
Suresoft

안전등급 계측제어계통 MMI 소프트웨어의 코드 커버리지 측정 방법

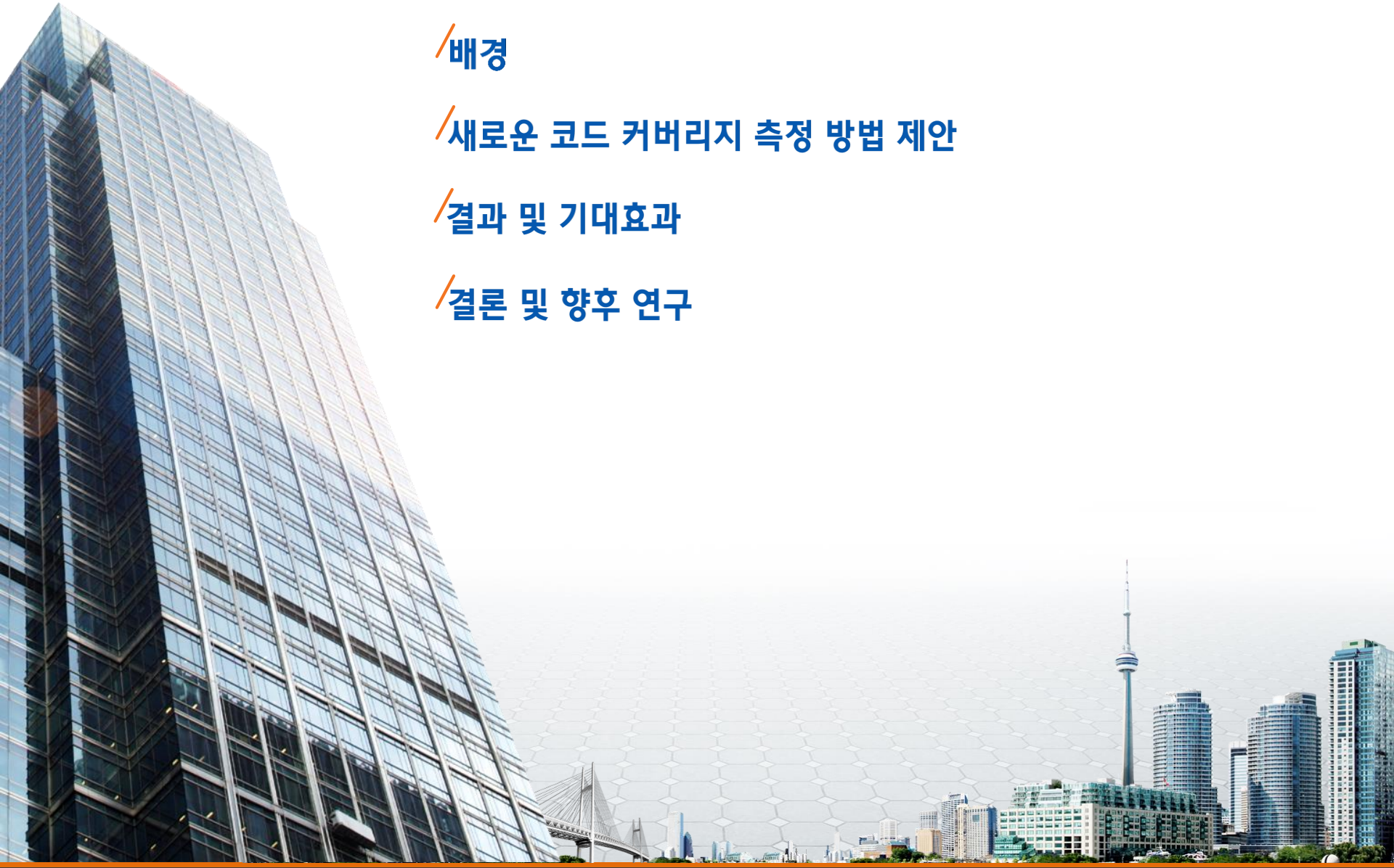
Introduction of Code Coverage Measurement of MMI Software for Safety I&C Systems

2016

슈어소프트테크㈜
이은형

목차

Suresoft



- 배경
- 새로운 코드 커버리지 측정 방법 제안
- 결과 및 기대효과
- 결론 및 향후 연구

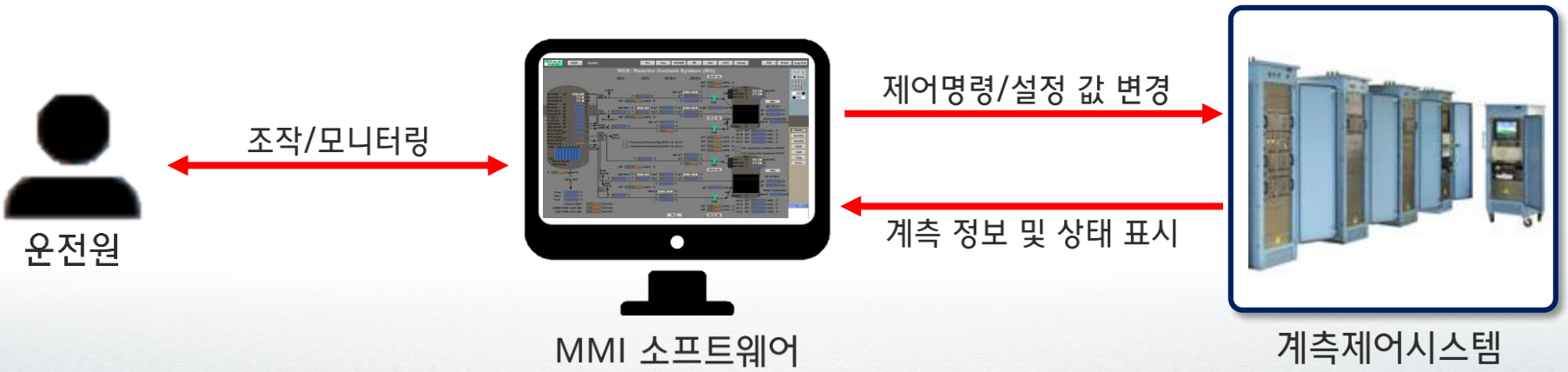
배경

배경

□ MMI[1] 소프트웨어 커버리지 측정 배경

- 소프트웨어 신뢰성 인식 제고 (계측제어 시스템과의 연계 기능-데이터 공유/변경)

[MMI 소프트웨어 동작 시나리오]



1) MMI (Man-Machine Interface): 사람과 시스템간의 연계 수단

배경

Suresoft

□ MMI[1] 소프트웨어 커버리지 측정 배경

- 규제기관의 인허가 기준 강화에 따른 코드 커버리지[2] 측정 요구
- 관련 표준 및 권고: IEC 60880, US NRC RG 1.171, KINS/RG-N08.18

E.2 Systematic approaches

Each module should be verified and tested in a systematic way according to objectives and their associated coverage criteria. In addition to manual verification and testing activities, automated verifier and testing tools should be used as much as possible. Test results should be checked with expected results derived from the program module specifications. Input to and output from the module should be handled in the same manner as in the safety system.

IEC 60880

Annex E. Software Verification & Testing

2. Test Program

The coverage of requirements and the internal structure of the code are two particularly important aspects of test coverage necessary for the unit testing of safety system software, as follows:

US NRC RG 1.171

Section 2.b. Coverage of Module Structure

나. 내부구조 커버리지(Coverage of Internal Structure)

전력산업기술기준 EMB-3500, 3.1.2(2)절은 소프트웨어 단위시험 활동의 완전성을 검사하기 위한 기준으로서 문장 커버리지(시험사례가 각 소스코드 문장을 포함하는)를 규정하고 있다. 문장 커버리지는 시험 완전성을 측정하기에는 불충분한 기준이다(NUREG/CR-6263). 그러므로 소프트웨어 단위시험을 위한 충분한 기준으로 문장 커버리지는 인정되지 않고 있다. 안전계통 소프트웨어에서 채택된 단위시험 커버리지 기준은 확인되고 정당화되어야 한다.

KINS/RG-N08.18

8.18절. 안전계통 소프트웨어의 단위시험

1) MMI (Man-Machine Interface): 사람과 시스템간의 연계 수단
2) 코드 커버리지: 소프트웨어 시험의 충분성을 나타내는 지표

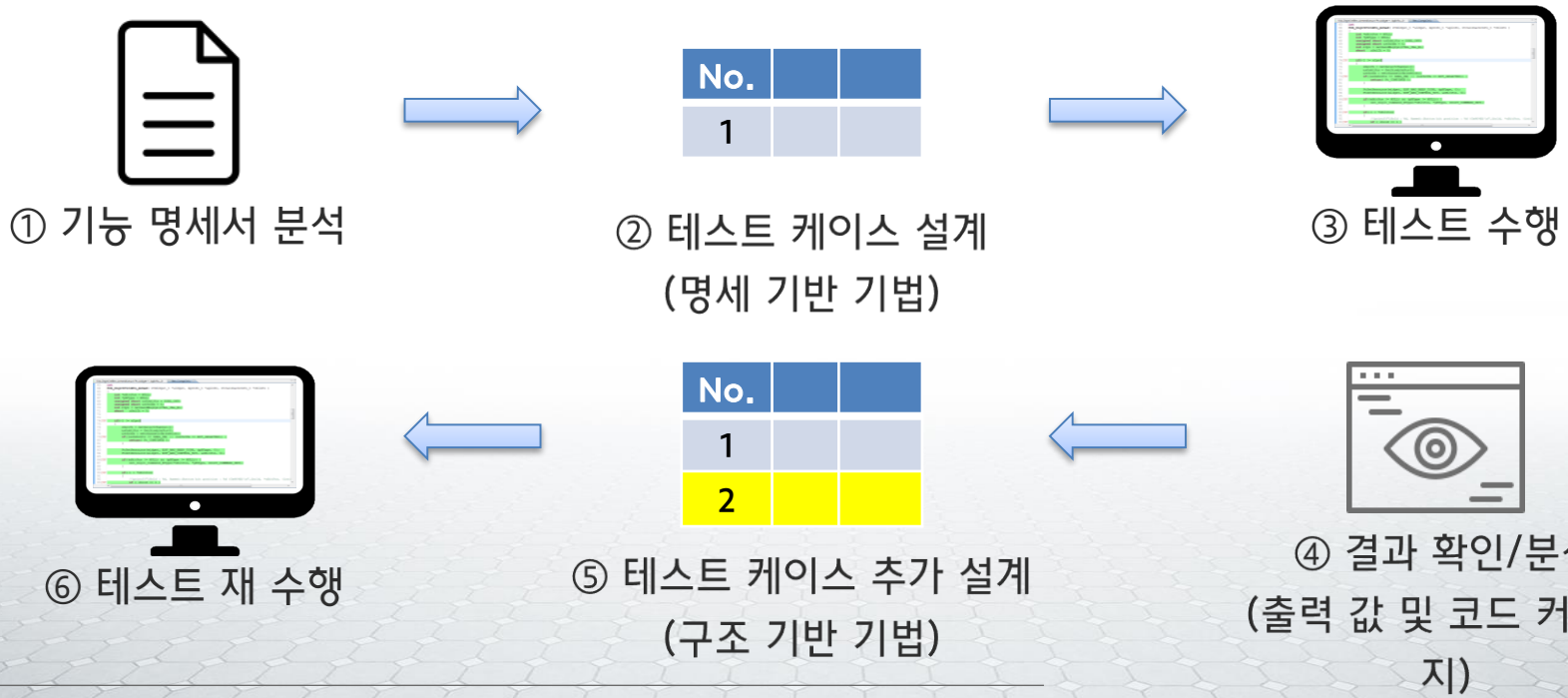
배경

Suresoft

□ 기존 MMI 소프트웨어 커버리지 측정 방법

- 시험환경: 대상 시스템 환경 + 시험 자동화 도구 (with CT[1])
- 테스트 시퀀스[2] 에 따라 수행

[MMI 소프트웨어 커버리지 측정 절차]



1) CT (Controller Tester): Suresofttech에서 개발한 동적시험 자동화 도구
2) 테스트시퀀스: ① → ② → ③ → ④ → ⑤ → ⑥ (목표 커버리지 달성 시까지 ④ → ⑤ → ⑥ 반복)

배경

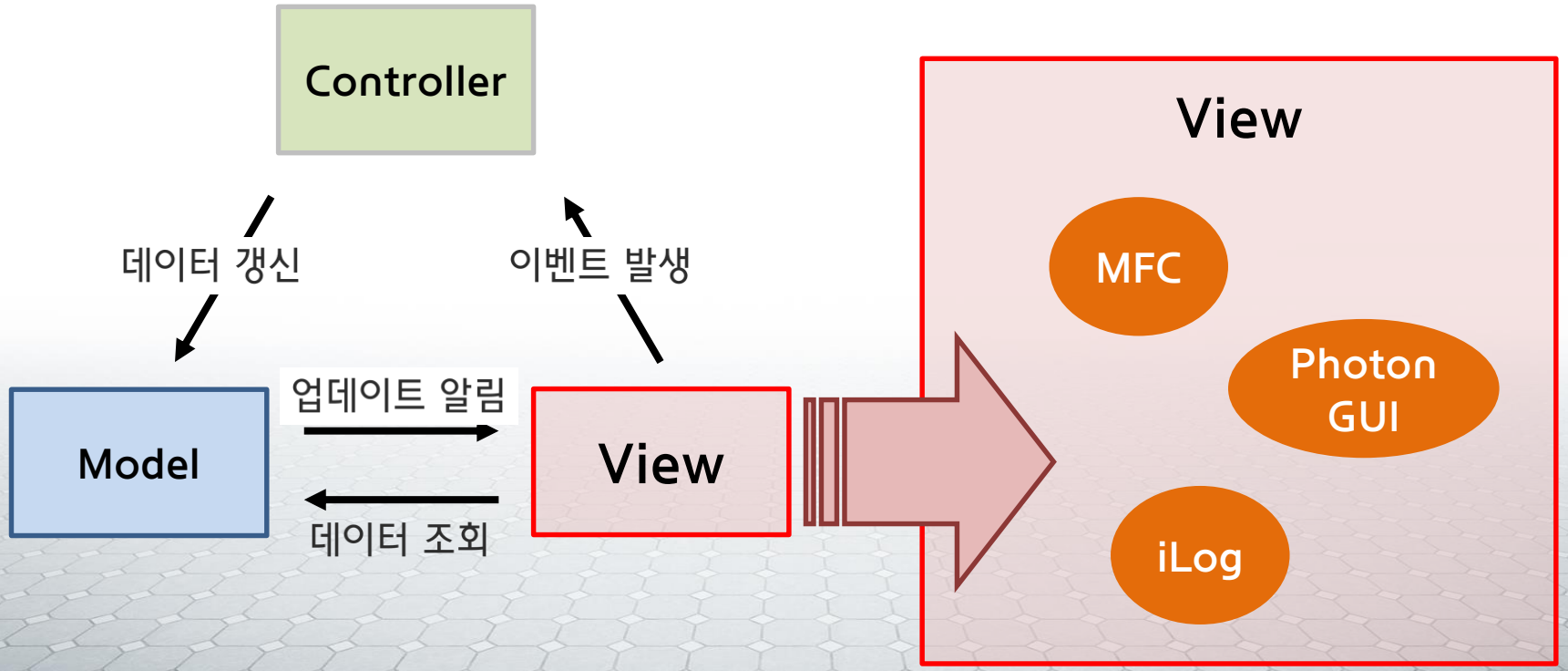
Suresoft

□ 기존 MMI 소프트웨어 커버리지 측정 방법의 문제점

- 방대한 소스코드, 높은 복잡도 ➔ 테스트비용 증가
- 다양한 프레임워크의 사용 (MFC, Photon Library, iLog)

➔ 환경구축 및 라이브러리 접근이 어려움

[MMI 소프트웨어 구조 (MVC 모델기반)]



새로운 코드 커버리지 측정 방법 제안

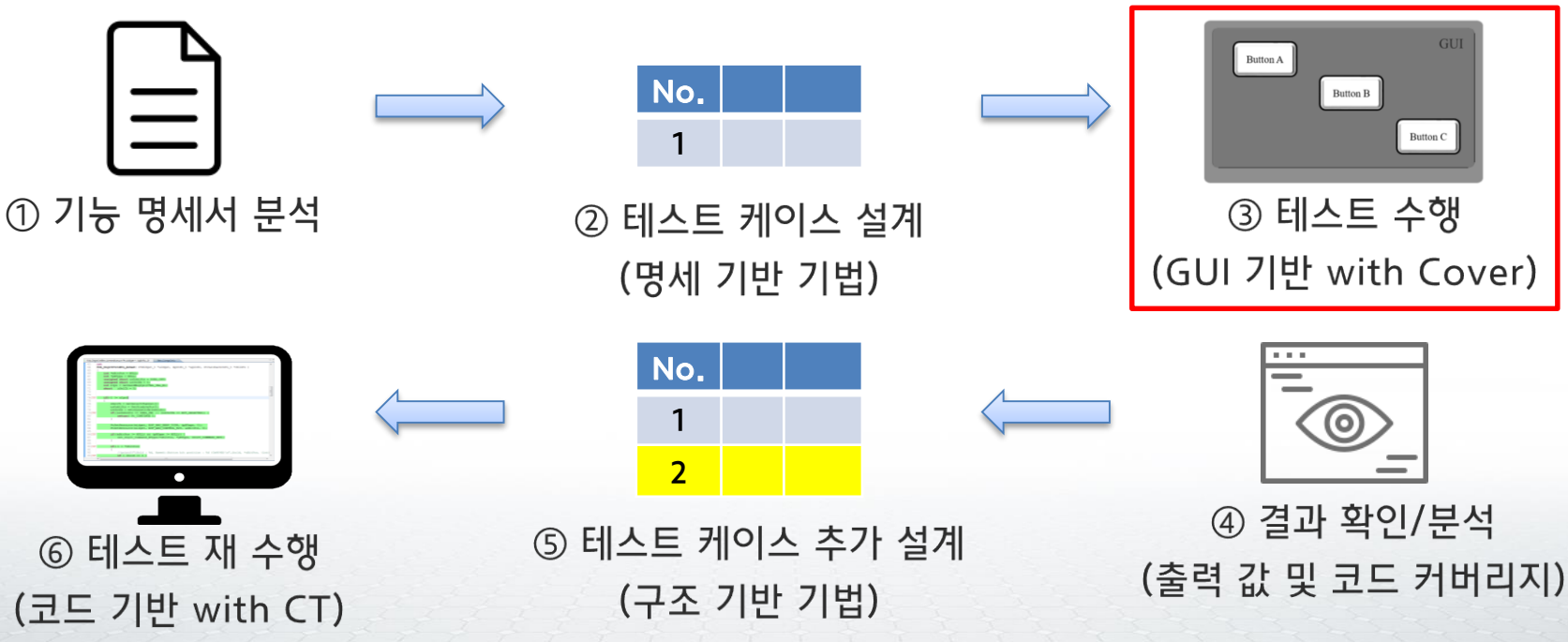
새로운 코드 커버리지 측정 방법 제안

Suresoft

□ 새로운 MMI 소프트웨어 커버리지 측정 방법

- 시험환경: 대상 시스템 환경 + 시험 자동화 도구 (with Cover[1] + CT[2])
- 테스트 시퀀스[3] 에 따라 수행

[새로운 MMI 소프트웨어 커버리지 측정 절차]



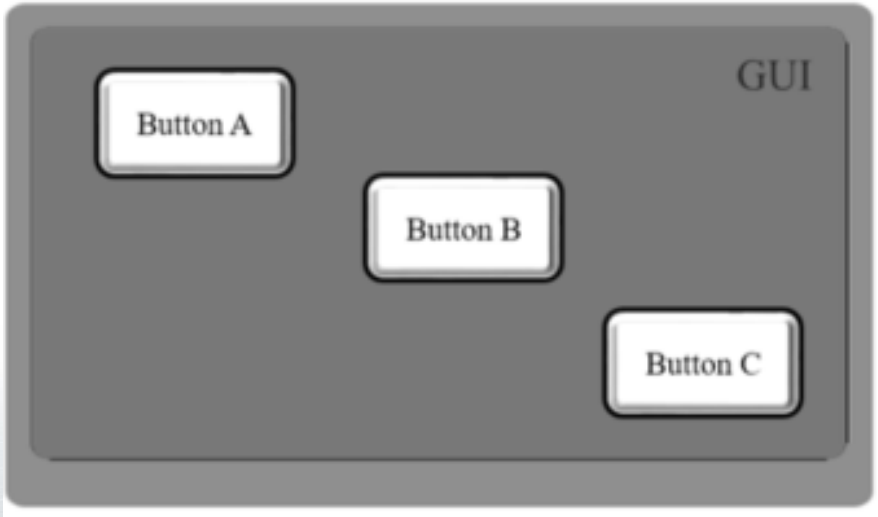
1) Cover: Suresofttech에서 개발한 GUI 기반 코드 커버리지 측정 도구
2) CT (Controller Tester): Suresofttech에서 개발한 동적시험 자동화 도구
3) 테스트시퀀스: ① → ② → ③ → ④ → ⑤ → ⑥ → ④ → ⑤ → ⑥ (④ → ⑤ → ⑥ 반복)

새로운 코드 커버리지 측정 방법 제안

Suresoft

□ COVER를 활용한 GUI 테스트 수행

- **GUI 테스트 수행 시 코드 커버리지 달성 가능**
- GUI 테스트로 도달이 어려운 코드는 기존 방식 사용



Button	Related Function
A	a1, a2
B	b1, b2, b3
C	c1, c2, c3

[COVER를 활용한 GUI 테스트 예시]

새로운 코드 커버리지 측정 방법 제안

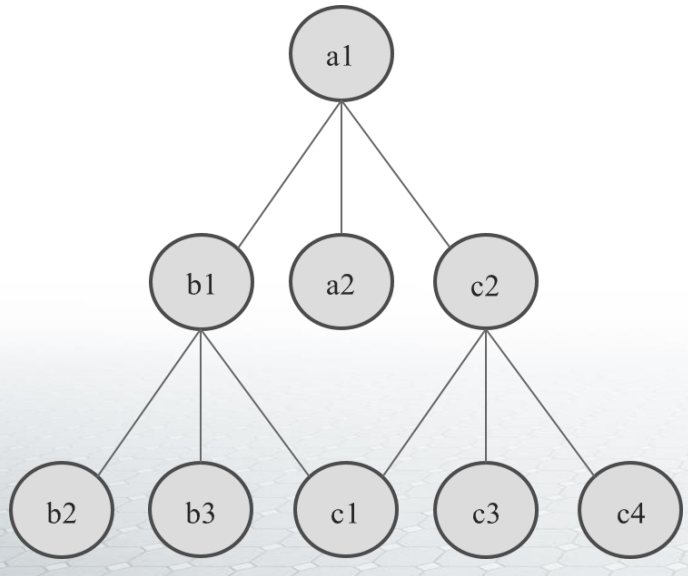
Suresoft

□ COVER를 활용한 GUI 테스트 수행

- **GUI 테스트 수행 시 코드 커버리지 달성 가능**
- GUI 테스트로 도달이 어려운 코드는 기존 방식 사용

☒ GUI 기반 시험 수행 전

코드 커버리지: 0 %



Test Case	Cover Function	Coverage
A Button Click	a1, a2	22.2%
B Button Click	b1, b2, b3	30.0%
A → B Button Click	a1, a2, b1, b2, b3	55.5%
A → B → C Button Click	a1, a2, b1, b2, b3, c1, c2, c3	88.8%

[COVER를 활용한 GUI 테스트 예시]

새로운 코드 커버리지 측정 방법 제안

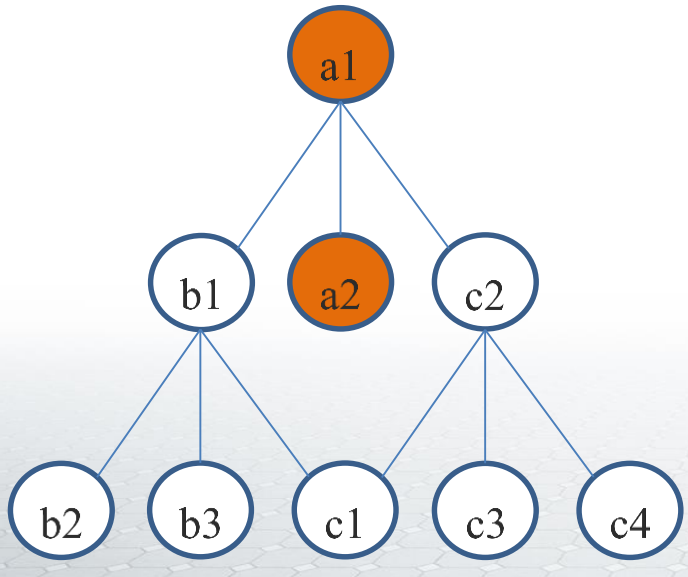
Suresoft

□ COVER를 활용한 GUI 테스트 수행

- **GUI 테스트 수행 시 코드 커버리지 달성 가능**
- GUI 테스트로 도달이 어려운 코드는 기존 방식 사용

☒ GUI 기반 시험 수행 전

코드 커버리지: 22.2 %



Test Case	Cover Function	Coverage
A Button Click	a1, a2	22.2%
B Button Click	b1, b2, b3	30.0%
A → B Button Click	a1, a2, b1, b2, b3	55.5%
A → B → C Button Click	a1, a2, b1, b2, b3, c1, c2, c3	88.8%

[COVER를 활용한 GUI 테스트 예시]

새로운 코드 커버리지 측정 방법 제안

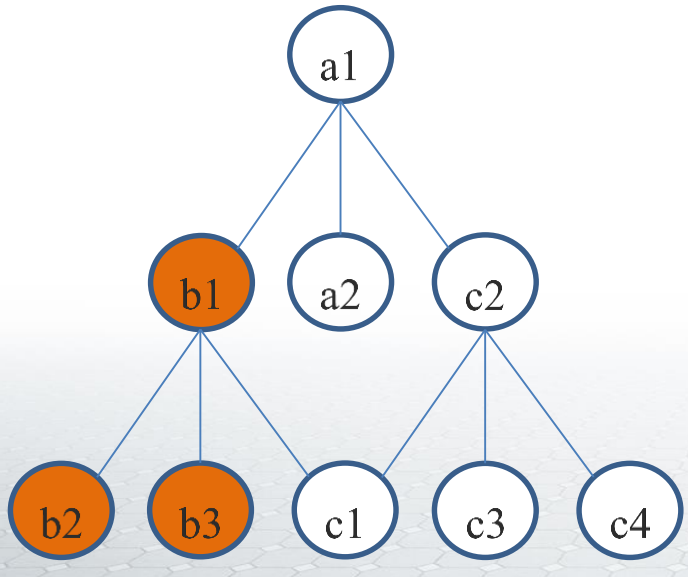
Suresoft

□ COVER를 활용한 GUI 테스트 수행

- **GUI 테스트 수행 시 코드 커버리지 달성 가능**
- GUI 테스트로 도달이 어려운 코드는 기존 방식 사용

☒ GUI 기반 시험 수행 전

코드 커버리지: 30.0 %



Test Case	Cover Function	Coverage
A Button Click	a1, a2	22.2%
B Button Click	b1, b2, b3	30.0%
A → B Button Click	a1, a2, b1, b2, b3	55.5%
A → B → C Button Click	a1, a2, b1, b2, b3, c1, c2, c3	88.8%

[COVER를 활용한 GUI 테스트 예시]

새로운 코드 커버리지 측정 방법 제안

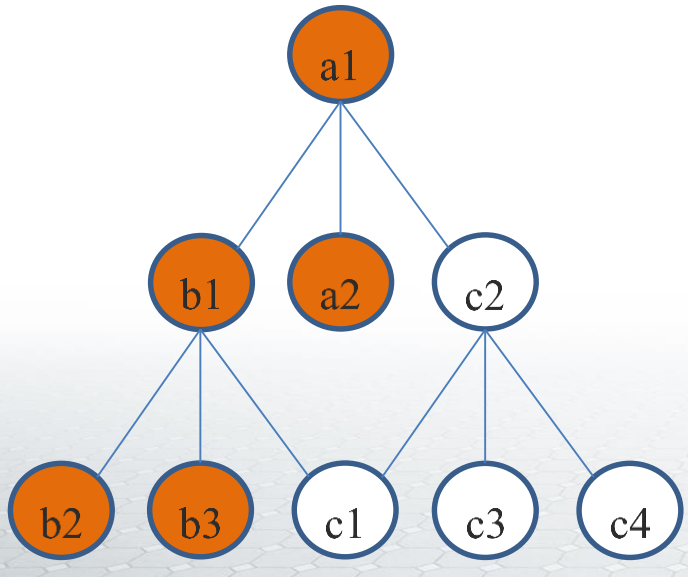
Suresoft

□ COVER를 활용한 GUI 테스트 수행

- **GUI 테스트 수행 시 코드 커버리지 달성 가능**
- GUI 테스트로 도달이 어려운 코드는 기존 방식 사용

☒ GUI 기반 시험 수행 전

코드 커버리지: 55.5 %



Test Case	Cover Function	Coverage
A Button Click	a1, a2	22.2%
B Button Click	b1, b2, b3	30.0%
A → B Button Click	a1, a2, b1, b2, b3	55.5%
A → B → C Button Click	a1, a2, b1, b2, b3, c1, c2, c3	88.8%

[COVER를 활용한 GUI 테스트 예시]

새로운 코드 커버리지 측정 방법 제안

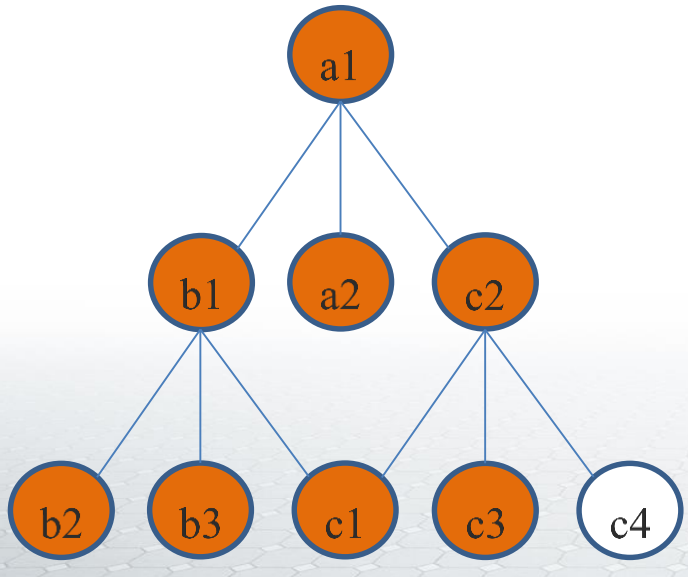
Suresoft

□ COVER를 활용한 GUI 테스트 수행

- GUI 테스트 수행 시 코드 커버리지 달성 가능
- GUI 테스트로 도달이 어려운 코드는 기존 방식 사용

☒ GUI 기반 시험 수행 전

코드 커버리지: 88.8 %



Test Case	Cover Function	Coverage
A Button Click	a1, a2	22.2%
B Button Click	b1, b2, b3	30.0%
A → B Button Click	a1, a2, b1, b2, b3	55.5%
A → B → C Button Click	a1, a2, b1, b2, b3, c1, c2, c3	88.8%

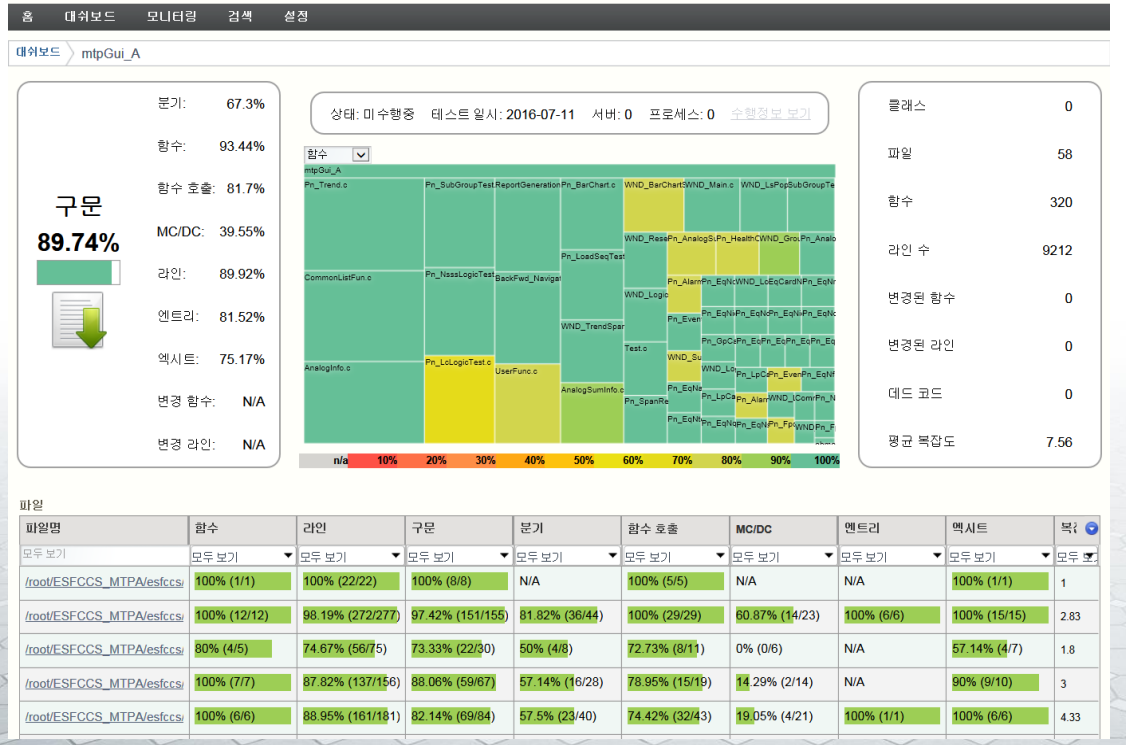
[COVER를 활용한 GUI 테스트 예시]

새로운 코드 커버리지 측정 방법 제안

Suresoft

□ COVER 소개

- GUI 실행을 통해 측정된 커버리지를 정량적인 수치로 표시하는 도구
- C/C++, JAVA 언어 사용 가능
- 구문 커버리지, 분기 커버리지 등 10가지의 코드 커버리지 측정 가능
- GUI 동작 시나리오 수행만으로 커버리지 달성 가능
- 변경된 함수, 라인에 대한 데이터 및 커버리지 확인 가능



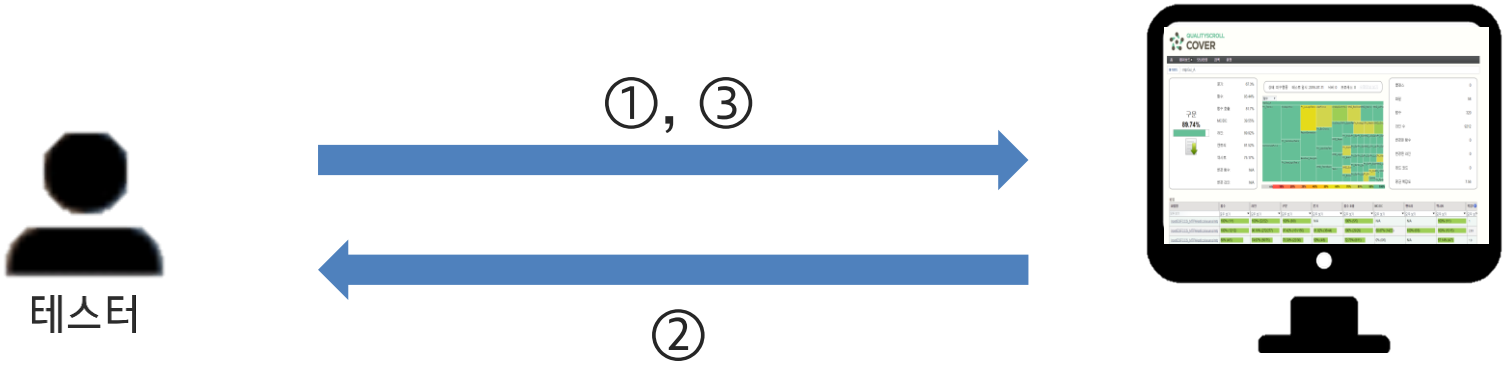
QUALITYSCROLL COVER

새로운 코드 커버리지 측정 방법 제안

Suresoft

□ COVER를 활용한 GUI 테스트 수행

- GUI 테스트 수행 시 코드 커버리지 달성 가능
- GUI 테스트로 도달이 어려운 코드는 기존 방식 사용



① 환경 구축 및 UI Test 수행

② 결과 확인

③ 결과 분석 및 UI Test Case 추가

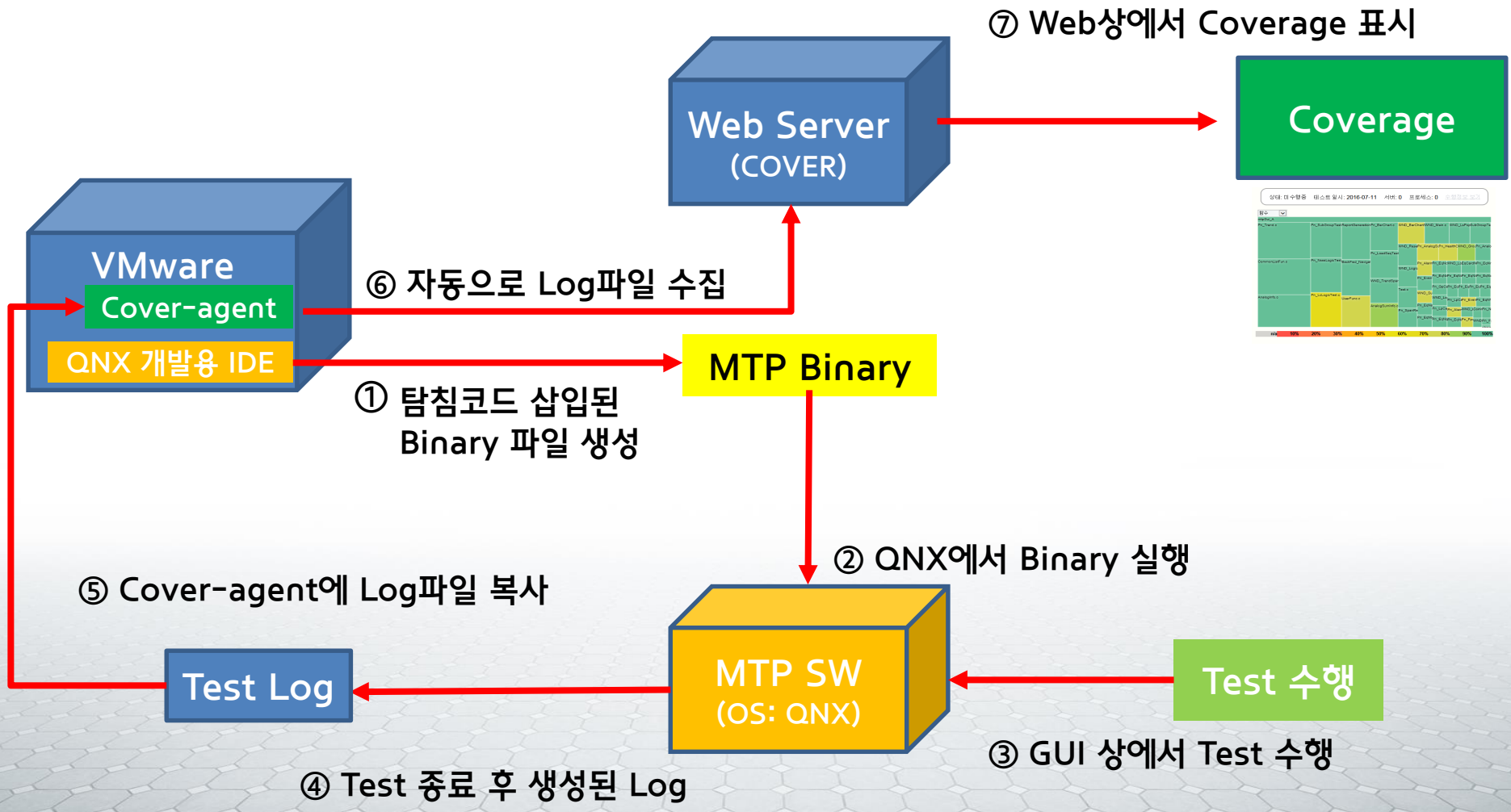
※ ① → ② → ③ 반복 수행을 통해 목표 커버리지 달성

결과 및 기대효과

결과 및 기대효과

Suresoft

[COVER 실제 적용 환경]



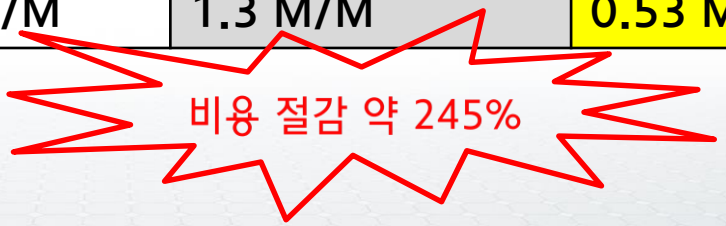
결과 및 기대효과

Suresoft

□ 변경 방식 적용 결과

- 기존 방식 대비 약 245%의 비용 절감 효과 발생
- GUI 테스트 커버리지 상승 시 추가 절감 효과 발생 가능

		기존 방식	새로운 방식
시험 한 전체 라인 수		9212	9212
1달 간 시험 가능한 라인 수		7000	68694
Coverage 구분	시범 적용 사례	Branch : 100% (CT : 100%)	Branch : 100% (COVER : 67.3%, CT : 32.7%)
Coverage 100% 달성을 위한 M/M		1.3 M/M	0.53 M/M



*) COVER로 9212Line 실제 시험일 : 2일 (67.3% 커버리지 달성)

1M/M 기준 시험 가능 라인 수 : COVER → 9212*0.673/2*20 = 62000

CT → 9212*0.327/9*20 = 6694

결론

결론 및 향후 연구

Suresoft

결론

계측제어 시스템에 적용되는 MMI 소프트웨어의 기존 커버리지 측정방법은 효율이 낮음.

새로운 커버리지 측정 방법이 이를 충분히 보완할 수 있으며, 해당 방법 적용을 통해 많은 경제적 이익을 얻을 수 있음.

향후 연구

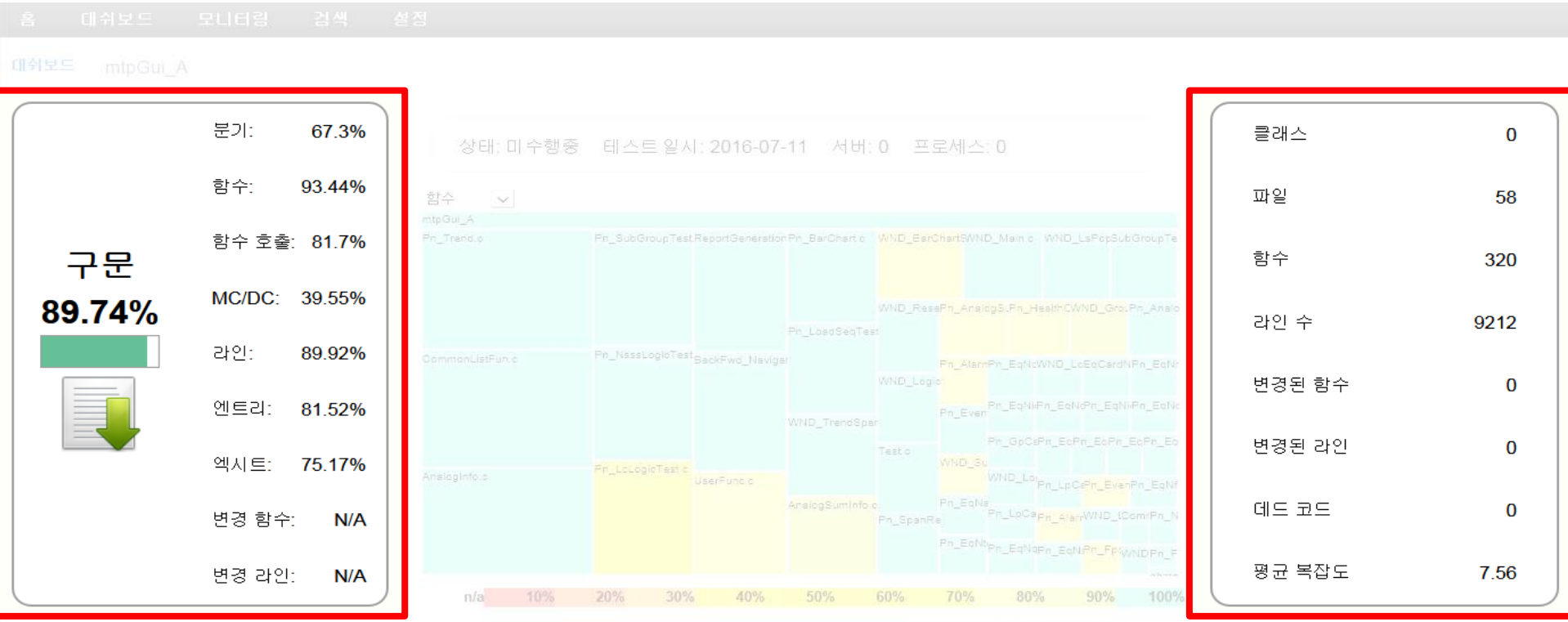
향후 새로운 커버리지 측정방법에 대한 적용 사례를 보다 많이 확보하여 해당 방법에 대한 신뢰성을 확보하고 이 과정에서 발생되는 문제점을 분석하고 이를 보완하기 위한 추가 연구 예정.

Thank you for your kind attention

www.suresofttech.com www.codescroll.com

부록

➤ COVER 화면 소개 1



각종 Coverage 요약

테스트를 수행한 파일에 대한 각종 정보 표시

소스코드의 정량적인 성숙도를 다양하게 나타냄

Report 내보내기 기능

부록

➤ COVER 화면 소개 2

각 파일에 대한 Code Coverage 표시

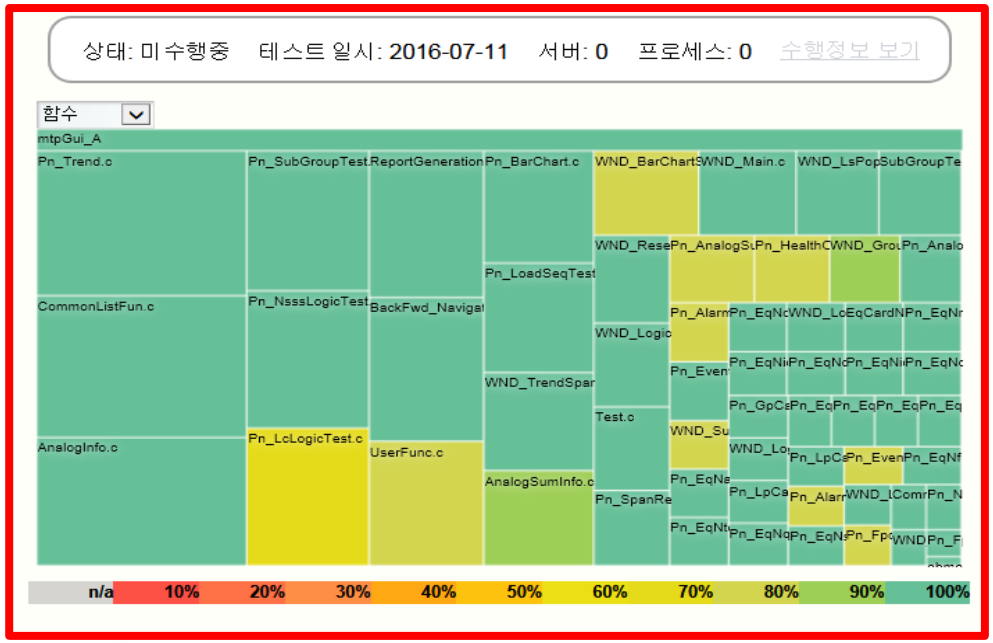
특정 파일을 선택 시 각 함수에 대한 Code Coverage 표시

파일

파일명	함수	라인	구문	분기	함수 호출	MC/DC	엔트리	엑시트	복
모두 보기	모두 보기	모두 보기	모두 보기	모두 보기	모두 보기	모두 보기	모두 보기	모두 보기	모두 보
/root/ESFCCS_MTPA/esfccs/	100% (1/1)	100% (22/22)	100% (8/8)	N/A	100% (5/5)	N/A	N/A	100% (1/1)	1
/root/ESFCCS_MTPA/esfccs/	100% (12/12)	98.19% (272/277)	97.42% (151/155)	81.82% (36/44)	100% (29/29)	60.87% (14/23)	100% (6/6)	100% (15/15)	2.83
/root/ESFCCS_MTPA/esfccs/	80% (4/5)	74.67% (56/75)	73.33% (22/30)	50% (4/8)	72.73% (8/11)	0% (0/6)	N/A	57.14% (4/7)	1.8
/root/ESFCCS_MTPA/esfccs/	100% (7/7)	87.82% (137/156)	88.06% (59/67)	57.14% (16/28)	78.95% (15/19)	14.29% (2/14)	N/A	90% (9/10)	3
/root/ESFCCS_MTPA/esfccs/	100% (6/6)	88.95% (161/181)	82.14% (69/84)	57.5% (23/40)	74.42% (32/43)	19.05% (4/21)	100% (1/1)	100% (6/6)	4.33

부록

➤ COVER 화면 소개 3



시험한 파일에 대한 커버리지 달성도를 색으로 표현

각 파일 클릭 시 세부정보 표시

Suresoft

COVER 화면 소개 3-1

대쉬보드 > mtpGui_A > /root/ESFCCS_MTPA/esfccs/source/mtpGui_A/src/Pn_NsssLogicTest.c

구문
94.33%

분기:	81.71%
함수:	100%
함수 호출:	82.22%
MC/DC:	67.44%
라인:	95.53%
엔트리:	76.92%
엑시트:	100%
변경 함수:	N/A
변경 라인:	N/A

상태: 미수행중 테스트 일시: 2016-07-11 서버: 0 프로세스: 0 수행정보 보기

함수

함수명	라인	구문	분기	함수 호출	MC/DC	엔트리	엑시트
모두 보기	모두 보기	모두 보기	모두 보기	모두 보기	모두 보기	모두 보기	모두 보기
fcb_NsssLogicTes	87.5% (2	92.31% (	75% (3/4	66.67% (	50% (1/2	N/A	100% (1/
fcb_NsssLogicTes	96.08% (	94.12% (	75% (9/1	83.33% (	50% (3/6	N/A	100% (1/
fcb_NsssLogicTes	100% (32	100% (15	87.5% (7.	N/A	75% (3/4	50% (1/2	100% (1/
fcb_NsssLogicTes	100% (55	100% (29	90% (9/1	100% (3/	80% (4/5	N/A	100% (1/
fcb_LogicTestRes	100% (20	100% (10	100% (4/	100% (3/	100% (2/	100% (1/	100% (1/

UTF-8 저장

```
125  int
126  fcb_NsssLogicTest_Realized( PtWidget_t *widget, ApInfo_t *apinfo,
     PtCallbackInfo_t *cbinfo )
127  {
128      int i = 0;
129      int j = 0;
130      int nTempIpn = -1;
131      char szLgTestOutId[M_LOGIC_OUT][LEN_IDCODE] = {"OP_TEST_TYPE",
     "OP_TEST_TARGET_ID",
132                                                    "OP_TEST_FUNC_ID", "OP_TEST_START"};
133
134      usLgGcId = 1;
135      usLgFuncId = 1;
136      nTestState = 0;
137      SetTestEnd(nTestState);
138      SetTestEnableState(TEST_STOP);
139      SetTestType(LOGIC_TEST_TYPE);
140
141      SetWidgetArray();
142      ResultBtnBlock(usLgFuncId);
143      ChangeWidgetMode(TEST_MODE_RESET, FUNC_ALL);
144
145 T    if(GetSpResult() == result_true) {
146          for(i = 0; i < M_LOGIC_OUT; i++)
147          {
148              nLgOutIpn[i] = GetIpn(szLgTestOutId[i]);
149 F            if(nLgOutIpn[i] == -1) {
150                  printf_fpd_sts(FPD_SP, "GUI", "Failed get to ipn.
     idcode : %s", szLgTestOutId[i]);
151              } else {
152                  nTempIpn = nLgOutIpn[i];
153                  tPdbData[nTempIpn].value.i = 0;
154                  //printf("2. NsssLogicTest Init output value. id : %s,
     ipn : %d\n", szLgTestOutId[i], nTempIpn);
155
156              }
157          }
158          for(i = 0; i < M_TARGET_CNT; i++)
159          {
160              for(j = 0; j < M_RESULT_BTN; j++)
161              {
162                  nLgFnshIpn[i][j] = GetIpn(szLogicFnshId[i][j]);
163 F                if(nLgFnshIpn[i][j] == -1) {
164                      printf_fpd_sts(FPD_SP, "GUI", "Failed get to ipn.
     idcode : %s", szLogicFnshId[i][j]);
165                  }
```

파일 세부정보 화면