KSEJW 2010

대학생을 위한 Dependability교육
- SW 테스팅 -

건국대학교 컴퓨터공학부
유준범
jbyoo@konkuk.ac.kr

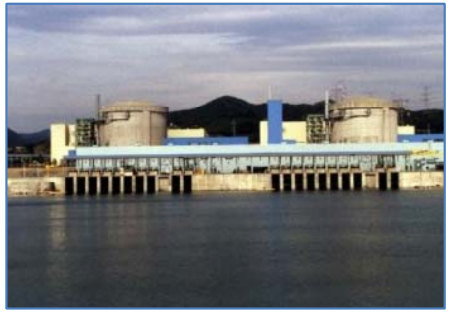
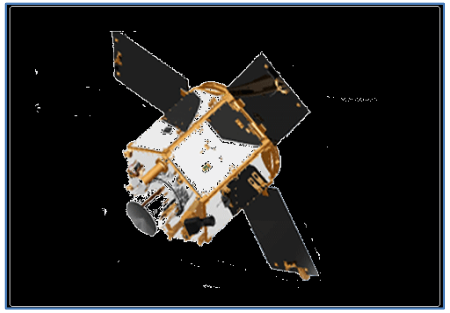
2010.08.19

Contents

- Motivation
  - Dependability
  - Software Engineering 관련 교과목 현황 (건국대학교 컴퓨터공학부)
- 2010 소프트웨어 검증 수업내용
  - 이론 수업
  - 실습 수업
- 수업 결과 분석
- 결론 및 향후 운영 방안

Motivation

- Dependability
  - Safety-critical system에 대한 관심 집중
    - 원자력발전소
    - 인공위성
    - 항공기 / 국방시스템
    - 고속철도
  - 소프트웨어 시스템의 안전성/정확성을 명명하는 척도
    - Dependable Software
    - Dependable System

Dependability를 위한 대학 교육

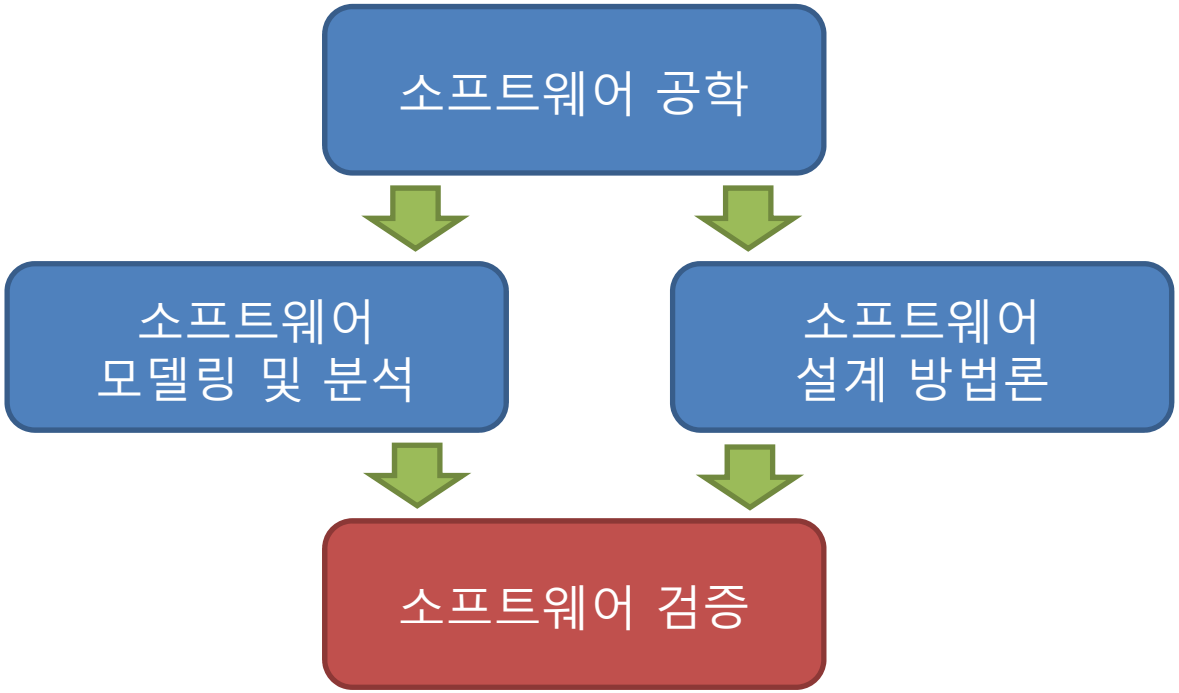
- Dependability 를 위한 연구 주제
  - 소프트웨어 V&V (Verification & Validation)
    - 정형 명세 및 검증 (Formal Specification & Verification)
    - 테스팅 (Testing)
    - 프로그램 정적분석 (Static Analysis)
  - 안전성분석 기법 (Safety Analysis)
  - 분야별로 특화된 개발 방법론

  - 대학원 수준의 특수/특화된 연구분야

- 학부생을 위한 Dependability 교과목 현황
  - **소프트웨어공학 개론**

학부생을 위한 Dependability 교과목

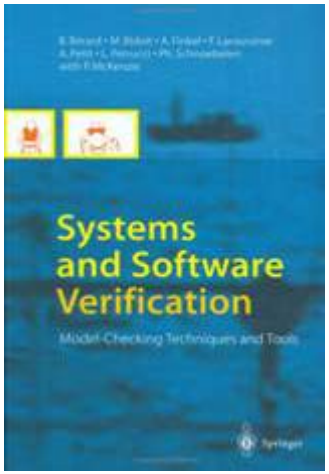
- 건국대학교 컴퓨터공학부 2010 기준, 소프트웨어공학 관련 교과목



- '소프트웨어 검증'

2009 소프트웨어 검증 수업

- 정형기법 개론
  (http://dslab.konkuk.ac.kr/Class/2009/09SV/09SV.htm)
  - 이론
    - 정형명세
    - 정형검증 (Model Checking)
  - 팀프로젝트
    - CVM (커피자판기 컨트롤러) 대상
      - NuSCR 정형명세
      - SMV 모델체킹 검증



"System and Software Verification"
by B.Bérard, et. al., Springer

- 수업 평가
  - 학부 4학년 학생이 수업하기에 어려움
  - 정형기법을 수행하기 위한 이론 학습에 많은 시간 소요됨
    - 정형 언어, 오토마타, 논리 에 대한 이해 부족
    - 관련 기본이론 수업이 부족

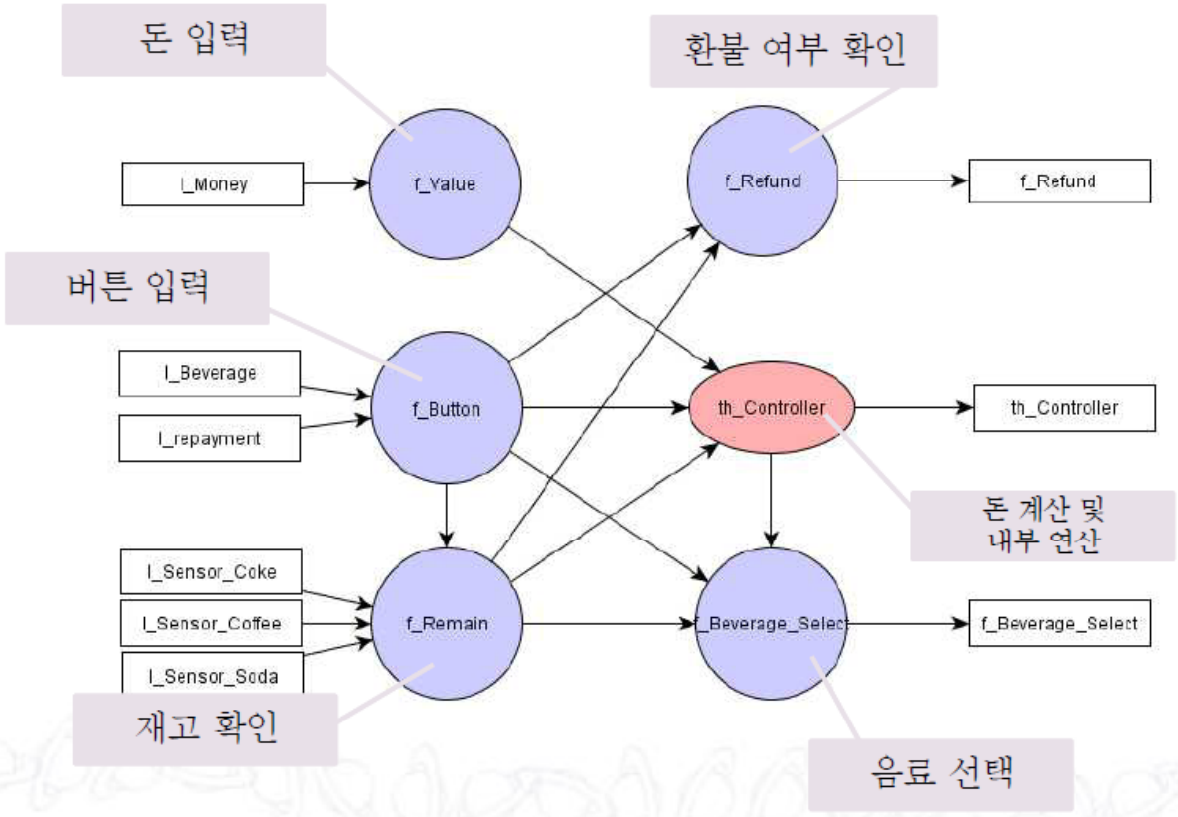
NUSRS를 이용한 커피 자판기 검증

교과목	소프트웨어 검증
담당교수님	유 준 범 교수님
Team 1	200212020 윤유성
	200212009 백형부
	200211994 김현길

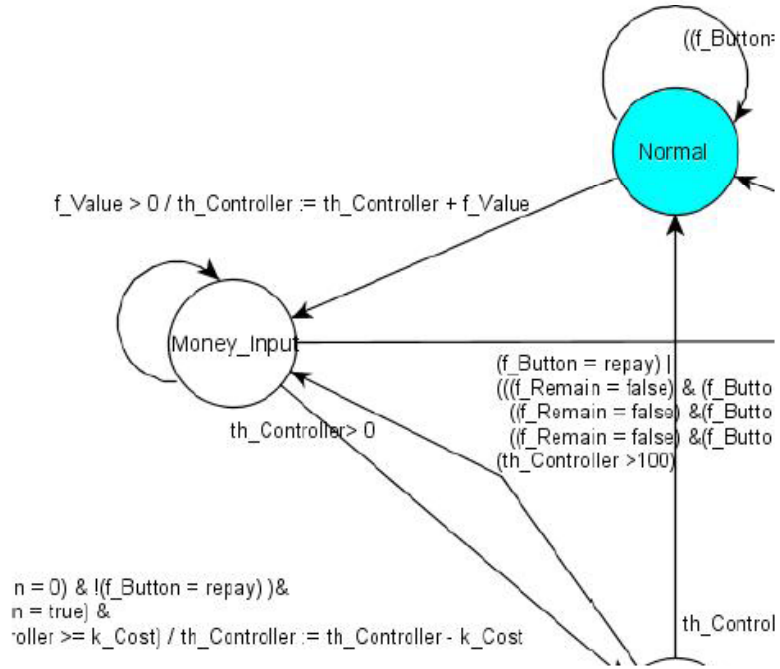
모델체킹 결과

대상	검증속성	검증속성 의미	검증결과	분석결과
g_Coffee_Bend	AG(f_Selection.f_Selection=1 -> AG th_Balance.th_Balance=0)	자판기의 동작이 '반환'이라면 잔액이 0이 된다.	true	검증속성이 만족됨을 확인함
	AG(f_Selection.f_Selection=1&th_Balance.th_Balance>0 -> AG h_Rebate.h_Rebate>0)	자판기의 동작이 '반환'이고 잔액이 남아 있다면 반환액이 생긴다.	true	검증속성이 만족됨을 확인함
	AG EX(f_Selection.f_Selection=2&f_Value.f_Value>0-> (th_Balance.th_Balance>0&th_Balance.th_Balance<=20))	자판기의 동작이 '동전 투입'이라면 잔액은 1~20의 값을 갖게 된다.	true	검증속성이 만족됨을 확인함
	AG EX(f_Selection.f_Selection=3&& th_Balance.th_Balance>=f_Value.f_Value -> h_Coffee.h_Coffee>0)	자판기의 동작이 '커피'이고 잔액이 커피금액보다 많으면 커피가 나온다.	true	검증속성이 만족됨을 확인함
	AG !((h_Rebate.h_Rebate & th_Balance.th_Balance) | (th_Balance.th_Balance>0 & h_Coffee.h_Coffee) | (h_Coffee.h_Coffee & h_Rebate.h_Rebate))	잔액, 반환액, 커피 중 동시에 두 가지 동작이 일어날 수 없다. (동시에 두 동작이 1이상의 값을 가질수 없다.)	true	검증속성이 만족됨을 확인함
	AG(f_Selection.f_Selection=3 & (th_Balance.th_Balance>f_Value.f_Value) -> AX th_Balance.th_Balance < th_Balance.th_Balance_t0)	자판기의 동작이 '커피'이고 잔액이 커피금액보다 많으면 다음 주기의 잔액은 현재 잔액보다 줄어든다.	true	검증속성이 만족됨을 확인함
	AG(f_Selection.f_Selection=0 -> AX th_Balance.th_Balance = th_Balance.th_Balance_t0)	자판기의 동작이 없다면 잔액은 유지된다.	true	검증속성이 만족됨을 확인함
	AG(f_Selection.f_Selection = 3 & f_Value.f_Value > 0 & th_Balance.th_Balance < f_Value.f_Value -> (AF h_Coffee.h_Coffee = 0))	자판기의 동작이 '커피'이고 잔액이 커피금액보다 적다면 커피는 나오지 않는다.	true	검증속성이 만족됨을 확인함
	AG(f_Selection.f_Selection = 3 & f_Value.f_Value = 1 -> (AF (th_Balance.th_Balance = th_Balance.th_Balance_t0 – f_Value.f_Value)))	커피가 나오면 잔액은 커피의 금액만큼 줄어든다	true	검증속성이 만족됨을 확인함

g_System (2)


돈 입력
환불 여부 확인
I_Money
f_Value
f_Refund
f_Refund
버튼 입력
I_Beverage
I_repayment
f_Button
th_Controller
th_Controller
돈 계산 및
내부 연산
I_Sensor_Coke
I_Sensor_Coffee
I_Sensor_Soda
f_Remain
f_Beverage_Select
f_Beverage_Select
재고 확인
음료 선택

Property - th_Controller (1)-1

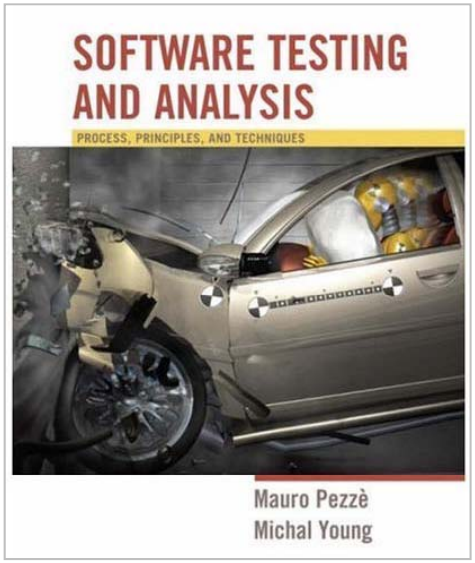


돈이 들어와서 음료 1개 값 이상의 잔액이 남으면, (음료가 나간 후) 반드시 그 다음에 Money_Input 상태로 돌아가야 함.

SPEC AG(FROM-Money_Input-To-Beverage_Out-taken & (th_Controller > k_Cost) -> AX(FROM-Beverage_Out-TO-Money_Input-taken)

2010 소프트웨어 검증 수업

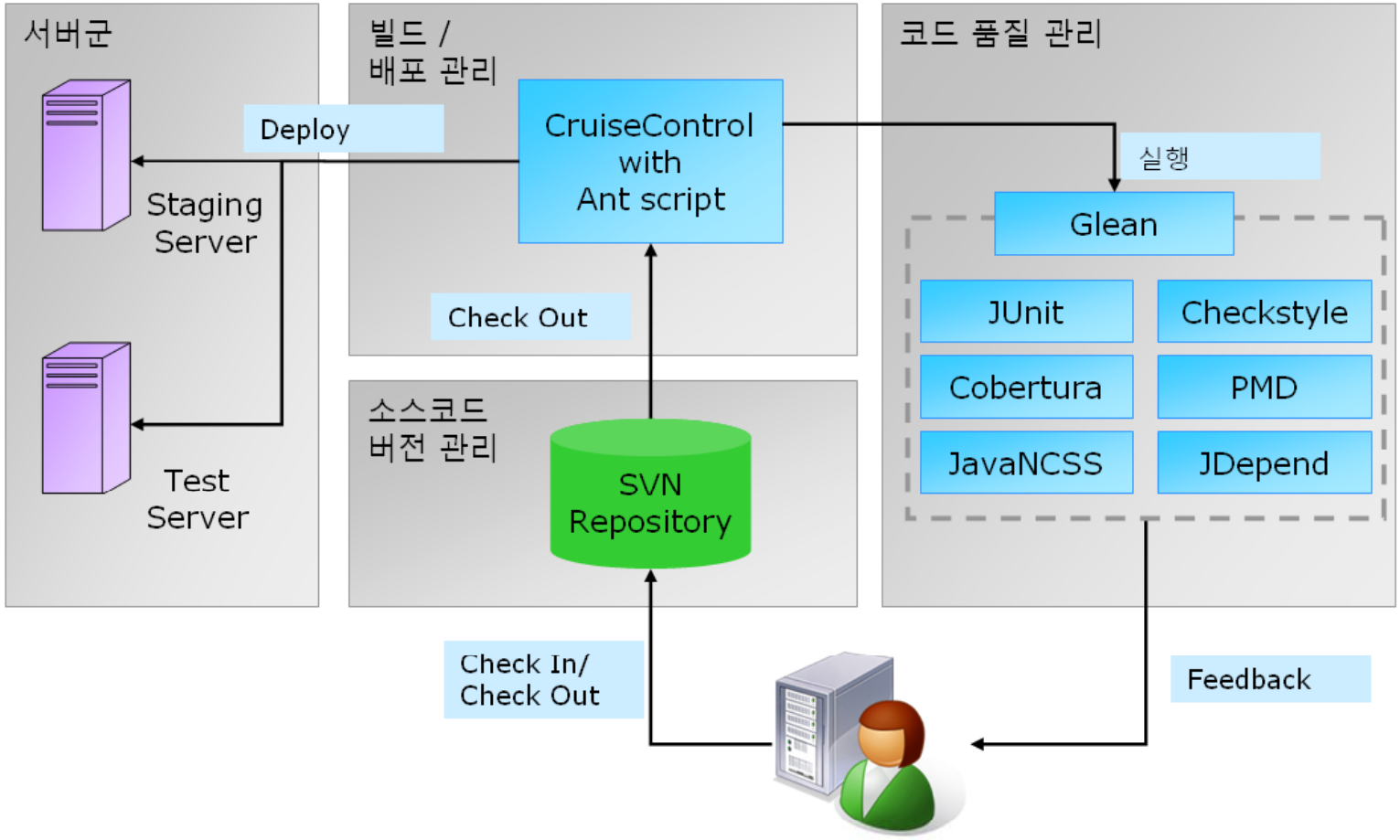
- 소프트웨어 테스팅 개론
  (http://dslab.konkuk.ac.kr/Class/2010/10SV/10SV.htm)
  - 이론
    - 소프트웨어 테스팅
    - 기본정의, 테스팅 기법분류, 관련 기법
  - 팀프로젝트
    - CTIP (Continuous Testing & Integration Platform) 구축
      - Java 프로그램 대상
      - 오픈소스 프로그램을 사용
      - CTIP 환경을 구축
      - Functional unit testing 수행
    - 5팀 (13명)



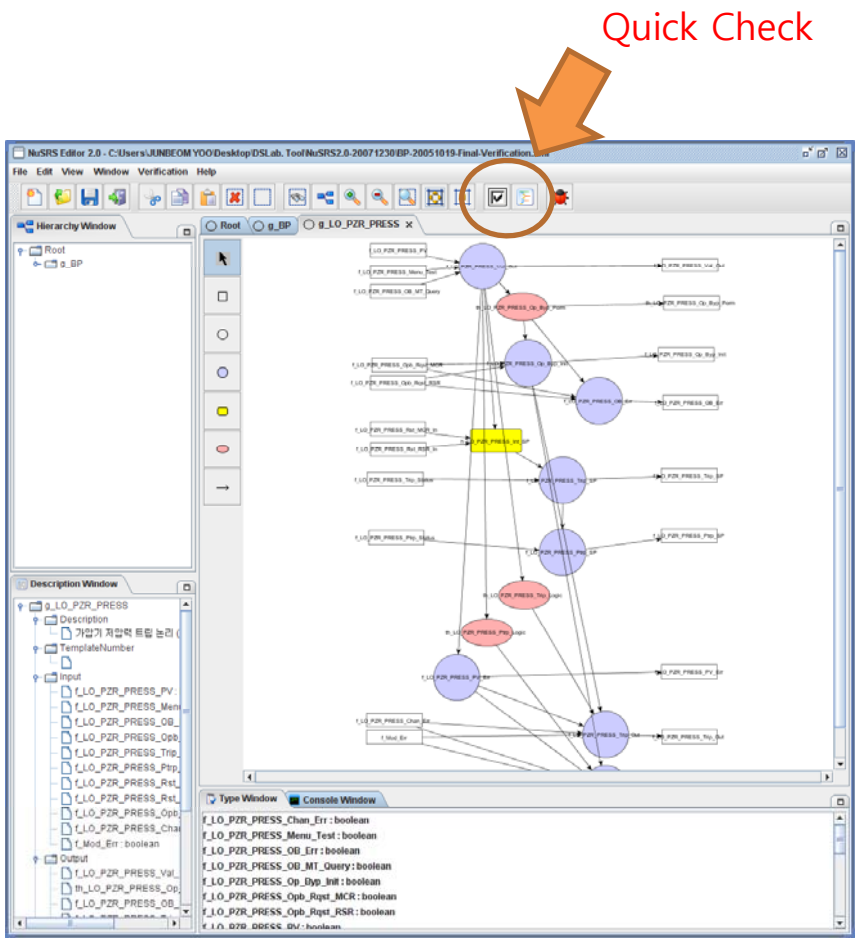
"Software Testing and Analysis"
Mauro Pezzè and Michal Young, WILEY

CTIP / CI

http://www.sereform.com/?m=20090302



Testing Target



NuSRS ver. 2.0

- Quick Check module
  - 명세에서 8 가지 오류를 찾는 모듈
- Testing Specification
  - ① SDT Condition, Action의 Undefined Variable
  - ② SDT Condition, Action에서 연산자 사용 잘못
  - ③ SDT에서 한 개의 Condition에 둘 이상의 Action 할당
  - ④ FSM, TTS Transition의 Undefined Variable
  - ⑤ FSM, TTS Transition의 연산자 사용 잘못
  - ⑥ FOD, FSM, TTS에서 Transition이 없는 노드
  - ⑦ FSM, TTS에서 Initial State로부터 Unrechable 노드
  - ⑧ FOD에서 Output 변수와 연결된 노드 동일 명칭 체크

이론수업 내용

Chapter 1. Software Test and Analysis in a Nutshell (fig)
Chapter 2. A Framework for Test and Analysis
Chapter 3. Basic Principles
Chapter 4. Test and Analysis Activities Within a Software process

기본 배경

Chapter 5. Finite Models
Chapter 6. Dependence and Data Flow Models
Chapter 8. Finite State Verification

기본 기술/이론

Chapter 9. Test Case Selection and Adequacy
Chapter 10. Functional Testing
Chapter 11. Combinatorial Testing
Chapter 12. Structural Testing
Chapter 13. Data Flow Testing
Chapter 14. Model based Testing
Chapter 15. Testing Object-Oriented Software
Chapter 16. Fault based Testing
Chapter 17. Test Execution

기법 및 방법론

팀프로젝트 내용

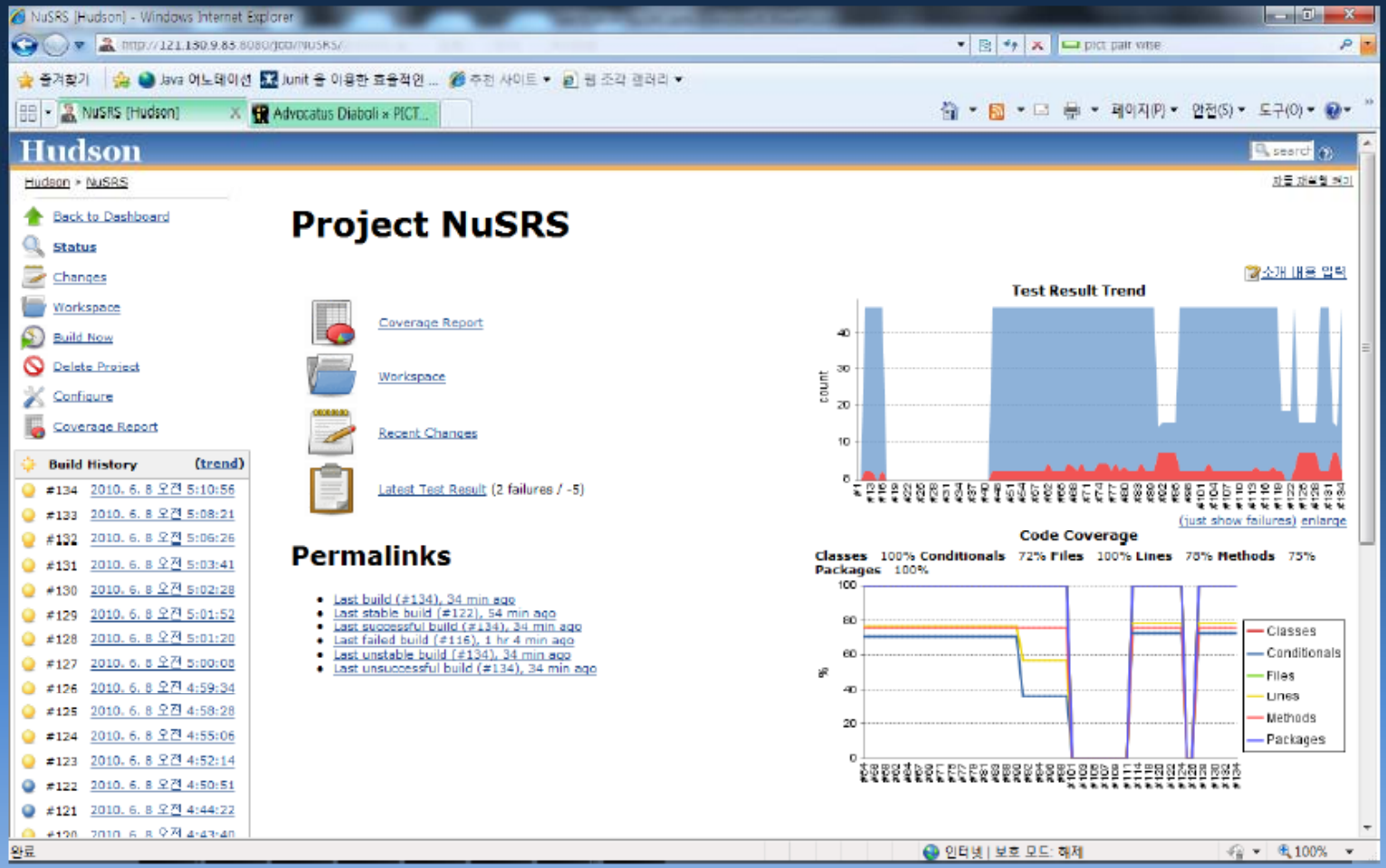
	T1	T2	T3	T4	T5
CI	Hudson [15]	-	Trac [16]	-	Hudson
Automatic Build	Ant	Ant	Ant	Ant	Ant
IDE (Integrated Development Env.)	Eclipse	Eclipse	Eclipse	Eclipse	Eclipse
Unit Test	JUnit	JUnit	JUnit	JUnit	JUnit
Requirements Management	JFeature	JFeature	JFeature	JFeature	JFeature
CM (Configuration Management)	Subclipse	Subversion	Google Code / Subclipse	CVS	Subversion
Coverage	Cobertura [17]	Eclemma [18]	Clover [19]	CodeCover [20]	Cobertura
Compiler	Java SDK	Java SDK	Java SDK	Java SDK	Java SDK
Etc.	Eclipse TPTP	-	Eclipse TPTP / FitNesse	-	Randoop

팀별 단위테스트 수행 결과

- Functional testing
- Unit testing

	1st round testing	2nd round testing
T1	Category-Partition Testing	Pairwise Testing
	44 test cases (2 fails)	15 test cases (7 fails)
	Spec.에 충실한 test 수행	PICT [21] 사용
T2	Brute Force Testing	Pairwise Testing
	18 test cases (2 fails)	37 test cases (7 Fails)
	-	TVG [22] 사용
T4	Category-Partition Testing	Pairwise Testing
	184 test cases	73 test cases (3 fails)
	중복되는 오류가 모두 검출되는 가에 집중함.	수동으로 직접 수행
T5	Brute Force Testing	Pairwise Testing
	47 test cases (7 fails)	37 test cases (12 fails)
	-	AllPairs [23] 사용

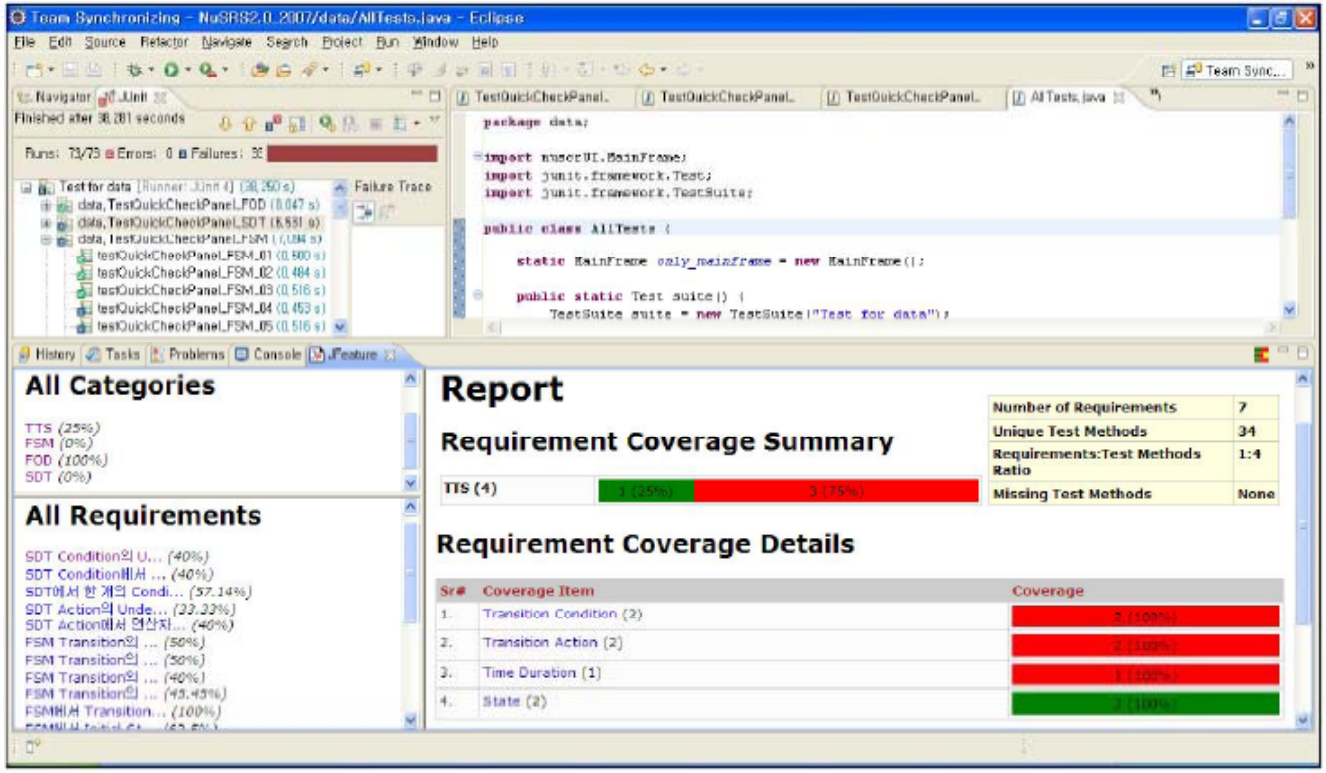
CTIP



Hudson CI 적용 결과 (T1)

8. Test Execution & Report in CTIP
2) Testing in CTIP – JFeature

➢ JFeature 확인



22

JFeature 적용 화면 (T4)

2. Category Parition Testing

Testcase Generate

- FOD

  12

Transition	Node Name	Node Type
yes	equal	function
no	no equal	history
		timed-history

- FSM

  192

Conditino Variable	Condition Operation	Conditino Pair	Action Variable	Action Operation	Action Pair	Transition
define	op_single	both side	define	op_single	both side	normal_direction
undefined	op_serial	one side	undefined	op_serial	one side	no transition
						abnormal_direction

- TTS

  1728

start_time	end_time	Conditino Variable	Condition Operation	Conditino Pair	Action Variable	Action Operation	Action Pair	Transition
number	number	define	op_single	one side	define	op_single	one side	no Transtiion
character	character	undefined	op_serial	both side	undefined	op_serial	both side	normal_direction
operation	operation							abnormal_direction

- SDT

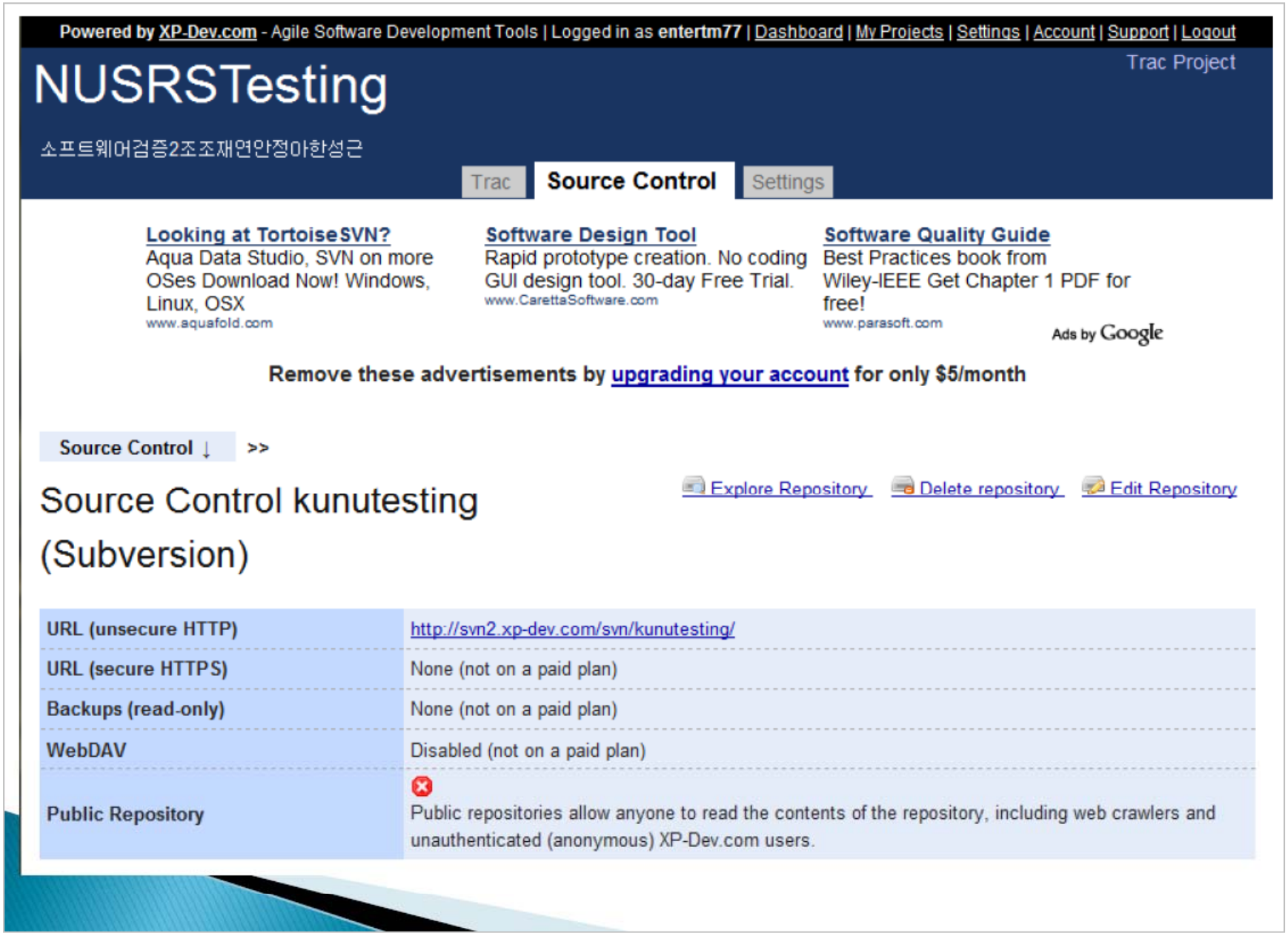
  192

Action Number	Conditino Variable	Condition Operation	Conditino Pair	Action Variable	Action Operation	Action Pair
0	define	op_single	one side	define	op_single	one side
1	undefine	op_serial	both side	undefined	op_serial	both side
many						

Total = 2124

KU 건국대학교 KONKUK UNIV.

Category-Partition Testing 적용 결과 (T5)



Source Control (CM) 적용 화면 (T2)

수업 결과 분석 (학생)

- 수강생 의견 (강의평가/인터뷰 취합 및 정리)

'소프트웨어 공학 개론' 수업을 통해 테스팅의 중요성은 알고 있었지만, 직접 실습해 봄으로써 그 중요성을 체험할 수 있었으며, 테스팅이 이론과 적용 측면에서 얼마나 어려운 학문 분야인가를 깨달았다.

테스팅을 위한 준비과정부터 테스트 실행 (Test Execution)을 포함한 CTIP 전 과정을 수업함으로써, 취업 후 업무에서 실제 적용할 수 있는 경험을 축적했다.

테스팅은 자동화 도구가 중심이 되는 구조 시험 (Structural Testing) 뿐만 아니라, 전문가의 능력이 중시되는 기능시험 (Functional Testing)도 중요하고 널리 사용된다는 사실을 이해했다.

테스트 전문가의 경험과 능력이 테스트 결과에 결정적인 영향을 끼침을 체험했으며, 졸업 후 일반적인 소프트웨어 엔지니어 보다는 테스팅과 같은 특화된 분야에 전문성을 가지는 전문가가 되어야겠다는 생각을 했다.

수업 결과 분석 (교수)

- 담당교수 의견

테스팅은 학부 4학년 학생이 수강하기에 충분한 기법이다.

테스팅 이론과 함께 적절히 높은 수준의 팀 프로젝트가 필요하다.

기능시험과 구조시험을 포괄하는 팀 프로젝트의 개발이 필요하며, JAVA와 C 언어를 모두 포함하면 좋겠다

구조시험에서는 산업계의 여러 자동화 도구를 사용하는 실습을 수행하면 효과적일 것으로 판단된다.

결론 및 향후 운영 방안

- 학부생을 대상으로 하는 Dependability 교육의 필요성이 증가
- 여러 기법들 중, 4학년을 대상으로, Software Testing 수업 실시

- 우려와는 달리, 학부 4학년 학생은
  - 테스팅의 제반 이론을 충분히 습득할 수 있으며,
  - 팀프로젝트를 통해, 이론을 실제로 적용할 수 있는 충분한 능력이 있음을 확임

- 이 수업을 통해,
  - 학생들이 졸업 후, '테스팅 환경 구축 전문가' (SE팀, SQA)로서 활약할 수 있는 자질과 능력을 갖추었음을 확인

- 향후에는,
  - 기능시험과 구조시험을 동시에 구현 및 실습
  - 다양한 언어 (C/C++/Java)를 대상
  - 구조/통합/시스템 시험 전체 life-cycle