Yan Zhang, Xiaomei Shen, Yaming Zhang, Xinyu Han and Longli Tang

China Institute of Marine Technology & Economy, Beijing, China.

ACSE 2023

Paper ID：A129

Research on Simulation Technology Based on Fault Injection

Background

Modern warfare often involves joint formation, multi-platform, and multi weapon system combat systems, and the battlefield environment is becoming increasingly complex. In this context, software needs to be able to stably and efficiently complete combat tasks in complex and ever-changing battlefield environments. The actual combat oriented software testing needs to focus on the verification of the interconnection, interoperability and interoperability capabilities of information system software, the actual combat mission capabilities, and the system combat technology indicators.

In the process of software testing, software system fault models can be studied from aspects such as system state faults, platform operation faults, and interface communication faults in the actual combat environment. Firstly generate fault constraint simulation data based on actual combat scenarios and test cases. Then inject faults into the target system, increase the probability of failure and failure of the target system. Finally, observe and recycle the response information of the system. Thus, the required experimental result data can be obtained.

Methods

By analysing the fields in common interface messages, such as device status messages, navigation messages, radar messages, etc., the representation rules for the message fields in software are obtained. The following table lists some common interface message field representation rules.The interaction behaviour between various devices (or subsystems) and software in a crosslinking environment is analysed mainly from aspects such as interaction time, feedback, synchronization, triggering, and process. Design error injection operators for crosslinking interfaces, including message error injection operators and communication error injection operators, for abnormal input and abnormal interaction processes.

Based on the method of message field constraints, message error injection operators are designed, as shown in the table below.

Table 1. Message error injection operator based on field constraint design.

Classification	Operator Name	Abbreviation	Example/Explanation
Based on message field constraints	Incomplete message data	DLR	A message has n bytes and is randomly missing one byte.
	Error value	VEV	The expected value of a field variable in the message is N, but it is actually notN.
	Variable accuracy too low	PLW	The value of floating point number shall be kept at N digits after the decimal point, and the variation shall be kept at N-M digits.
	Variable accuracy too high	PHG	The value of floating point number is kept at N digits after the decimal point, and the variation is kept at N+M digits.
	Enumerate values outside of a set	ECG	The machine code is of enumeration type, with values of 0102, 0103, and 0104; Mutate it to a value of 0208 outside the set.
	Signed data symbol bit error	SER	A field variable in the message changes from negative to positive or from positive to negative.
	Signed number 0 and sign bit change	SCZ	The variable in one field of the message is 0, and the symbol bit changes.
Based on message field mutation	Set certain bytes of the message to 0	BT0	The operator function is Byte_To0 (n, i, j, ...), where the range of values for n is [1, 2, 3, 4], which respectively represent the number of bytes used in the field; The value range of i and j is [0, 31].
	Set certain bytes of the message to 1	BT1	The operator function is Byte_To1 (n, i, j, ...), where the range of values for n is [1, 2, 3, 4], which respectively represent the number of bytes used in the field; The value range of i and j is [0, 31].

A total of 7 communication error injection operators have been designed, as shown in the following table.

Table 2. Interaction Behaviour Analysis Results.

Operator Name	Abbreviation	Example/Explanation
Continuous communication with random loss of data message	MLR	Data packets are sent once every n seconds, and one is randomly lost
No data input	MNO	Do not send any messages
Communication duplication	MRR	The software actually needs to send a message, mutated into an interface that repeatedly sends the message
Periodic data has a long cycle time. PLL sends periodic data with a smaller cycle interval	PLL	The data cycle interval for periodic transmission decreases
Periodic data cycle time is too short	PSS	The data cycle interval for periodic transmission increases
Periodic data with random cycle time (normal distribution)	PRR	The data period sent in a periodic manner is normally randomly distributed around the correct value
Data synchronization exception	DSE	When a message field is N, another field needs to be M, but the other field mutates to a value that is not M.

The fault injection testing process for crosslinking interfaces mainly includes test case generation, test execution, and test data collection. The specific process is shown in the following figure.

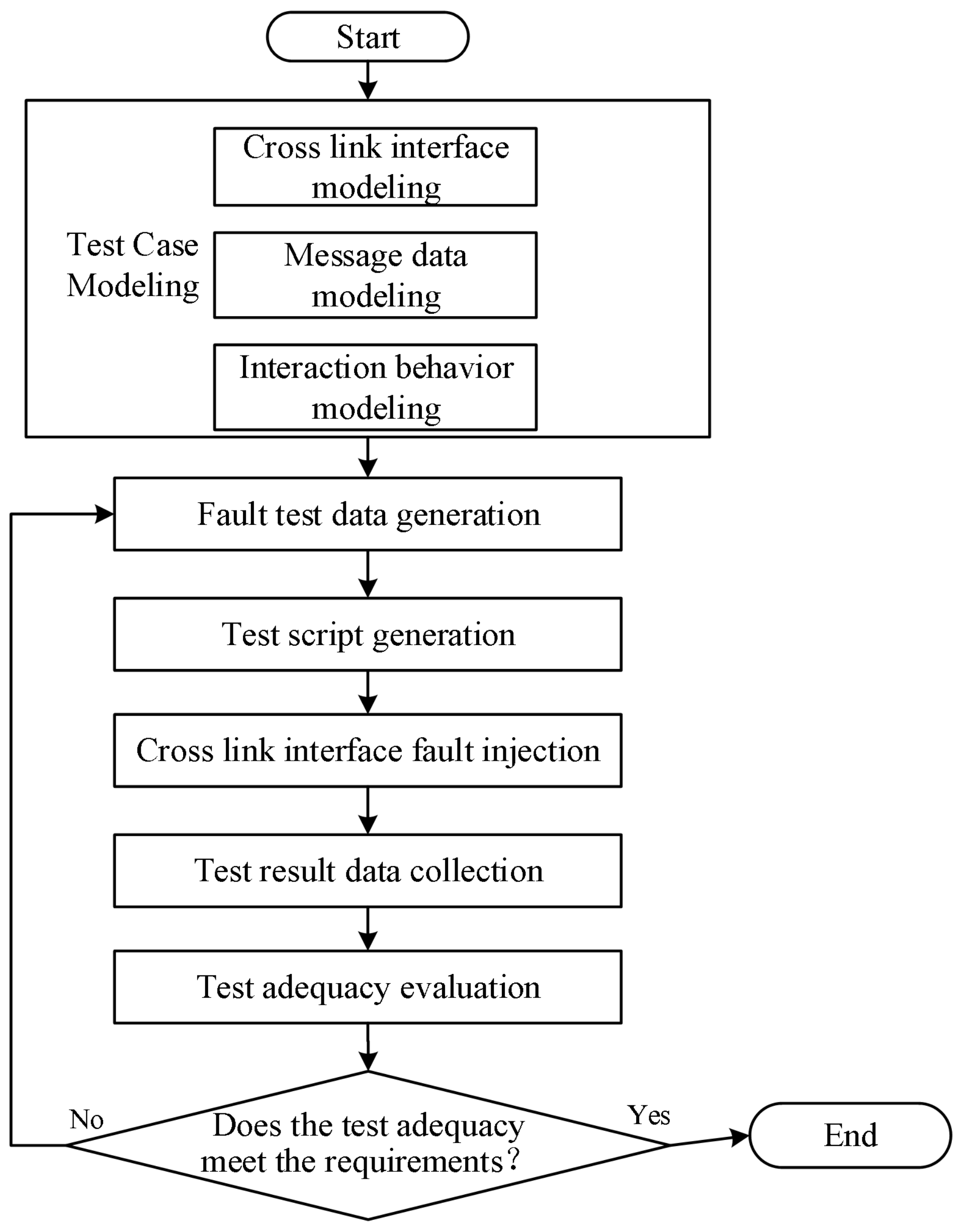
Start
Test Case Modeling
Cross link interface modeling
Message data modeling
Interaction behavior modeling
Fault test data generation
Test script generation
Cross link interface fault injection
Test result data collection
Test adequacy evaluation
No
Does the test adequacy meet the requirements?
Yes
End

Figure 1: Crosslinking Interface Fault Injection Test Process.

Results and discussion

This article studies fault injection technology from two aspects: cross-linking interface and external environment. A design approach for fault injection operators have been proposed, and the fault injection testing process and adequacy evaluation methods have been summarized, which helps to obtain system response data to faults and make the software testing results closer to the actual combat environment.

References

[1] W. Lu, R. Wang, C. Zeng, C. Liu and X. Wang, "A General Fault Injection Method Based on JTAG," 2018 Prognostics and System Health Management Conference (PHM-Chongqing), Chongqing, China, 2018, pp. 604-608 **[2]** D. Zhou, P. Yang and Q. Ou, "Analysis of Fault Characteristics Based on Clock Glitch Injection," 2021 IEEE 5th Advanced Information Technology, Electronic and Automation Control Conference (IAEAC), Chongqing, China, 2021, pp. 785-790 **[3]** A. Gangolli, Q. H. Mahmoud and A. Azim, "A Machine Learning Based Approach to Detect Fault Injection Attacks in IoT Software Systems," 2022 IEEE International Conference on Systems, Man, and Cybernetics (SMC), Prague, Czech Republic, 2022, pp. 2900-2905 **[4]** C. -A. Mao, Y. Xie, Y. Xie, H. Chen and H. Shi, "An Automated Fault Injection Platform for Fault Tolerant FFT Implemented in SRAM-Based FPGA," 2018 31st IEEE International System-on-Chip Conference (SOCC), Arlington, VA, USA, 2018, pp. 192-196 **[5]** J. Xu and P. Xu, "The Research of Memory Fault Simulation and Fault Injection Method for BIT Software Test," 2012 Second International Conference on Instrumentation, Measurement, Computer, Communication and Control, Harbin, China, 2012, pp. 718-722 **[6]** J. Goldberg, "Fault-type Injection Testing For Fault-tolerant Computers," Third Int'l Workshop on Integrating Error Models with Fault Injection, Annapolis, MD, USA, 1994, pp. 17-17.