2012 AEC/NRC Bilateral Technical Meeting
2012年AEC/NRC雙邊技術會議

Update of Digital I&C Research Activities in INER

Program for Software Development and Testing Research of Digital I&C Systems

Hui-Wen Huang

June 20, 2012

Institute of Nuclear Energy Research

Outline

- Background
- Schedule and Progress
- Study on Digital I&C software test coverage
- Study on software development and test related regulations and standards

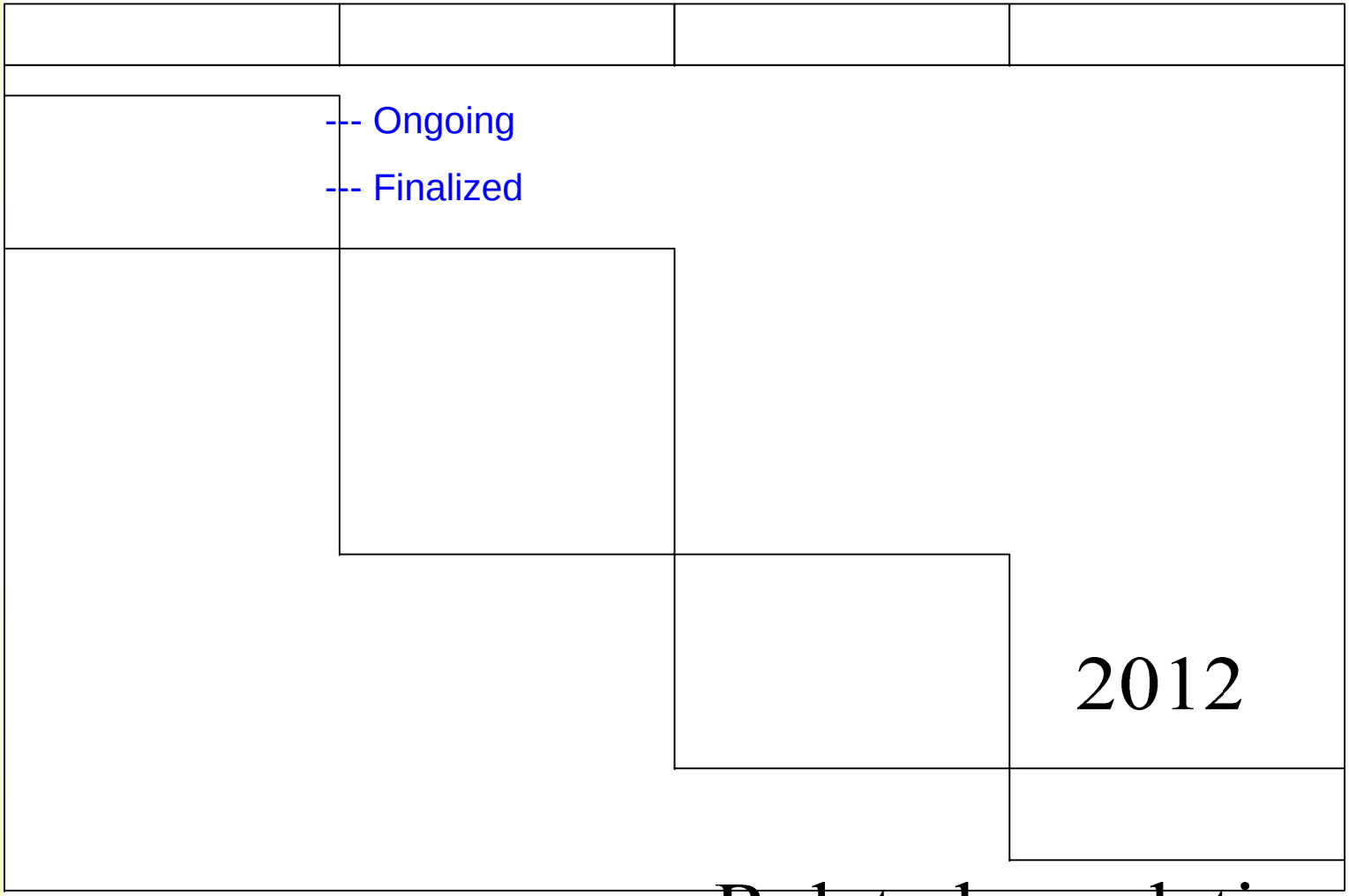
Background

- On May 20, 2011, NRC RES/DE staff met researchers from INER and the Taiwan Atomic Energy Council (AEC) Representative in Washington, DC, Science & Technology Division at the NRC Research Offices in Rockville, MD
- To discuss mutually agreed upon technical topics including “Program for Software Development and Testing Research of Digital I&C Systems” (which is under Diagnostics & Prognosis, INER request).
- This project is performed under the umbrella of TECRO-AIT Nuclear Cooperation project, and allocated within the Active Working Items “AE-IN-NR-C18 Instrumentation and Controls” whose subject is “Digital I & C Information Exchange”.

Background (cont.)

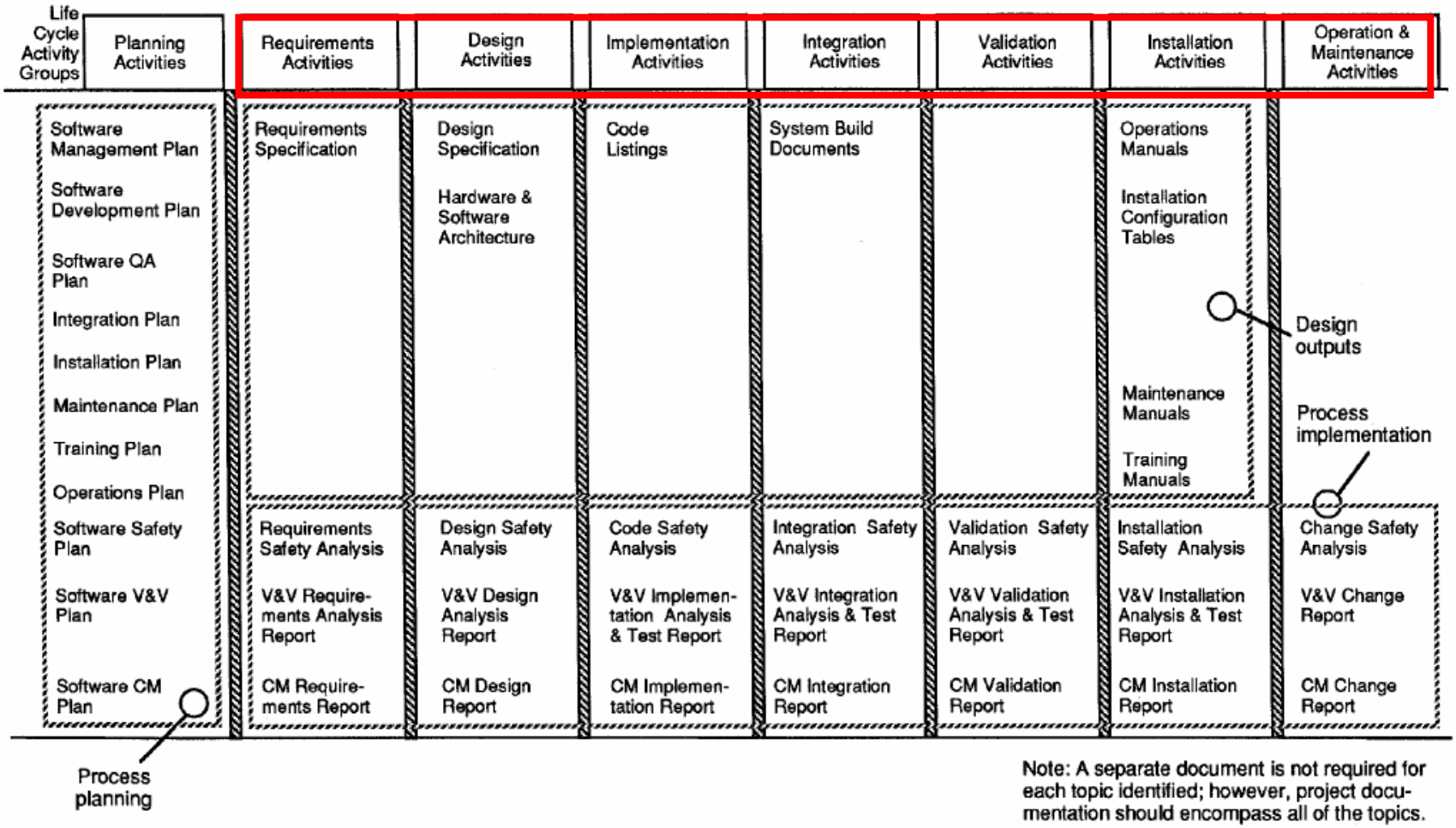
- The purpose of this research project is to enhance the review capability of digital instrumentation and control (I&C) system software development and testing.
- The expectation of this project is to obtain the experience of software development and testing.
- It can make the reviewers have sufficient capability to identify the in-depth software defects which are created by the software developer unintentionally.

Schedule and Progress

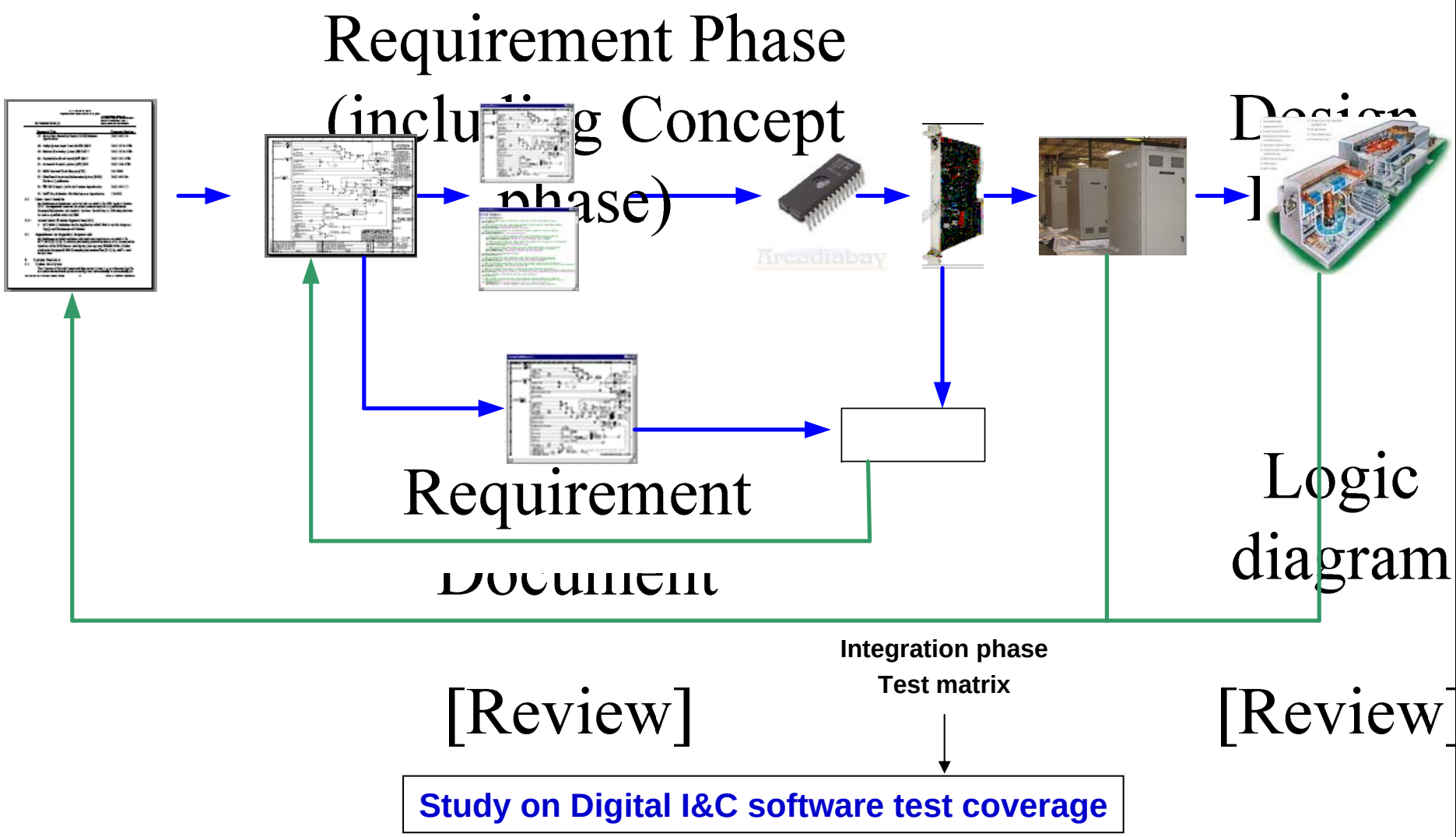


-Related regulations and standards

Study on Digital I&C software test coverage theory



Flow of Documents Through the Software Life Cycle (BTP 7-14)


Requirement Phase
(including Concept
phase)
Design
Requirement
Document
Logic
diagram
[Review]
Integration phase
Test matrix
[Review]
Study on Digital I&C software test coverage

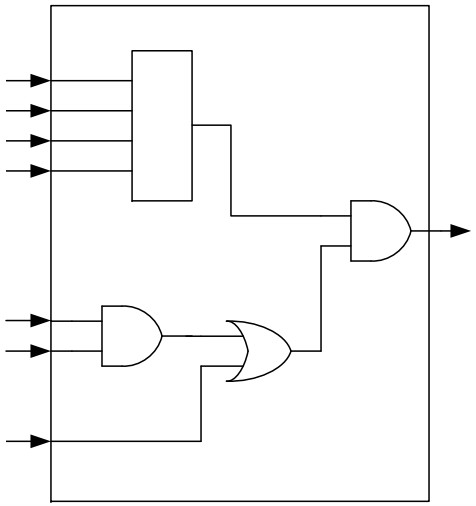
Study on Digital I&C software test coverage theory (cont.)

- Survey on test coverage theories
  - Non-deterministic Strategies
    - Constrained Array Test System (CATS)
    - Automatic Efficient Test Generator (AETG)
    - Genetic Algorithm (GA)
    - Random Combination Strategies (Rand)
  - Deterministic Strategies
    - Orthogonal Arrays (OA)
    - Covering Arrays (CA)
    - In Parameter Order (IPO)
    - IPO_T
    - Parameter-Order-General (IPOG)
    - IPOG-D

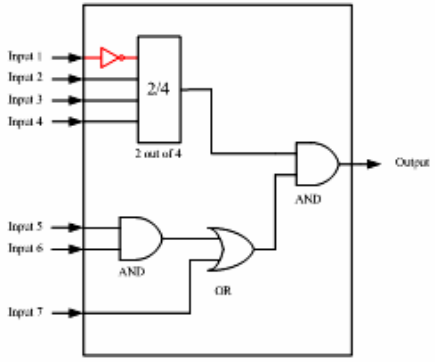
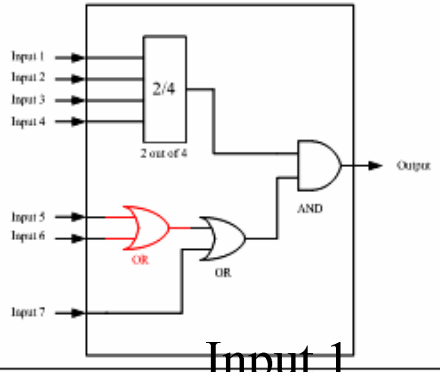
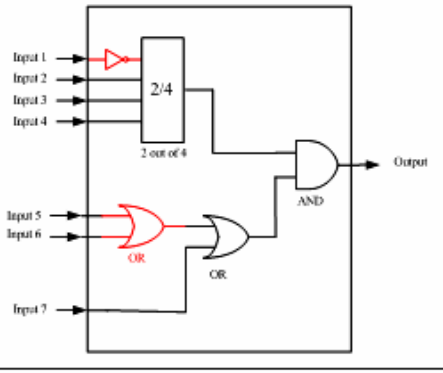
Study on Digital I&C software test coverage theory (cont.)

- Case analysis – 2 Strategies were selected
  - Orthogonal Arrays (OA)
  - In Parameter Order (IPO)

Study on Digital I&C software test coverage theory (cont.)



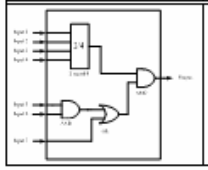
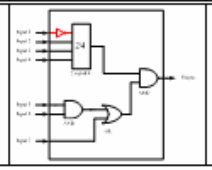
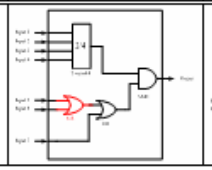
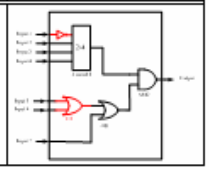
Basis Case
7 inputs
1 output
2^7 = 128 test input combinations
(2^{20} = 1,048,576)

Logic Fault Injection Teat Case 1	Logic Fault Injection Teat Case 2	Logic Fault Injection Teat Case 3
		

Inject software faults to simulate the software implementation faults

Study on Digital I&C software test coverage theory (cont.)

- Assumption
  - The basis case represents the independent software tool, the fault injected test case logic represents the control card.
  - If at least one output in the test different from the output in the basis case (the independent software tool), there must be logic faults in the control card or the logic from independent software tool.

基準型例 Basic Case	延時錯誤置入測試型例-1 Logic Fault Injection Test Case 1	延時錯誤置入測試型例-2 Logic Fault Injection Test Case 2	延時錯誤置入測試型例-3 Logic Fault Injection Test Case 3
			

Test No.								Basic Case	Logic Fault Injection Test Case 1	Logic Fault Injection Test Case 2	Logic Fault Injection Test Case 3	PO-2-wire
	Input 1	Input 2	Input 3	Input 4	Input 5	Input 6	Input 7	Output	Output	Output	Output	
1	0	0	0	0	0	0	0	0	0	0	0	1
2	0	1	0	0	0	0	0	0	0	0	0	
3	1	0	0	0	0	0	0	0	0	0	0	
4	1	1	0	0	0	0	0	0	0	0	0	
5	0	0	1	0	0	0	0	0	0	0	0	
6	0	1	1	0	0	0	0	0	0	0	0	6
7	1	0	1	0	0	0	0	0	0	0	0	
8	1	1	1	0	0	0	0	0	0	0	0	
9	0	0	0	1	0	0	0	0	0	0	0	
10	0	1	0	1	0	0	0	0	0	0	0	

Test No.								Basic Case	Logic Fault Injection Test Case 1	Logic Fault Injection Test Case 2	Logic Fault Injection Test Case 3	PO-2-wire
	Input 1	Input 2	Input 3	Input 4	Input 5	Input 6	Input 7	Output	Output	Output	Output	
11	1	0	0	1	0	0	0	0	0	0	0	
12	1	1	0	1	0	0	0	0	0	0	0	
13	0	0	1	1	0	0	0	0	0	0	0	
14	0	1	1	1	0	0	0	0	0	0	0	
15	1	0	1	1	0	0	0	0	0	0	0	
16	1	1	1	1	1	0	0	0	0	1	1	
17	0	0	0	0	1	0	0	0	0	0	0	
18	0	1	0	0	1	0	0	0	0	0	1	
19	1	0	0	0	1	0	0	0	0	0	0	
20	1	1	0	0	1	0	0	0	0	1	0	
21	0	0	1	0	1	0	0	0	0	0	1	
22	0	1	1	0	1	0	0	0	0	1	1	
23	1	0	1	0	1	0	0	0	0	1	0	
24	1	1	1	0	1	0	0	0	0	1	1	
25	0	0	0	1	1	0	0	0	0	0	1	25-3
26	0	1	0	1	1	0	0	0	0	1	1	
27	1	0	0	1	1	0	0	0	0	1	0	
28	1	1	0	1	1	0	0	0	0	1	1	
29	0	0	1	1	1	0	0	0	0	1	1	
30	0	1	1	1	1	0	0	0	0	1	1	
31	1	0	1	1	1	0	0	0	0	1	1	
32	1	1	1	1	1	0	0	0	0	1	1	
33	0	0	0	0	0	1	0	0	0	0	0	

Test No.								Basic Case	Logic Fault Injection Test Case 1	Logic Fault Injection Test Case 2	Logic Fault Injection Test Case 3	PO-2-wire
	Input 1	Input 2	Input 3	Input 4	Input 5	Input 6	Input 7	Output	Output	Output	Output	
34	0	1	0	0	0	1	0	0	0	0	1	
35	1	0	0	0	0	1	0	0	0	0	0	
36	1	1	0	0	0	1	0	0	0	1	0	
37	0	0	1	0	0	1	0	0	0	0	1	
38	0	1	1	0	0	1	0	0	0	1	1	
39	1	0	1	0	0	1	0	0	0	1	0	
40	1	1	1	0	0	1	0	0	0	1	1	
41	0	0	0	1	0	1	0	0	0	0	1	
42	0	1	0	1	0	1	0	0	0	1	1	
43	1	0	0	1	0	1	0	0	0	1	0	
44	1	1	0	1	0	1	0	0	0	1	1	44-2,3
45	0	0	1	1	0	1	0	0	0	1	1	
46	0	1	1	1	0	1	0	0	0	1	1	
47	1	0	1	1	0	1	0	0	0	1	1	
48	1	1	1	1	1	1	0	1	1	1	1	
49	0	0	0	0	1	1	0	0	0	0	0	
50	0	1	0	0	1	1	0	0	1	0	1	
51	1	0	0	0	1	1	0	0	0	0	0	
52	1	1	0	0	1	1	0	1	0	1	0	
53	0	0	1	0	1	1	0	0	1	0	1	
54	0	1	1	0	1	1	0	1	1	1	1	
55	1	0	1	0	1	1	0	1	0	1	0	
56	1	1	1	0	1	1	0	1	1	1	1	

Test No.								Basic Case	Logic Fault Injection Test Case 1	Logic Fault Injection Test Case 2	Logic Fault Injection Test Case 3	PO-2-wire
	Input 1	Input 2	Input 3	Input 4	Input 5	Input 6	Input 7	Output	Output	Output	Output	
57	0	0	0	1	1	1	0	0	1	0	1	
58	0	1	0	1	1	1	0	1	1	1	1	
59	1	0	0	1	1	1	0	1	0	1	0	
60	1	1	0	1	1	1	0	1	1	1	1	
61	0	0	1	1	1	1	0	1	1	1	1	
62	0	1	1	1	1	1	0	1	1	1	1	
63	1	0	1	1	1	1	0	1	1	1	1	
64	1	1	1	1	1	1	0	1	1	1	1	
65	0	0	0	0	0	0	1	0	0	0	0	
66	0	1	0	0	0	0	1	0	1	0	1	
67	1	0	0	0	0	0	1	0	0	0	0	
68	1	1	0	0	0	0	1	1	0	1	0	
69	0	0	1	0	0	0	1	0	1	0	1	
70	0	1	1	0	0	0	1	1	1	1	1	
71	1	0	1	0	0	0	1	1	0	1	0	
72	1	1	1	0	0	0	1	1	1	1	1	
73	0	0	0	1	0	0	1	0	1	0	1	
74	0	1	0	1	0	0	1	1	1	1	1	
75	1	0	0	1	0	0	1	1	0	1	0	
76	1	1	0	1	0	0	1	1	1	1	1	
77	0	0	1	1	0	0	1	1	1	1	1	
78	0	1	1	1	0	0	1	1	1	1	1	
79	1	0	1	1	0	0	1	1	1	1	1	

Test No.	Input 1	Input 2	Input 3	Input 4	Input 5	Input 6	Input 7	Base Case Output	Logic Fault Injection Test Case 1 Output	Logic Fault Injection Test Case 2 Output	Logic Fault Injection Test Case 3 Output	PO-2-wire
80	1	1	1	1	1	0	1	1	1	1	1	
81	0	0	0	0	1	0	1	0	0	0	0	
82	0	1	0	0	1	0	1	0	1	0	1	
83	1	0	0	0	1	0	1	0	0	0	0	
84	1	1	0	0	1	0	1	1	0	1	0	
85	0	0	1	0	1	0	1	0	1	0	1	
86	0	1	1	0	1	0	1	1	1	1	1	
87	1	0	1	0	1	0	1	1	0	1	0	87-1,3
88	1	1	1	0	1	0	1	1	1	1	1	
89	0	0	0	1	1	0	1	0	1	0	1	
90	0	1	0	1	1	0	1	1	1	1	1	
91	1	0	0	1	1	0	1	1	0	1	0	
92	1	1	0	1	1	0	1	1	1	1	1	
93	0	0	1	1	1	0	1	1	1	1	1	
94	0	1	1	1	1	0	1	1	1	1	1	
95	1	0	1	1	1	0	1	1	1	1	1	
96	1	1	1	1	1	0	1	1	1	1	1	
97	0	0	0	0	0	1	1	0	0	0	0	97
98	0	1	0	0	0	1	1	0	1	0	1	
99	1	0	0	0	0	1	1	0	0	0	0	
100	1	1	0	0	0	1	1	1	0	1	0	
101	0	0	1	0	0	1	1	0	1	0	1	
102	0	1	1	0	0	1	1	1	1	1	1	

Test No.	Input 1	Input 2	Input 3	Input 4	Input 5	Input 6	Input 7	Base Case Output	Logic Fault Injection Test Case 1 Output	Logic Fault Injection Test Case 2 Output	Logic Fault Injection Test Case 3 Output	PO-2-wire
103	1	0	1	0	0	1	1	1	0	1	0	
104	1	1	1	0	0	1	1	1	1	1	1	
105	0	0	0	1	0	1	1	0	1	0	1	
106	0	1	0	1	0	1	1	1	1	1	1	
107	1	0	0	1	0	1	1	1	0	1	0	
108	1	1	0	1	0	1	1	1	1	1	1	
109	0	0	1	1	0	1	1	1	1	1	1	
110	0	1	1	1	0	1	1	1	1	1	1	
111	1	0	1	1	0	1	1	1	1	1	1	
112	1	1	1	1	1	1	1	1	1	1	1	
113	0	0	0	0	1	1	1	0	0	0	0	
114	0	1	0	0	1	1	1	0	1	0	1	
115	1	0	0	0	1	1	1	0	0	0	0	
116	1	1	0	0	1	1	1	1	0	1	0	
117	0	0	1	0	1	1	1	0	1	0	1	
118	0	1	1	0	1	1	1	1	1	1	1	
119	1	0	1	0	1	1	1	1	0	1	0	
120	1	1	1	0	1	1	1	1	1	1	1	
121	0	0	0	1	1	1	1	0	1	0	1	
122	0	1	0	1	1	1	1	1	1	1	1	
123	1	0	0	1	1	1	1	1	0	1	0	
124	1	1	0	1	1	1	1	1	1	1	1	
125	0	0	1	1	1	1	1	1	1	1	1	

Test No.	Input 1	Input 2	Input 3	Input 4	Input 5	Input 6	Input 7	Base Case Output	Logic Fault Injection Test Case 1 Output	Logic Fault Injection Test Case 2 Output	Logic Fault Injection Test Case 3 Output	PO-2-wire
126	0	1	1	1	1	1	1	1	1	1	1	126
127	1	0	1	1	1	1	1	1	1	1	1	
128	1	0	1	1	1	1	1	1	1	1	1	

The OA test suites and test results

Test No.	Input							Output	Output		
	1	2	3	4	5	6	7	Basis Case	Case 1	Case 2	Case 3
1	0	0	0	0	0	0	0	0	0	0	0
2	0	1	1	1	1	0	0	0	0	***1 (logic fault is detected)***	***1 (logic fault is detected)***
3	1	0	1	1	0	1	0	0	0	***1 (logic fault is detected)***	***1 (logic fault is detected)***
4	1	1	0	0	1	1	0	1	***0 (logic fault is detected)***	1	***0 (logic fault is detected)***
5	1	1	0	1	0	0	1	1	1	1	1
6	1	0	1	0	1	0	1	1	***0 (logic fault is detected)***	1	***0 (logic fault is detected)***
7	0	1	1	0	0	1	1	1	1	1	1
8	0	0	0	1	1	1	1	0	***1 (logic fault is detected)***	0	***1 (logic fault is detected)***

The test suites generated by the IPO have 7 test cases, which also can detect all of three injecting logical faults

The OA test suites and test results

Test No.	Input							Output	Output		
	1	2	3	4	5	6	7	Basis Case	Case 1	Case 2	Case 3
1	0	0	0	0	0	0	0	0	0	0	0
2	0	1	1	0	0	0	0	0	0	0	0
3	0	0	0	1	1	0	0	0	0	0	***1 (logic fault is detected)***
4	1	1	0	1	0	1	0	0	0	***1 (logic fault is detected)***	***1 (logic fault is detected)***
5	1	0	1	0	1	0	1	1	***0 (logic fault is detected)***	1	***0 (logic fault is detected)***
6	0	0	0	0	0	1	1	0	0	0	0
7	0	1	1	1	1	1	1	1	1	1	1

The test suites caused by the IPO and the output among basis case and cases injecting logical fault

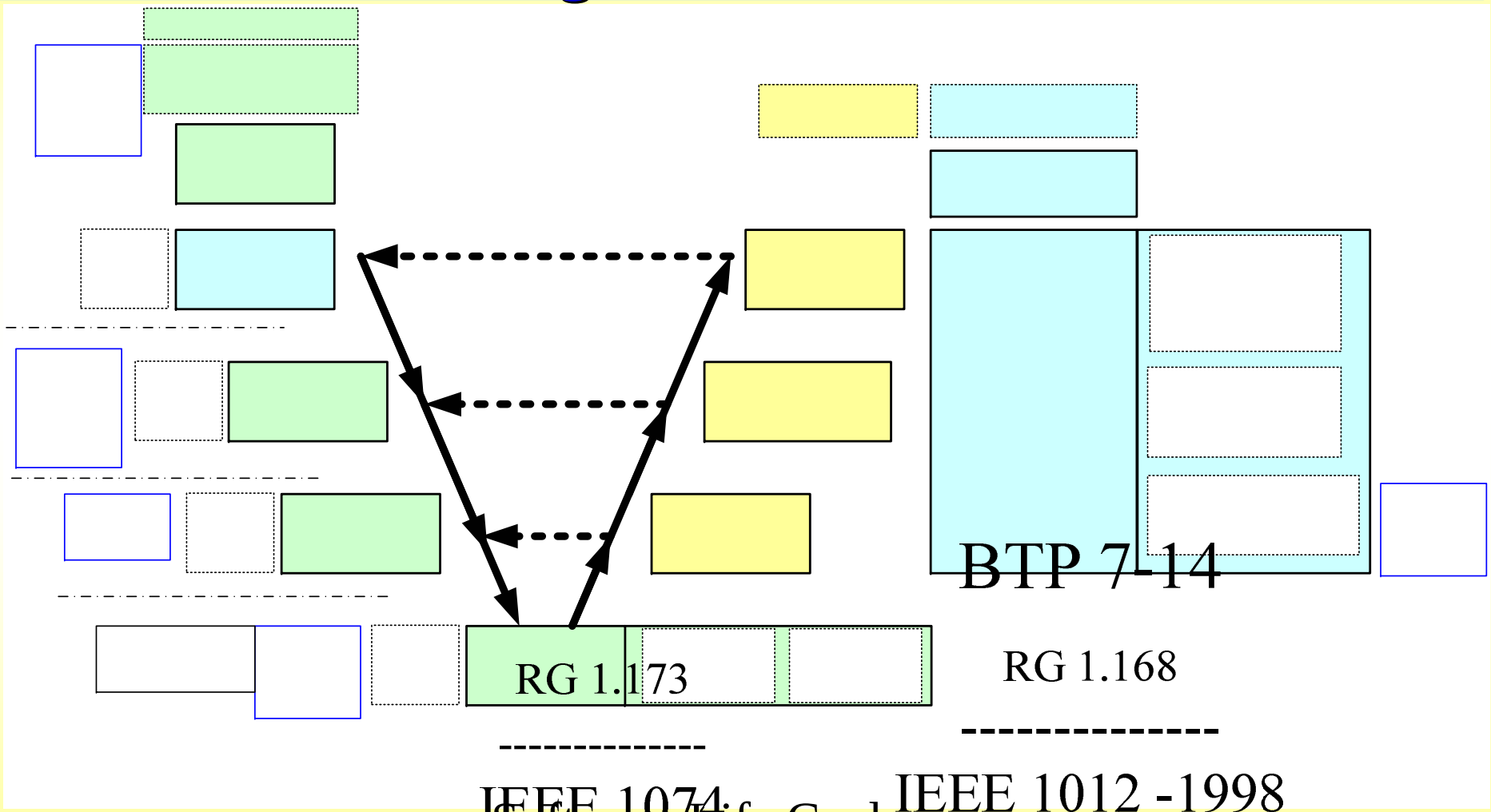
Study on Digital I&C software test coverage theory (cont.)

- Discussion
  - A number of random selections of 7-test-case suites were also performed
  - The result shows OA and IPO have better capability, than random selection, to detect the software implementation faults in a logic diagram

Study on Digital I&C software test coverage theory (cont.)

- Recommendation for Integration phase test matrix
  - Build up the automatic testing tool and verification tool
  - Perform full test if the test combination scale is small (e.g., less than 1,000 combinations);
  - Otherwise use the test strategies (e.g., OA or IPO) and random input tests
- The tests in Integration phase is not able to identify some of the design errors in Concept phase and Requirement phase.
- These faults should be avoided by review and validated by FAT and SAT.

Study on software development and test related regulations and standards



BTP-14: Guidance on Software Reviews for Digital Computer-Based Instrumentation and Control Systems

IEEE 1012: Standard for Software Verification and Validation

Study on software development and test related regulations and standards (cont.)

- Life Cycle
  - RG 1.173 Developing Software Life Cycle Processes for Digital Computer Software Used in Safety Systems of Nuclear Power Plants, 09/1997
  - IEEE Std 1074-2006, "IEEE Standard for Developing Software Life Cycle Processes"
- Requirement
  - RG 1.172 Software Requirements Specifications for Digital Computer Software Used in Safety Systems of Nuclear Power Plants, 09/1997
  - IEEE STD 830-2009, " IEEE Recommended Practice for Software Requirements Specifications"
- Design
  - IEEE STD 1016-2009, "IEEE Standard for Information Technology—Systems Design—Software Design Descriptions"

Study on software development and test related regulations and standards (cont.)

- Test
  - RG 1.171 Software Unit Testing for Digital Computer Software Used in Safety Systems of Nuclear Power Plants, 09/1997
  - NUREG/ CR 6463, "Review Guidelines for Software Languages for Use in Nuclear Power Plant Safety Systems"
  - RG 1.170 Software Test Documentation for Digital Computer Software Used in Safety Systems of Nuclear Power Plants, 09/1997
  - IEEE STD 829-2008, "IEEE Standard for Software Test Documentation"

- The study is still ongoing

Institute of Nuclear Energy Research