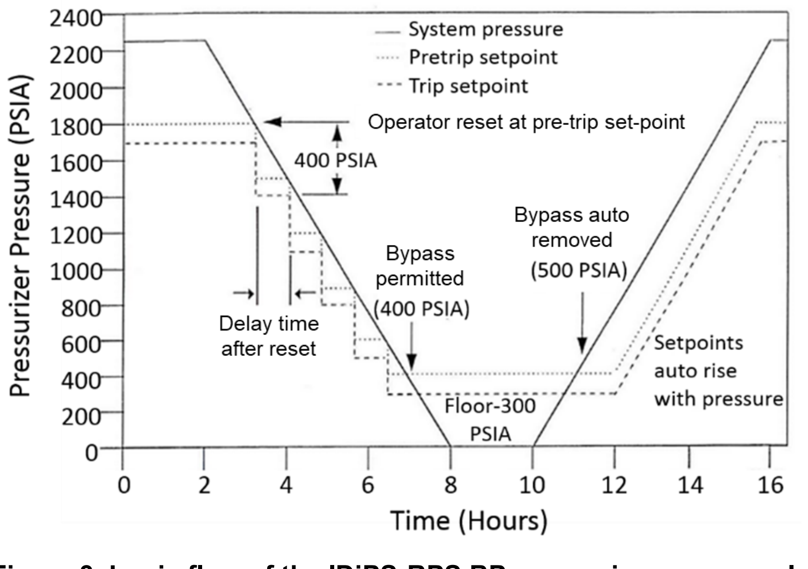


Figure 3. Logic flow of the IDiPS-RPS BP pressurizer-pressure-low trip logic.

The test cases were developed focusing on the trip generation situation by the BP software and constructed based on the operational profile of the software input and internal variables. For each test case, the test results are generated by capturing the final state of output variable after the software program is executed in the developed test-bed. Since the test cases are developed focusing on the trip initiation condition by the target software, the test case is verified as an error-free portion and saved as correct output if the value of the trip variable corresponds to true. However, if there is any test case that results in the value of trip signal variable as false, it is saved as wrong output which should be reviewed and debugged, and the test should be restarted from the beginning, if necessary. Figure 4 illustrates the procedure of software testing using the developed software test-bed with the test cases.

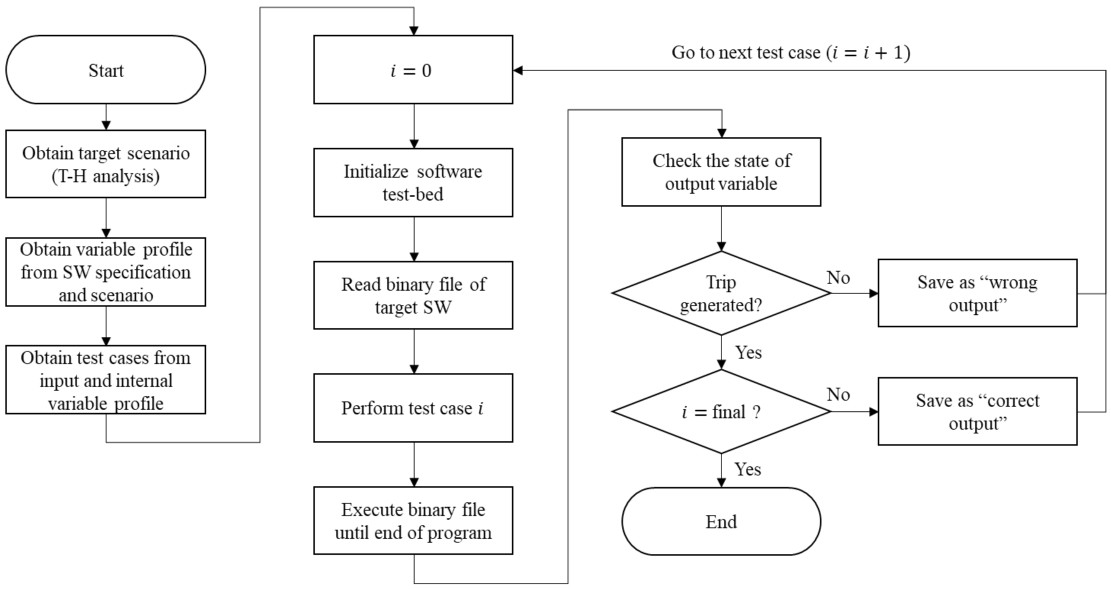
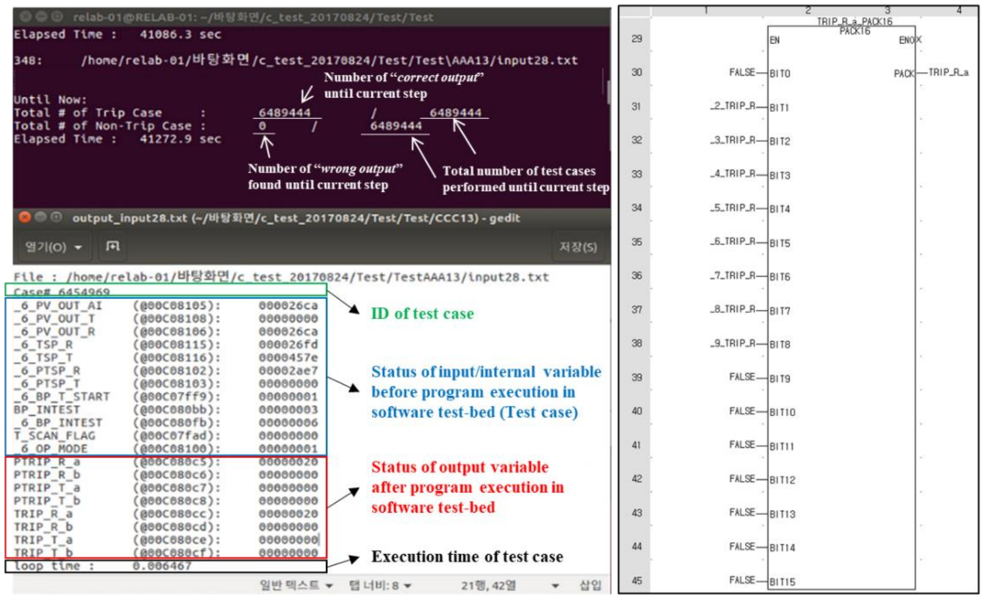

Start
Obtain target scenario (T-H analysis)
Obtain variable profile from SW specification and scenario
Obtain test cases from input and internal variable profile
$i = 0$
Initialize software test-bed
Read binary file of target SW
Perform test case i
Execute binary file until end of program
Go to next test case ($i = i + 1$)
Check the state of output variable
Trip generated?
No
Save as "wrong output"
Yes
i = final ?
No
Save as "correct output"
Yes
End

Figure 4. Testing procedure using simulation-based software test-bed.

Figure 5 shows a part of test results using the test cases developed for the trip initiation condition of the pressurizer-pressure-low trip logic software as a case study. The output variable of the BP trip logic software is TRIP_R_a variable which is sent to CP as a trip signal for the voting logic. The TRIP_R_a variable is packed with trip signal output of various trip logics. For example, the _6_TRIP_R variable which is the trip signal for pressurizer-pressure-low trip logic is packed at 5th bit of the TRIP_R_a variable. As shown in Figure 5, the test result showed that the state of the TRIP_R_a variable after program execution is set to 0x20 which indicates that the 5th bit of the trip signal (pressurizer-pressure-low trip signal) is set to 0x1 meaning that the software generated correct output for given test case. In result, all 705,892,684 test cases developed from the previous section generated the trip signal. The test was conducted in 76.04 hours using sixteen 3.60 GHz logical processors, that is 6.205 ms were spent per test case in average using the software test-bed.



```
relab-01@RELAB-01: ~/바탕화면/c_test_20170824/Test/Test
Elapsed Time :    41086.3 sec

348:      /home/relab-01/바탕화면/c_test_20170824/Test/Test\AAA13/input28.txt

Until Now:
Total # of Trip Case     :     6489444      /    6489444
Total # of Non-Trip Case :     0       /    6489444
Elapsed Time :    41272.9 sec
```

```
output_input28.txt (~/바탕화면/c_test_20170824/Test/Test/CCC13) - gedit
File : /home/relab-01/바탕화면/c_test_20170824/Test/TestAAA13/input28.txt
Case# 6454969
_6_PV_OUT_AI     (@00C08105):    000026ca
_6_PV_OUT_T      (@00C08108):    00000000
_6_PV_OUT_R      (@00C08106):    000026ca
_6_TSP_R         (@00C08115):    000026fd
_6_TSP_T         (@00C08116):    0000457e
_6_PTSP_R        (@00C08102):    00002ae7
_6_PTSP_T        (@00C08103):    00000000
_6_BP_T_START    (@00C07ff9):    00000001
BP_INTEST        (@00C080bb):    00000003
_6_BP_INTEST     (@00C080fb):    00000006
T_SCAN_FLAG      (@00C07fad):    00000000
_6_OP_MODE       (@00C08100):    00000001
PTRIP_R_a        (@00C080c5):    00000020
PTRIP_R_b        (@00C080c6):    00000000
PTRIP_T_a        (@00C080c7):    00000000
PTRIP_T_b        (@00C080c8):    00000000
TRIP_R_a         (@00C080cc):    00000020
TRIP_R_b         (@00C080cd):    00000000
TRIP_T_a         (@00C080ce):    00000000
TRIP_T_b         (@00C080cf):    00000000
loop time :      0.006467
```

Number of "correct output" until current step
Number of "wrong output" found until current step
Total number of test cases performed until current step
열기(O)
저장(S)
ID of test case
Status of input/internal variable before program execution in software test-bed (Test case)
Status of output variable after program execution in software test-bed
Execution time of test case
일반 텍스트
탭 너비: 8
21행, 42열
삽입
TRIP_R_a_PACK16
PACK16
EN
ENO
FALSE
BIT0
_2_TRIP_R
BIT1
_3_TRIP_R
BIT2
_4_TRIP_R
BIT3
_5_TRIP_R
BIT4
_6_TRIP_R
BIT5
_7_TRIP_R
BIT6
_8_TRIP_R
BIT7
_9_TRIP_R
BIT8
FALSE
BIT9
FALSE
BIT10
FALSE
BIT11
FALSE
BIT12
FALSE
BIT13
FALSE
BIT14
FALSE
BIT15
PACK
TRIP_R_a

Figure 5. An example of test result for IDiPS-RPS BP pressureizer-pressure-low trip logic software.

CONCLUSION

In this study, the software test method utilizing simulation-based software test-bed was proposed. The software test-bed was developed considering the characteristics of the safety-critical PLC and the CPU architecture and memory map of the PLC microprocessor. The software test case was developed in consideration of the digital signal processing features of the PLC as well as the plant thermo-hydraulics data for the plant transients or accidents in a NPP. As an application of the proposed software test method, the software test cases were developed for the pressurizer-low-pressure trip of IDiPS-RPS BP software logic and tested by capturing the state of output variable stored in the memory map after the end of the trip logic program.
An important characteristic of the proposed software test approach is that the developed software test-bed can effectively reduce the software testing time per test case by emulating the software behavior given the software input and internal states in machine

language level compared to the existing black-box testing. In addition, the number of test cases to achieve exhaustive testing of the safety-critical software can be quantitatively derived. Therefore, the proposed software test method can be used to support the software reliability quantification of NPP safety-critical I&C applications and further ensure the safety of the software-based digital systems.
Although this study focuses on the safety nuclear software which is relatively simple, the exhaustive testing of a more complicated software including non-safety nuclear software may not be possible due to its size or complexity. In this case, a probability-based approach such as input-profile-based software testing method [] can be used with the software test-bed proposed in this study in order to ensure a very low software failure probability.
The proposed framework focuses on verifying that the software logic is error-free when demand comes, other causes of software error should be investigated in consideration of the running environment. While the application software can be tested as error-free using the framework proposed in this study, the software will not generate a safety signal if the operating system kernel does not properly call the application software, or if there is any error in the hardware module that affects the application or the operating system software. In addition, other causes of potential software error, such as wrong input by operator mistake, noise from sensors or the signal transmission path, and incorrect or inadequate software requirements also need to be considered to completely model the software failure and estimate the software reliability.

ACKNOWLEDGEMENT

This work was supported by the project of 'Evaluation of human error probabilities and safety software reliabilities in digital environment (L16S092000),' which was funded by the Central Research Institute (CRI) of the Korea Hydro and Nuclear Power (KHNP) company.

REFERENCES

1. M. Hassan, W.E. Vesely. “Digital I&C systems in nuclear power plants: risk-screening of environmental stressors and a comparison of hardware unavailability with an existing analog system,” Brookhaven National Laboratory, NUREG/CR-

6579, 1998.
2. National Research Council. “Digital instrumentation and control systems in nuclear power plants: safety and reliability issues,” National Academies Press, 1997.
3. H. G. Kang, T. Sung. “An analysis of safety-critical digital systems for risk-informed design,” Reliability Engineering & System Safety, 78, pp. 307-314, (2002).
4. K. Korsah, M. D. Muhlheim, R. Wood. “A Qualitative Assessment of Current CCF Guidance Based on a Review of Safety System Digital Implementation Changes with Evolving Technology,” Oak Ridge National Laboratory, ORNL/SR-2016/148, 2016.
5. K. C. Kwon, M. S. Lee. “Technical review on the localized digital instrumentation and control systems,” Nuclear Engineering and Technology, 41, pp. 447-454, (2009).
6. M. Lee, S. Song, D. Yun. “Development and application of POSAFE-Q PLC platform,” International Atomic Energy Agency, IAEA-CN-194, 2012.
7. International Electrotechnical Commission. “Programmable Controllers - Part 3: Programming Languages,” International Electrotechnical Commission, IEC 61131-3, 1993.
8. K. Koo, B. You, T. W. Kim, S. Cho, J. S. Lee. “Development of application programming tool for safety grade PLC (POSAFE-Q),” Transactions of the Korean Nuclear Society Spring Meeting, Chuncheon, Korea, (2006).
9. J. Palomar, R. H. Wyman. “The programmable logic controller and its application in nuclear reactor systems,” U.S. Nuclear Regulatory Commission, NUREG/CR-6090, 1993.
10. W. Bolton. “Programmable logic controllers,” Newnes, 2015.
11. J. G. Choi, D. Y. Lee. “Development of RPS trip logic based on PLD technology,” Nuclear Engineering and Technology, 44, pp. 697-708, (2012).
12. H. G. Kang, H. G. Lim, H. J. Lee, M. C. Kim, and S. C. Jang, “Input-profile-based software failure probability quantification for safety signal generation systems,” Reliability Engineering & System Safety, 94(10), pp. 1542–1546, (2009).