Software Verification

System Testing #2 & Static Analysis

201011329 박대규
201111393 최서현
201111374 윤원경

Software Verification

INDEX

01 jFeature

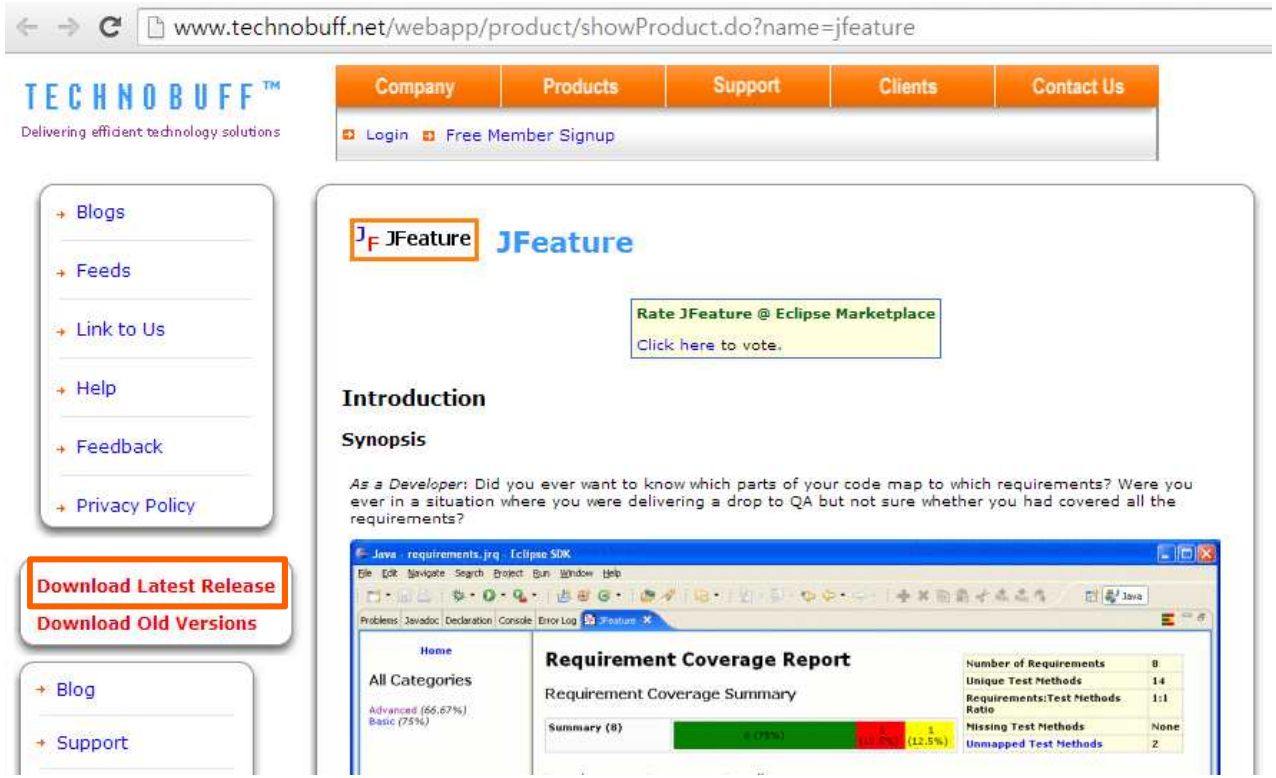
What is jFeature?

- jFeature is the tool for connecting jUnit test cases with requirements.

- You can use jFeature as Eclipse plugin, it's easy to install and use.

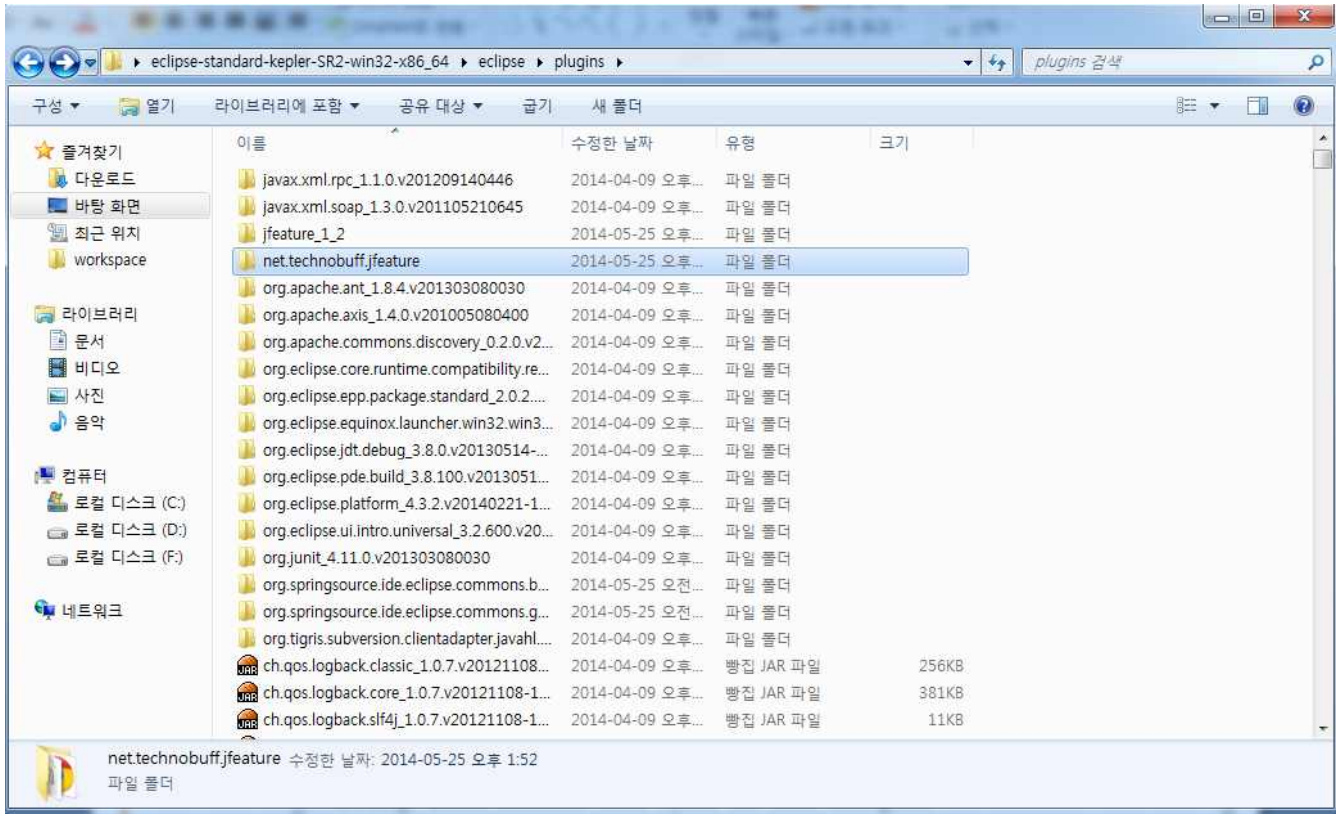
- jFeature offers requirement coverage report and round trip engineering.

How to install jFeature



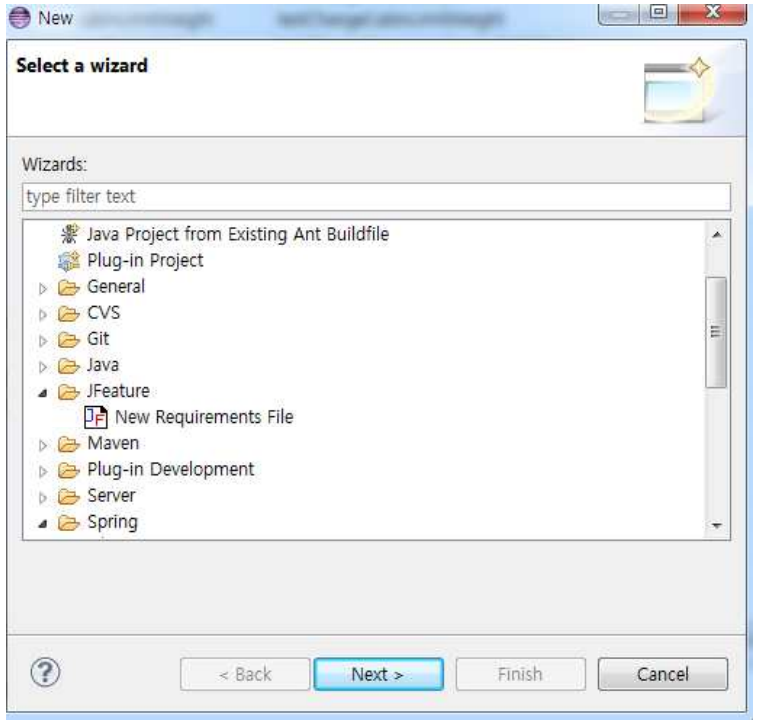
http://www.technobuff.net/webapp/product/showProduct.do?name=jfeature

How to install jFeature



- Unzip and locate the files to the eclipse\plugins

How to use jFeature



- Open eclipse and right click your project-[new]-[others]-[jFeature]-[New Requirements File]

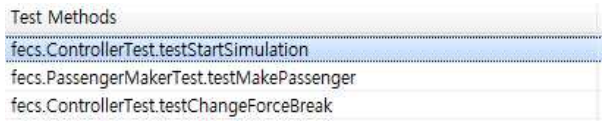
How to use jFeature

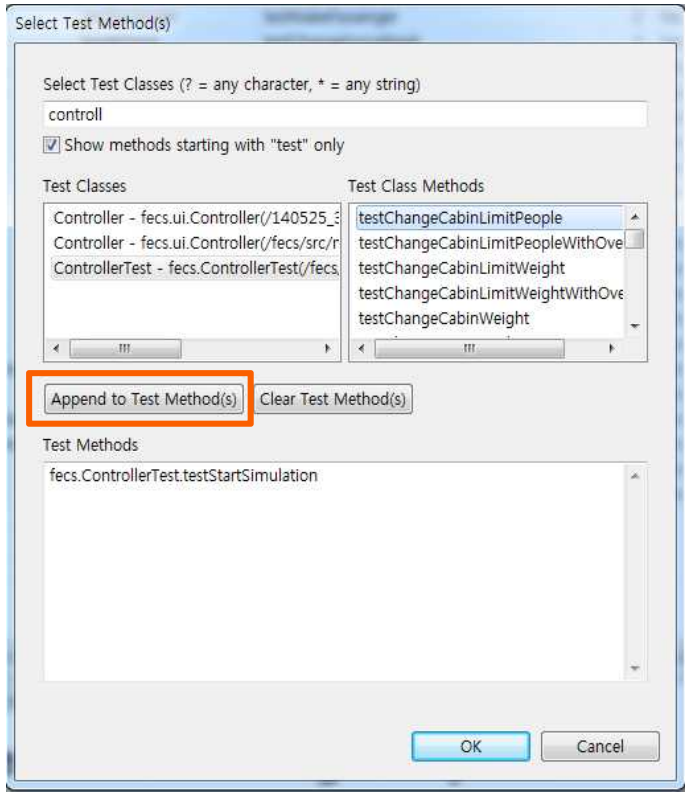
ICircumstance.java | ControllerTest.java | AbstractSpringBasedTestSupport.java | fecs/pom.xml | UserInterface.java | requirements.jrq

Requirement ID	Category	Sub Category	Title	T	P.	Mu...	Test Methods
REQ_1_02_1	sysAction	simulation	testStartSimulation		1	No	fecs.ControllerTest.testStartSimulation
REQ_1_04_3	config	genPassenger	testMakePassenger		2	No	fecs.PassengerMakerTest.testMakePassenger
REQ_1_08_1	config	breakForce	testChangeForceBreak		2	No	fecs.ControllerTest.testChangeForceBreak
REQ_1_07_1	config	motorOut	testChangeMotorOutput		2	No	fecs.ControllerTest.testChangeMotorOutput
REQ_1_13_1	config	cabinLimitWeight	testChangeCabinLimitWeight		2	No	fecs.ControllerTest.testChangeCabinLimitWeight
REQ_1_04_2	config	genPassenger	testChangePassengerCreatedWithOver		4	No	fecs.ControllerTest.testChangePassengerCreatedWithOver
REQ_1_04_1	config	genPassenger	testChangePassengerCreated		2	No	fecs.ControllerTest.testChangePassengerCreated
REQ_1_12_2	config	totalPassenger	testChangeTotalNumPassengersWithOver		4	No	fecs.ControllerTest.testChangeTotalNumPassengersWithOver
REQ_1_12_1	config	totalPassenger	testChangeTotalNumPassenger		2	No	fecs.ControllerTest.testChangeTotalNumPassengers
REQ_1_09_1	config	passengerWeight	testChangePassengerWeight		2	No	fecs.ControllerTest.testChangePassengerWeight
REQ_1_05_1	config	cabinLimitPassenger	testChangeCabinLimitPeople		2	No	fecs.ControllerTest.testChangeCabinLimitPeople
REQ_1_06_3	config	gravity	testChangeGravityUnder0		4	No	fecs.ControllerTest.testChangeGravityUnder0
REQ_1_11_1	config	cabinWeight	testChangeCabinWeight		2	No	fecs.ControllerTest.testChangeCabinWeight
REQ_1_06_2	config	gravity	testChangeGravityOver50		4	No	fecs.ControllerTest.testChangeGravityOver50
REQ_1_06_1	config	gravity	testChangeGravity		2	No	fecs.ControllerTest.testChangeGravity
REQ_1_03_1	sysAction	simulation	testStopSimulation		1	No	fecs.ControllerTest.testStopSimulation
REQ_1_13_2	config	cabinLimitWeight	testChangeCabinLimitWithOver		4	No	fecs.ControllerTest.testChangeCabinLimitWeightWithOver
REQ_1_03_2	sysAction	simulation	testStopSimulationWithOthers		3	No	fecs.ControllerTest.testStopSimulationWithOthers
REQ_1_01_1	sysAction	simulation	testTriggerFail		3	No	fecs.ControllerTest.testTriggerFail
REQ_1_05_2	config	cabinLimitPassenger	testChangeCabinLimitPeopleWithOverLimit		4	No	fecs.ControllerTest.testChangeCabinLimitPeopleWithOverLimit
REQ_1_10_1	config	enterProbability	testChangeMoreEnterProbability		2	No	fecs.ControllerTest.testChangeMoreEnterProbability

- Fill the requirement ID, category, sub category, title, priority… etc.

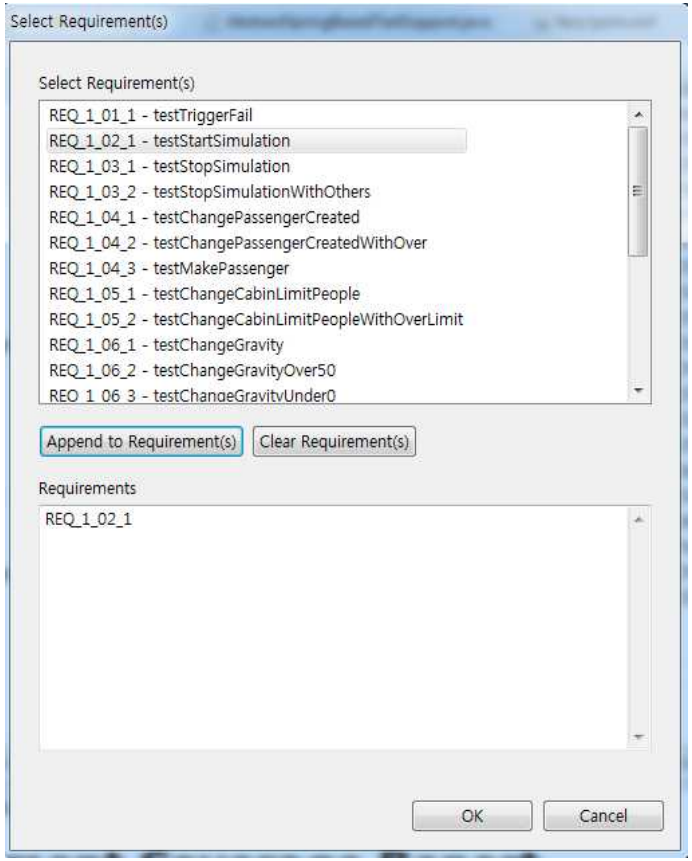
How to use jFeature





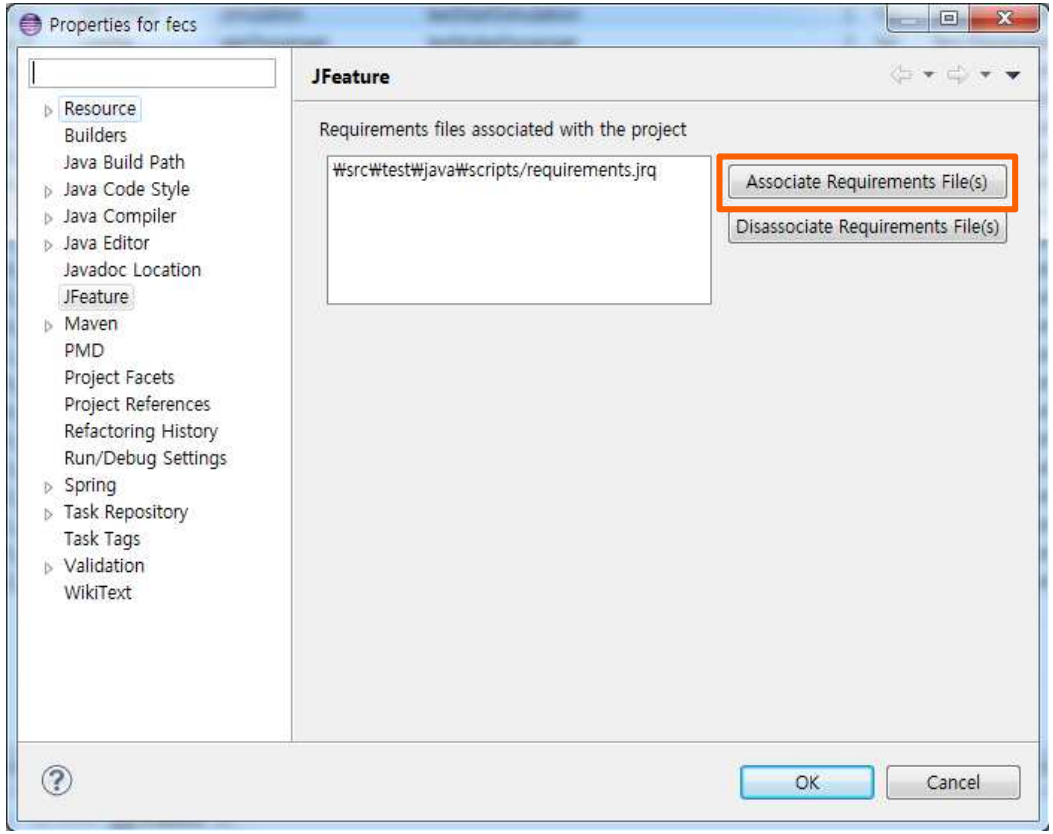
- Click the test methods button and append to test methods.

How to use jFeature



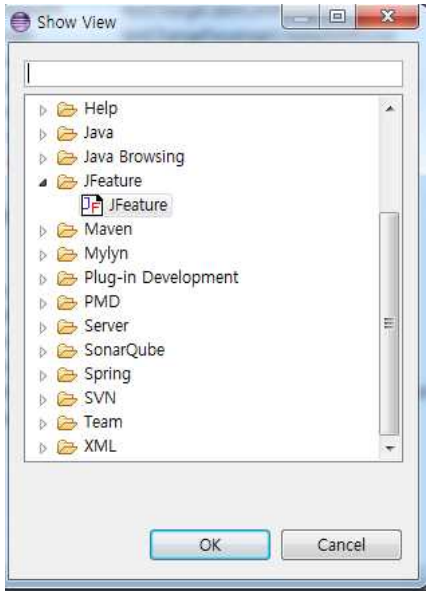
- You can also define the dependencies between test case.

How to use jFeature



- If you finish, you need to match jrq file and the project.
- Right click your project-[properties]-[jFeature]-[Associate Requirements Files]

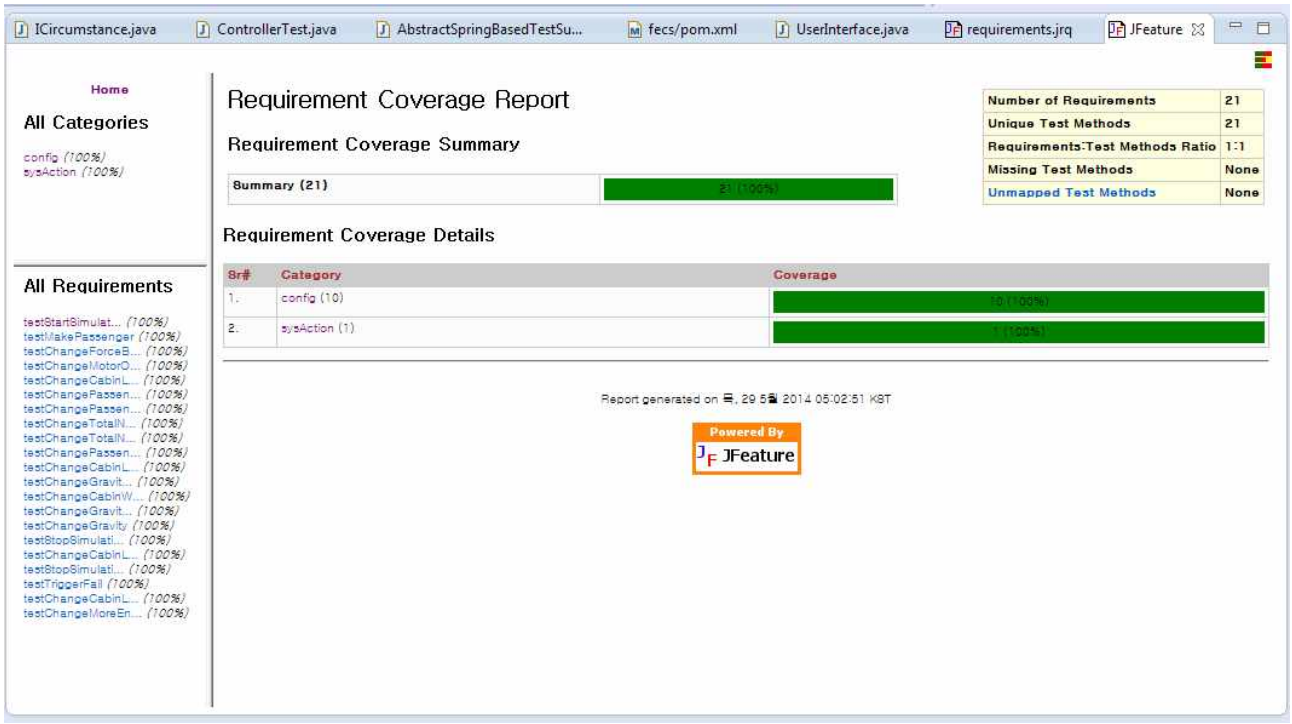
How to use jFeature



- Right click your jrq file-[Run as]-[jUnit test case]
- [Window]-[Show View]-[jFeature] and click small button in upper right side.

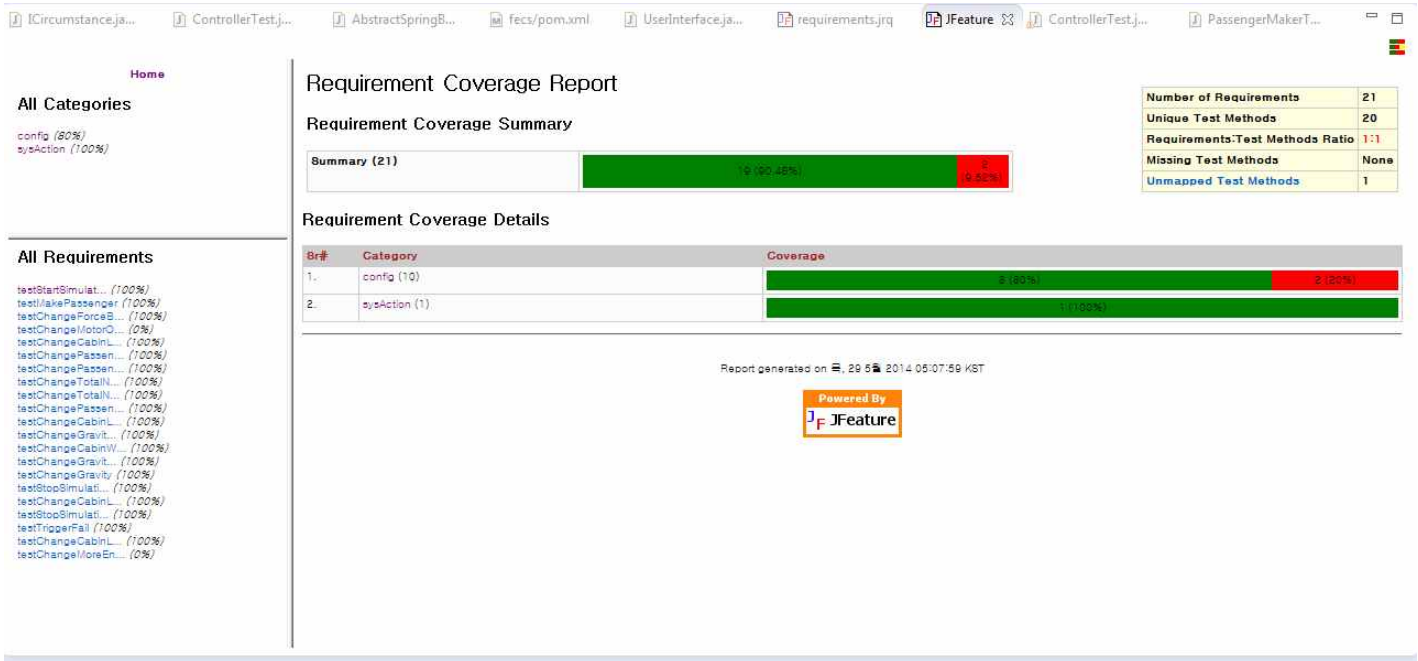
How to use jFeature

- 01 jFeature
- 02 System Testing
- 03 Testlink
- 04 Static Analysis

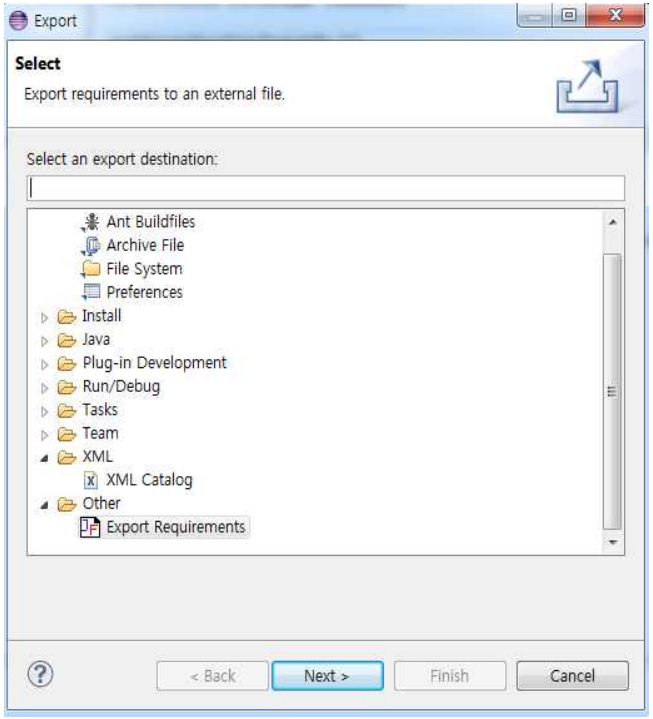


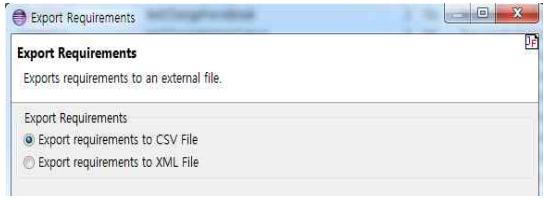
How to use jFeature

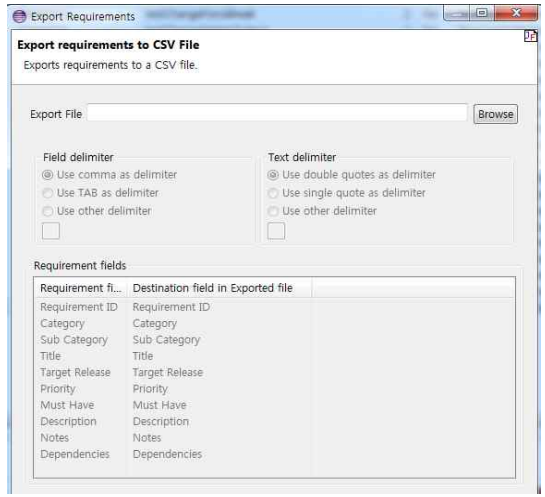
- 01 jFeature
- 02 System Testing
- 03 Testlink
- 04 Static Analysis



How to use jFeature







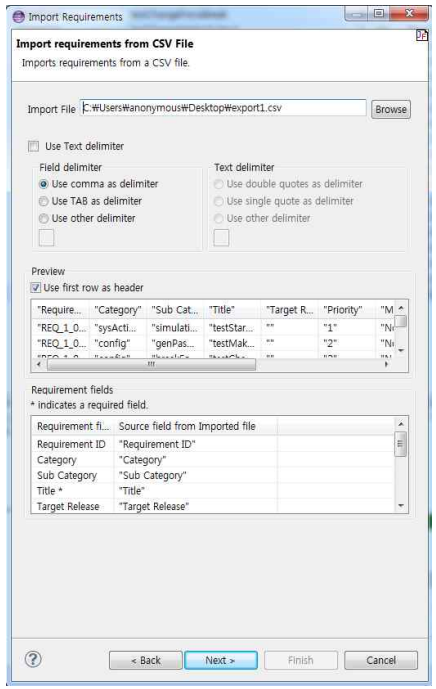
- Right click your jrq file-[Export]

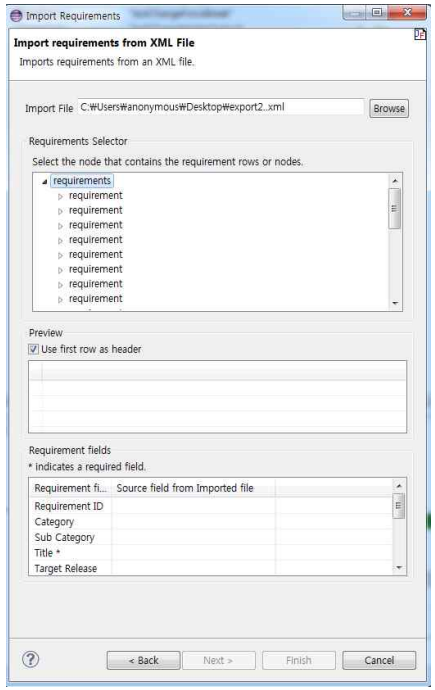
How to use jFeature

A	B	C	D	E	F	G
Requirem	Category	Sub Categ	Title	Target Rel	Priority	Must Have
REQ_1_02_	sysAction	simulation	testStartSimulation		1	No
REQ_1_04_	config	genPasser	testMakePassenger		2	No
REQ_1_08_	config	breakForc	testChangeForceBrea		2	No
REQ_1_07_	config	motorOut	testChangeMotorOu		2	No
REQ_1_13_	config	cabinLimit	testChangeCabinLim		2	No
REQ_1_04_	config	genPasser	testChangePassenge		4	No
REQ_1_04_	config	genPasser	testChangePassenge		2	No
REQ_1_12_	config	totalPasse	testChangeTotalNum		4	No
REQ_1_12_	config	totalPasse	testChangeTotalNum		2	No
REQ_1_09_	config	passenger	testChangePassenge		2	No
REQ_1_05_	config	cabinLimit	testChangeCabinLim		2	No
REQ_1_06_	config	gravity	testChangeGravityUr		4	No
REQ_1_11_	config	cabinWeig	testChangeCabinWe		2	No
REQ_1_06_	config	gravity	testChangeGravityO\		4	No
REQ_1_06_	config	gravity	testChangeGravity		2	No
REQ_1_03_	sysAction	simulation	testStopSimulation		1	No
REQ_1_13_	config	cabinLimit	testChangeCabinLim		4	No
REQ_1_03_	sysAction	simulation	testStopSimulationW		3	No
REQ_1_01_	sysAction	simulation	testTriggerFail		3	No
REQ_1_05_	config	cabinLimit	testChangeCabinLim		4	No
REQ_1_10_	config	enterProb	testChangeMoreEnte		2	No

```
----+----1----+----2----+----3----+----4----+----
<?xml version="1.0" encoding="UTF-8"?>
<requirements>
  <requirement>
    <data><![CDATA[Requirement ID]]></data>
    <data><![CDATA[Category]]></data>
    <data><![CDATA[Sub Category]]></data>
    <data><![CDATA[Title]]></data>
    <data><![CDATA[Target Release]]></data>
    <data><![CDATA[Priority]]></data>
    <data><![CDATA[Must Have]]></data>
    <data><![CDATA[Description]]></data>
    <data><![CDATA[Notes]]></data>
    <data><![CDATA[Dependencies]]></data>
  </requirement>
  <requirement>
    <data><![CDATA[REQ_1_02_1]]></data>
    <data><![CDATA[sysAction]]></data>
    <data><![CDATA[simulation]]></data>
    <data><![CDATA[testStartSimulation]]></data>
    <data><![CDATA[]]></data>
    <data><![CDATA[1]]></data>
    <data><![CDATA[No]]></data>
    <data><![CDATA[]]></data>
    <data><![CDATA[]]></data>
    <data><![CDATA[]]></data>
  </requirement>
  <requirement>
    <data><![CDATA[REQ_1_04_3]]></data>
    <data><![CDATA[config]]></data>
    <data><![CDATA[genPassenger]]></data>
    <data><![CDATA[testMakePassenger]]></data>
    <data><![CDATA[]]></data>
    <data><![CDATA[2]]></data>
    <data><![CDATA[No]]></data>
    <data><![CDATA[]]></data>
    <data><![CDATA[]]></data>
    <data><![CDATA[]]></data>
  </requirement>
```

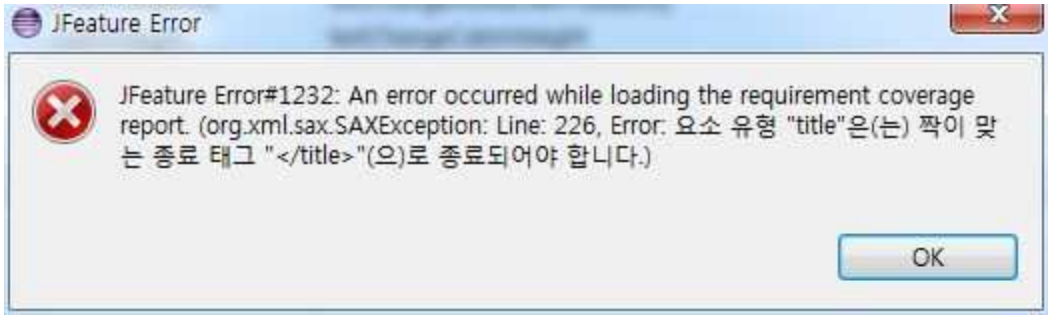
How to use jFeature





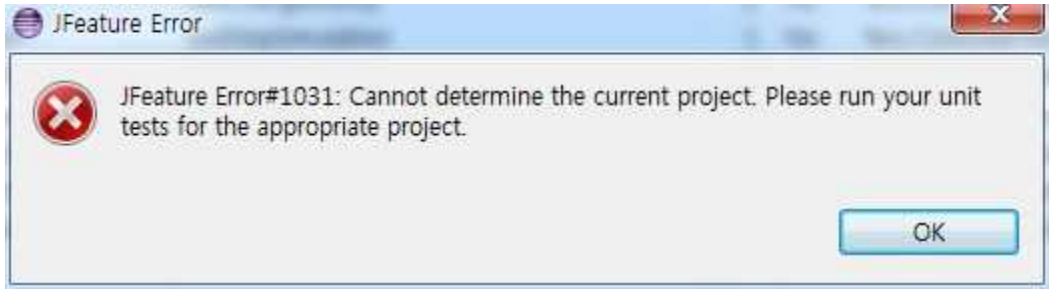
- Right click your project-[Import]

Trouble Shooting



- Encoding problem!
- Replace Korean to English.

Trouble Shooting



- Run the unit test.

After using jFeature…

- jFeature is strong tool for managing requirements.

- Also, you can see the unit test coverage report.

- But….

- There are a lot of errors for export and import.

- It’s based on jrq files but you cannot import jrq file.

02 System Testing

System testing report

01 jFeature

02 System Testing

03 Testlink

04 Static Analysis

case	조합	설명	이전결과	이후결과
1	102.301	승객생성에 0이하의 정수나 혹은 실수를 넣고 시뮬레이션을 시작한다.	pass	pass
2	112.301	모터출력에 0이하의 실수를 넣고 시뮬레이션을 시작한다.	pass	pass
3	122.301	전체승객수에 0이하의 정수나 혹은 실수를 넣고 시뮬레이션을 시작한다.	pass	pass
4	132.301	캐빈정원에 0이하의 정수나 혹은 실수를 넣고 시뮬레이션을 시작한다.	pass	pass
5	301.302	시뮬레이션을 시작한 후 정지한다	pass	pass
6	301.202.311.401	시뮬레이션을 시작한 후 운행속도가 표시될 때 화재 버튼을 클릭하면 인명피해가 발생한다.	fail	pass
7	301.202.311.402	시뮬레이션을 시작한 후 운행속도가 표시될 때 화재 버튼을 클릭하면 ev 우선동작이 발생한다.	fail	fail
8	301.202.311.403	시뮬레이션을 시작한 후 운행속도가 표시될 때 화재 버튼을 클릭하면 화재진압이 된다.	fail	pass
9	301.202.312.411	시뮬레이션을 시작한 후 운행속도가 표시될 때 수해 버튼을 클릭하면 1층에 인명피해가 발생한다.	fail	pass
10	301.202.312.412	시뮬레이션을 시작한 후 운행속도가 표시될 때 수해 버튼을 클릭하면 ev 우선동작이 발생한다.	fail	pass
11	301.202.313.421	시뮬레이션을 시작한 후 운행속도가 표시될 때 지진 버튼을 클릭하면 랜덤하게 조당 한사람씩 사망한다.	fail	fail
12	301.202.313.422	시뮬레이션을 시작한 후 운행속도가 표시될 때 지진 버튼을 클릭하면 ev 정지가 발생한다.	fail	pass
13	301.202.314.431	시뮬레이션을 시작한 후 운행속도가 표시될 때 추락 버튼을 클릭하면 인명피해가 발생한다.	fail	pass
14	301.202.314.432	시뮬레이션을 시작한 후 운행속도가 표시될 때 추락 버튼을 클릭하면 추락하지 않은 캐빈이 멈춘다.	fail	fail
15	301.201.142.311.401	시뮬레이션을 시작한 후 캐빈무게가 표시될 때 캐빈한계무게를 현재총무게보다 크게 설정하면 화재버튼이 동작하여 인명피해가 발생한다.	fail	pass
16	301.201.142.311.402	시뮬레이션을 시작한 후 캐빈무게가 표시될 때 캐빈한계무게를 현재총무게보다 크게 설정하면 화재버튼이 동작하여 ev 우선동작이 발생한다.	fail	fail
17	301.201.142.311.403	시뮬레이션을 시작한 후 캐빈무게가 표시될 때 캐빈한계무게를 현재총무게보다 크게 설정하면 화재버튼이 동작하여 화재진압이 된다.	fail	pass
18	301.201.141.311.401	시뮬레이션을 시작한 후 캐빈무게가 표시될 때 캐빈한계무게를 현재총무게보다 작거나 같게 설정하면 화재버튼이 동작하여 인명피해가 발생한다.	fail	pass
19	301.201.141.311.402.	시뮬레이션을 시작한 후 캐빈무게가 표시될 때 캐빈한계무게를 현재총무게보다 작거나 같게 설정하면 화재버튼이 동작하여 ev 우선동작이 발생한다.	fail	pass
20	301.201.141.311.403	시뮬레이션을 시작한 후 캐빈무게가 표시될 때 캐빈한계무게를 현재총무게보다 작거나 같게 설정하면 화재버튼이 동작하여 화재진압이 된다.	fail	pass
21	301.201.142.312.411	시뮬레이션을 시작한 후 캐빈무게가 표시될 때 캐빈한계무게를 현재총무게보다 크게 설정하면 수해버튼이 동작하여 1층에 인명피해가 발생한다.	fail	pass
22	301.201.142.312.412	시뮬레이션을 시작한 후 캐빈무게가 표시될 때 캐빈한계무게를 현재총무게보다 크게 설정하면 수해버튼이 동작하여 ev 우선동작이 발생한다.	fail	pass
23	301.201.141.312.411	시뮬레이션을 시작한 후 캐빈무게가 표시될 때 캐빈한계무게를 현재총무게보다 작거나 같게 설정하면 수해버튼이 동작하여 1층에 인명피해가 발생한다.	fail	pass
24	301.201.141.312.412	시뮬레이션을 시작한 후 캐빈무게가 표시될 때 캐빈한계무게를 현재총무게보다 작거나 같게 설정하면 수해버튼이 동작하여 ev 우선동작이 발생한다.	fail	pass
25	301.201.142.313.421	시뮬레이션을 시작한 후 캐빈무게가 표시될 때 캐빈한계무게를 현재총무게보다 크게 설정하면 지진버튼이 동작하여 랜덤하게 조당 한사람씩 사망한다.	fail	fail
26	301.201.142.313.422	시뮬레이션을 시작한 후 캐빈무게가 표시될 때 캐빈한계무게를 현재총무게보다 크게 설정하면 지진버튼이 동작하여 ev 정지가 발생한다.	fail	pass
27	301.201.141.313.421	시뮬레이션을 시작한 후 캐빈무게가 표시될 때 캐빈한계무게를 현재총무게보다 작거나 같게 설정하면 지진버튼이 동작하여 랜덤하게 조당 한사람씩 사망한다.	fail	pass
28	301.201.141.313.422	시뮬레이션을 시작한 후 캐빈무게가 표시될 때 캐빈한계무게를 현재총무게보다 작거나 같게 설정하면 지진버튼이 동작하여 ev 정지가 발생한다.	fail	pass
29	301.201.142.314.431	시뮬레이션을 시작한 후 캐빈무게가 표시될 때 캐빈한계무게를 현재총무게보다 크게 설정하면 추락버튼이 동작하여 인명피해가 발생한다.	fail	pass
30	301.201.142.314.432	시뮬레이션을 시작한 후 캐빈무게가 표시될 때 캐빈한계무게를 현재총무게보다 크게 설정하면 추락버튼이 동작하여 추락하지 않은 캐빈이 멈춘다.	fail	fail
31	301.201.141.314.431	시뮬레이션을 시작한 후 캐빈무게가 표시될 때 캐빈한계무게를 현재총무게보다 작거나 같게 설정하면 추락버튼이 동작하여 인명피해가 발생한다.	fail	pass
32	301.201.141.314.432	시뮬레이션을 시작한 후 캐빈무게가 표시될 때 캐빈한계무게를 현재총무게보다 작거나 같게 설정하면 추락버튼이 동작하여 추락하지 않은 캐빈이 멈춘다.	fail	pass

- Combinatorial testing

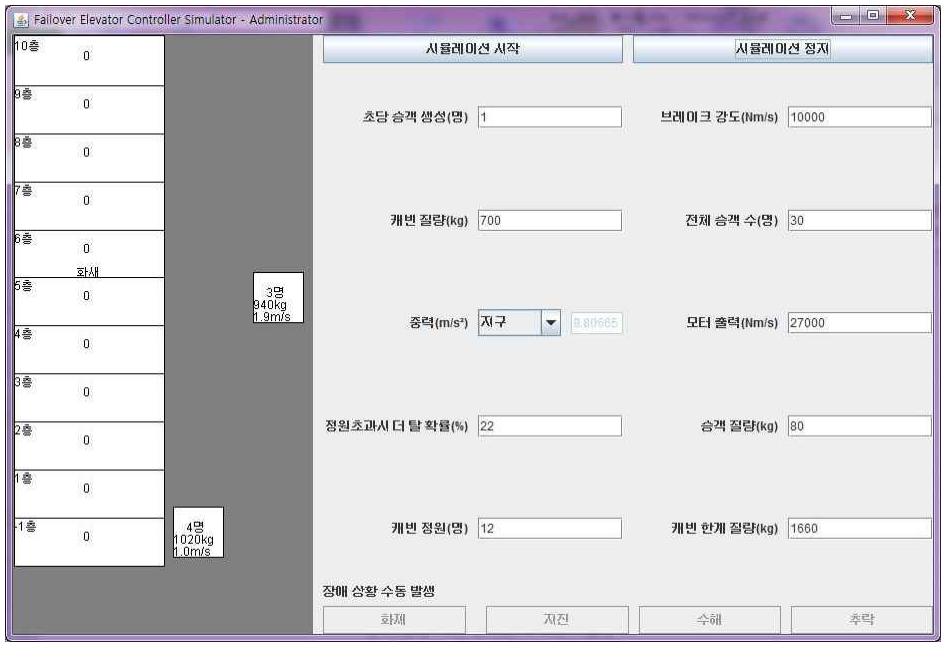
System testing report

BRUTE FORCE TESTING			
33	어떠한 내/외적 상황에서도 시스템이 정지되지 않는다.	fail	fail
34	시뮬레이션 시작명령 후 10초 이내에 시뮬레이션이 시작된다.	pass	pass
35	2개의 캐빈이 같은 방향으로 움직여야 하는 경우에 같은 층에 멈추는 일이 없도록 한다.	pass	pass
36	승객 생성 설정에 따라 1초 이내에 1층에만 입력한 숫자의 승객이 추가된다.	fail	pass
37	현재 승객수와 설정한 추가될 승객 수를 합산하여 설정된 전체 승객 수보다 크면, 그 차이만큼만 생성된다.	fail	pass
38	승객이 탑승 시에 캐빈의 무게는 추가되나 가속도는 일정하다.	pass	pass
39	정원 초과될 경우에 승객이 탈 확률을 설정하면 확률에 따라 승객이 탑승한다.	pass	pass
40	정원 초과시 탑승할 확률이 0초과 1이하의 값이 아니면 에러를 발생시킨다.	fail	pass
41	중력 설정에 입력한 값이 0초과 50이하의 정수가 아니면 에러를 발생시킨다.	fail	pass
42	캐빈 무게에 입력한 값이 양의 실수가 아니면 에러를 발생시킨다.	pass	pass
43	브레이크 강도에 입력한 값이 양의 실수가 아니면 에러를 발생시킨다.	pass	pass
44	승객 무게가 0보다 작거나 같다면 에러를 발생시킨다.	pass	pass
45	화재버튼을 클릭하면 화재를 일으킬 층을 입력한다.	pass	pass
46	추락버튼을 클릭하면 추락을 일으킬 캐빈을 선택한다.	pass	pass
47	장애 상황 시에 승객은 추가되지 않는다.	fail	pass
48	장애가 발생 했을 시 장애를 표시한다.	fail	pass
49	화재 발생 시에 소방관은 1층에만 생성된다.	fail	pass
50	소방관이 화재가 난 층에 도착하게 되면 진압이 완료된다.	fail	pass
51	수해시 두개의 캐빈을 지상 10층에서부터 다시 작동시킨다.	pass	pass
52	지진 발생 시 1분동안 지진 상황이 지속된 후 종료된다.	pass	pass
53	추락한 캐빈이 지하 1층에 도착하거나 브레이크에 의해 멈춘다면 추락은 종료된다.	fail	pass

- Brute force testing

Test case #7

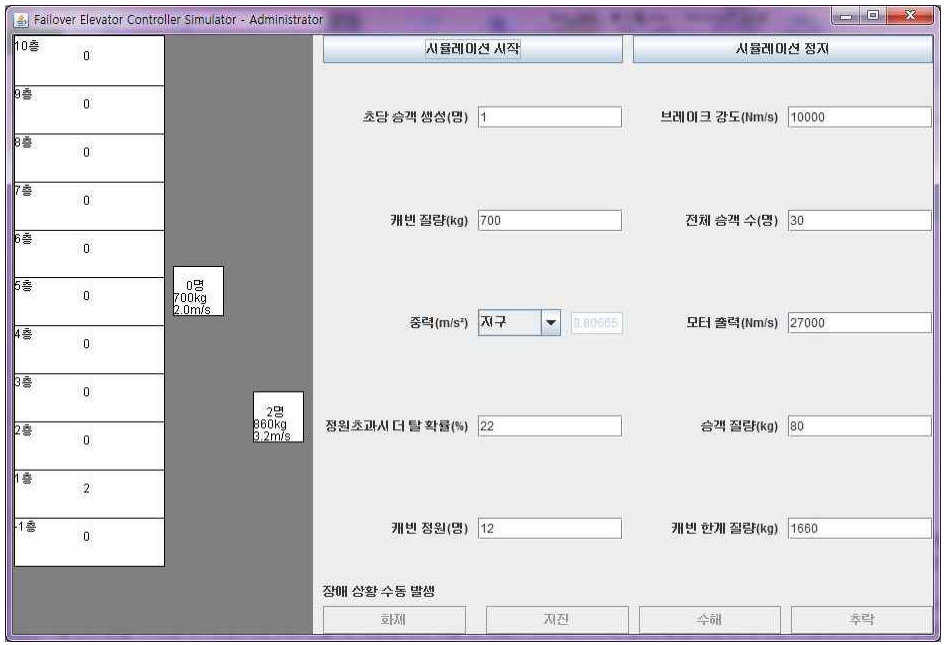
- 301.202.311.402 : 시뮬레이션 시작 후 운행속도가 표시될 때 화재버튼을 클릭하면 **ev** 우선동작이 발생한다.



- EV가 **1**층과 화재 층에 우선동작하지 않고 **1**층은 우선동작 하지만 다른 층에 멈췄다가 화재가 난 층으로 간다.

Test case #11

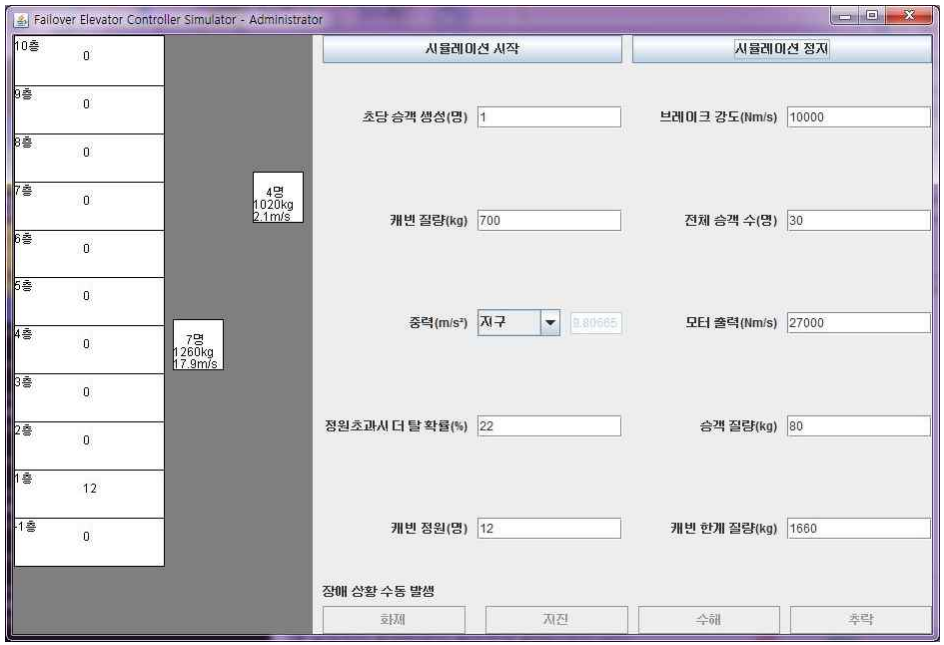
- 301.202.313.421 : 시뮬레이션을 시작한 후 운행속도가 표시될 때 지진 버튼을 클릭하면 랜덤하게 초당 한사람씩 사망한다.



- 아무도 죽지 않아 사람 수에 변화가 없다.

Test case #14

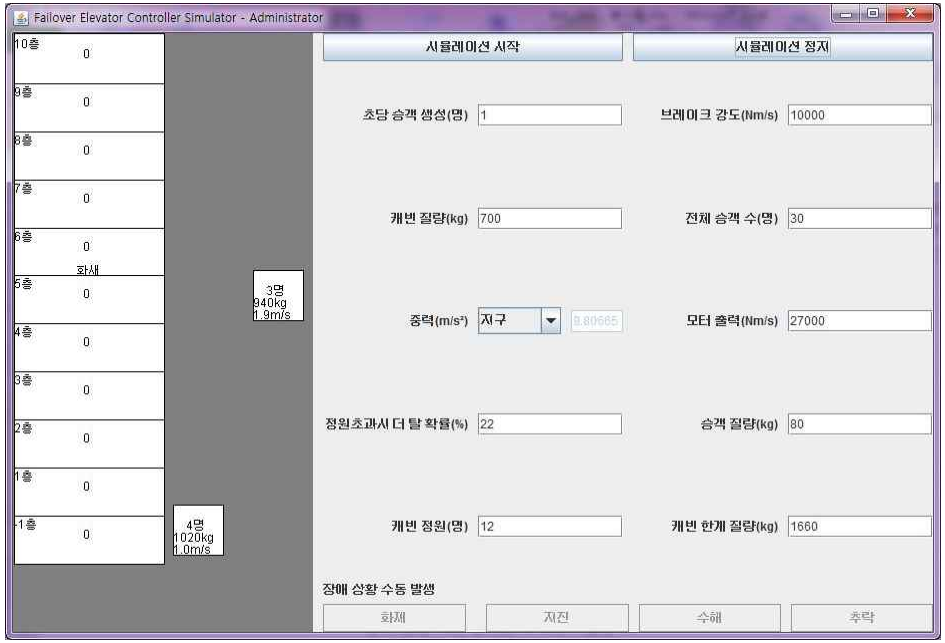
- 301.202.314.432 : 시뮬레이션을 시작한 후 운행속도가 표시될 때 추락 버튼을 클릭하면 추락하지 않은 캐빈이 멈춘다.



- 추락하는 경우에 가끔 속도가 내려가기도 하지만, 대부분의 경우에 속도가 오히려 증가한다.

Test case #16

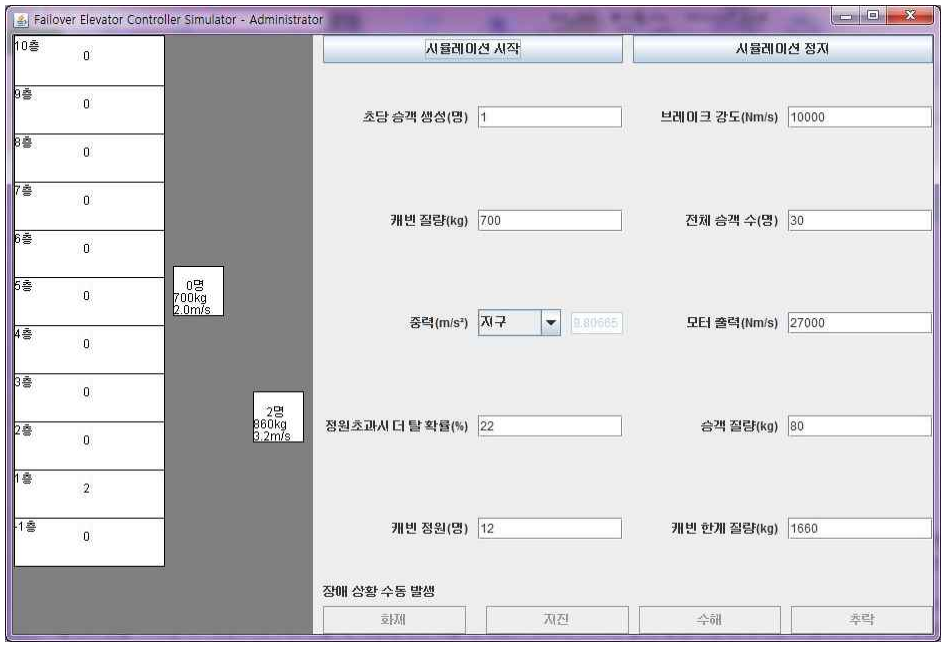
- 301.201.142.311.402 : 시뮬레이션을 시작한 후 캐빈무게가 표시될 때 캐빈한계무게를 현재총무게보다 크게 설정하면 화재버튼이 동작하여 **ev** 우선동작이 발생한다.



- EV가 **1**층과 화재 층에 우선동작하지 않고 **1**층은 우선동작 하지만 다른 층에 멈췄다가 화재가 난 층으로 간다.

Test case #25

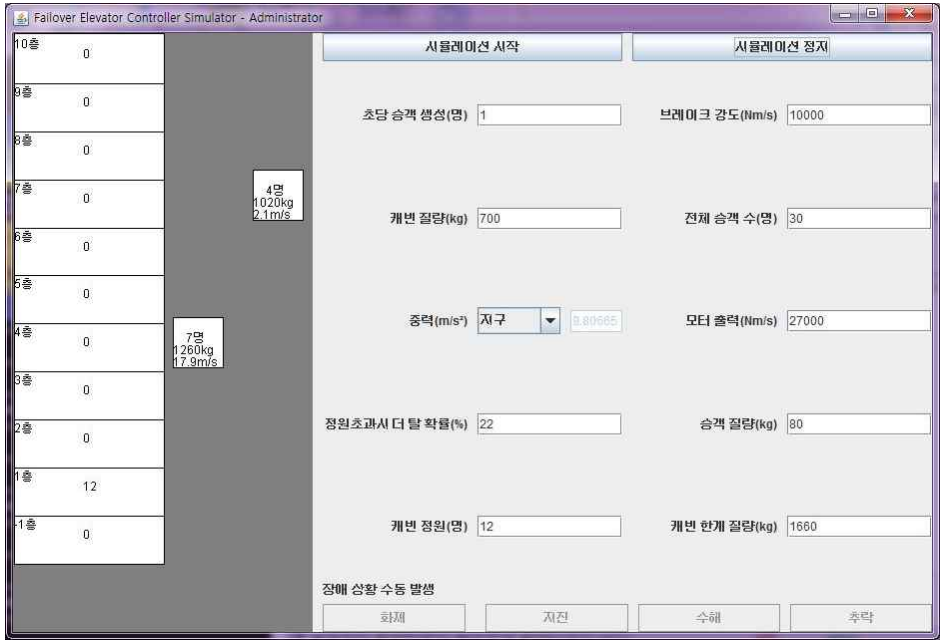
- 301.201.142.313.421 : 시뮬레이션을 시작한 후 캐빈무게가 표시될 때 캐빈한계무게를 현재총무게보다 크게 설정하면 지진버튼이 동작하여 랜덤하게 초당 한사람씩 사망한다.



- 아무도 죽지 않아 사람 수에 변화가 없다.

Test case #30

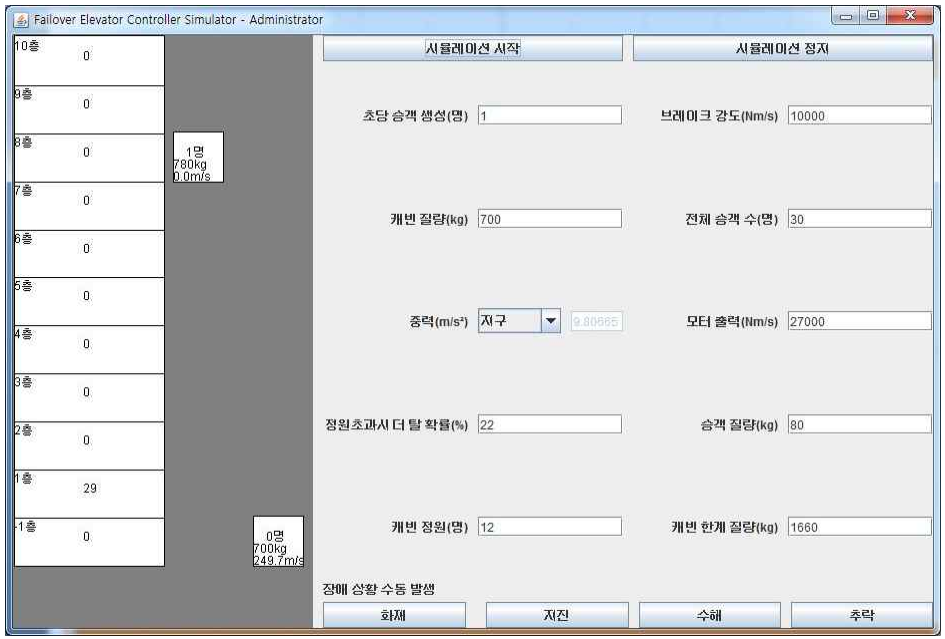
- 301.201.142.314.432 : 시뮬레이션을 시작한 후 캐빈무게가 표시될 때 캐빈한계무게를 현재총무게보다 크게 설정하면 추락버튼이 동작하여 추락하지 않은 캐빈이 멈춘다.



- 추락하는 경우에 가끔 속도가 내려가기도 하지만, 대부분의 경우에 속도가 오히려 증가한다.

Test case #36

- 어떠한 내/외적 상황에서도 시스템이 정지되지 않는다.



- 시작하자마자 아무 조작을 가하지 않았음에도 불구하고 한쪽 캐빈이 추락하여 속도가 계속 늘어나는 상황이 발생.

Test case #40

- 정원 초과시 탑승할 확률이 **0**초과 **1**이하의 값이 아니면 에러를 발생시킨다.

- **spec**에 명시하지 않은 상태로 **0**이상 **100**이하로 범위가 더 크게 바뀌었다.
엄밀하게 보면 **Fail.** 확률임을 감안하여 일단은 통과시켰다.

Test case #41

- 중력 설정에 입력한 값이 **0**초과 **50**이하의 정수가 아니면 에러를 발생시킨다.

- 대부분의 입력 오류의 경우 바로 직전의 정상입력 값으로 범위가 돌아가나 중력의 경우에는 직전의 입력 값으로 돌아가지는 않는다.

03 Testlink

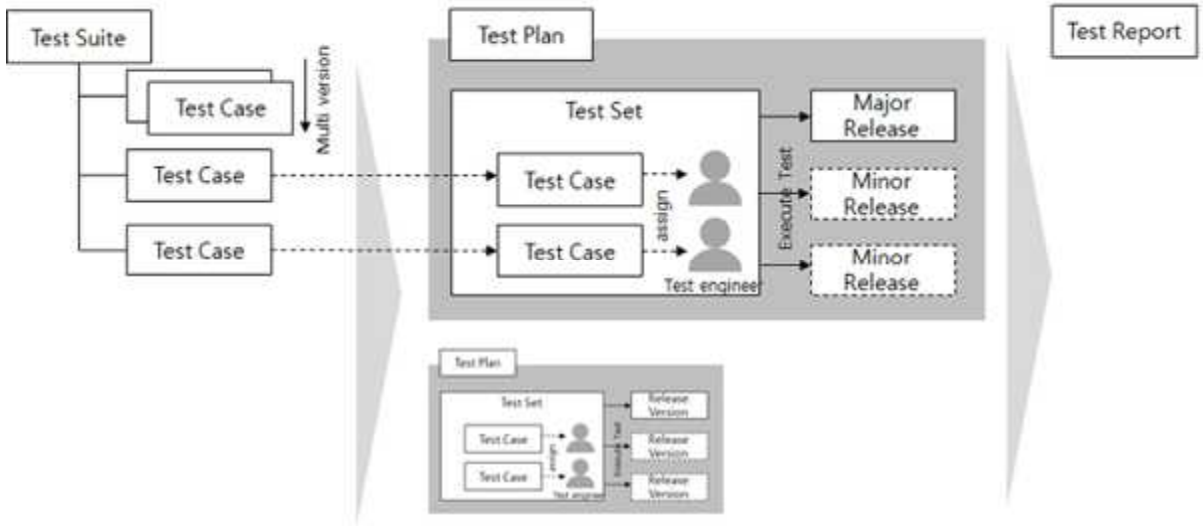
What is testlink?

- Testlink is a web-based test management system that facilitates software quality assurance.

- Testlink offers support for not only test cases, but also test suites, test plans, and test projects.

- If you want to use this, you can use MySQL and PostgreSQL as databases.

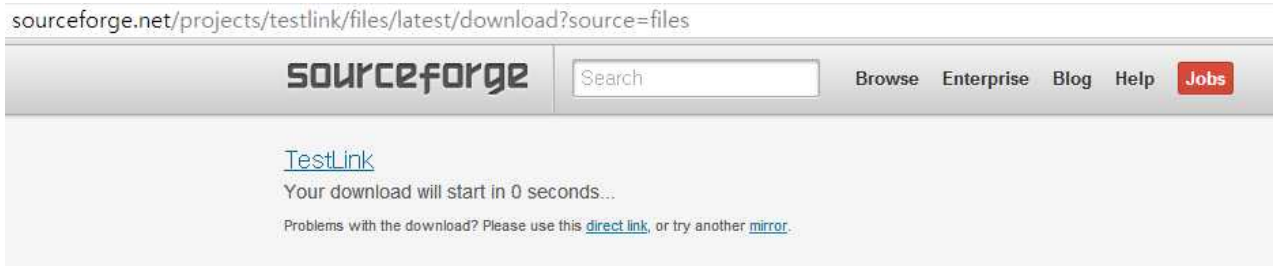
Test cases, suites, plans, projects



How to install testlink

- http://www.apmsetup.com/download.php

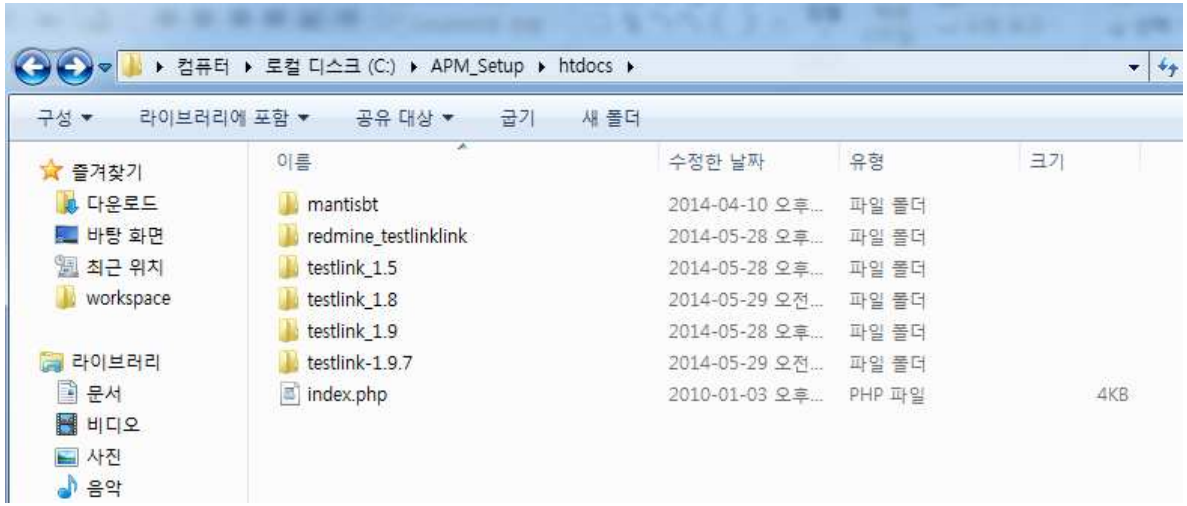
How to install testlink



-http://sourceforge.net/projects/testlink/files/latest/download?source=files

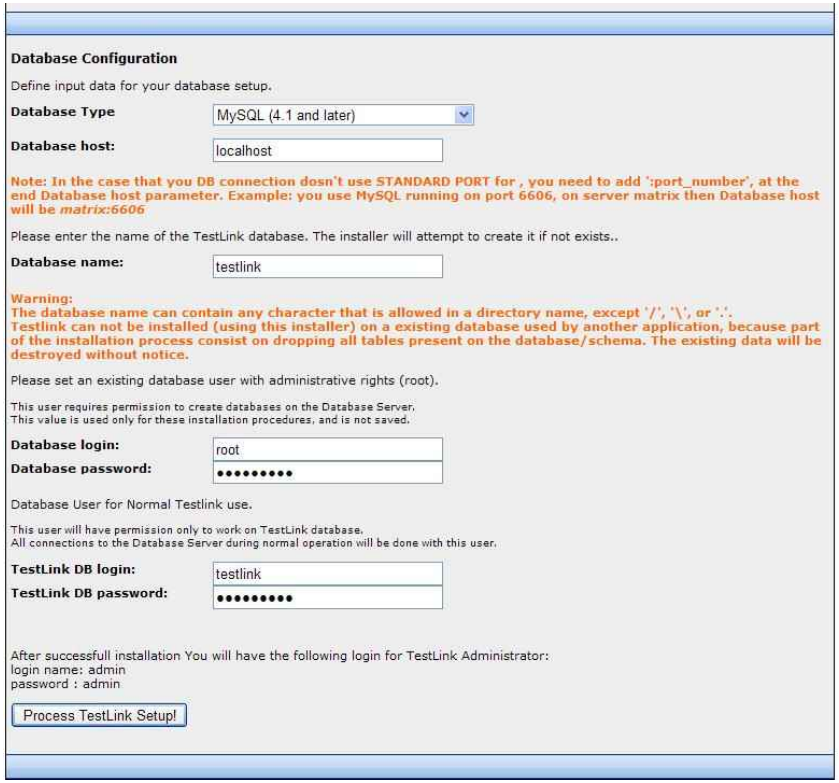
-Download 1.9.7 version!!!!!

How to install testlink



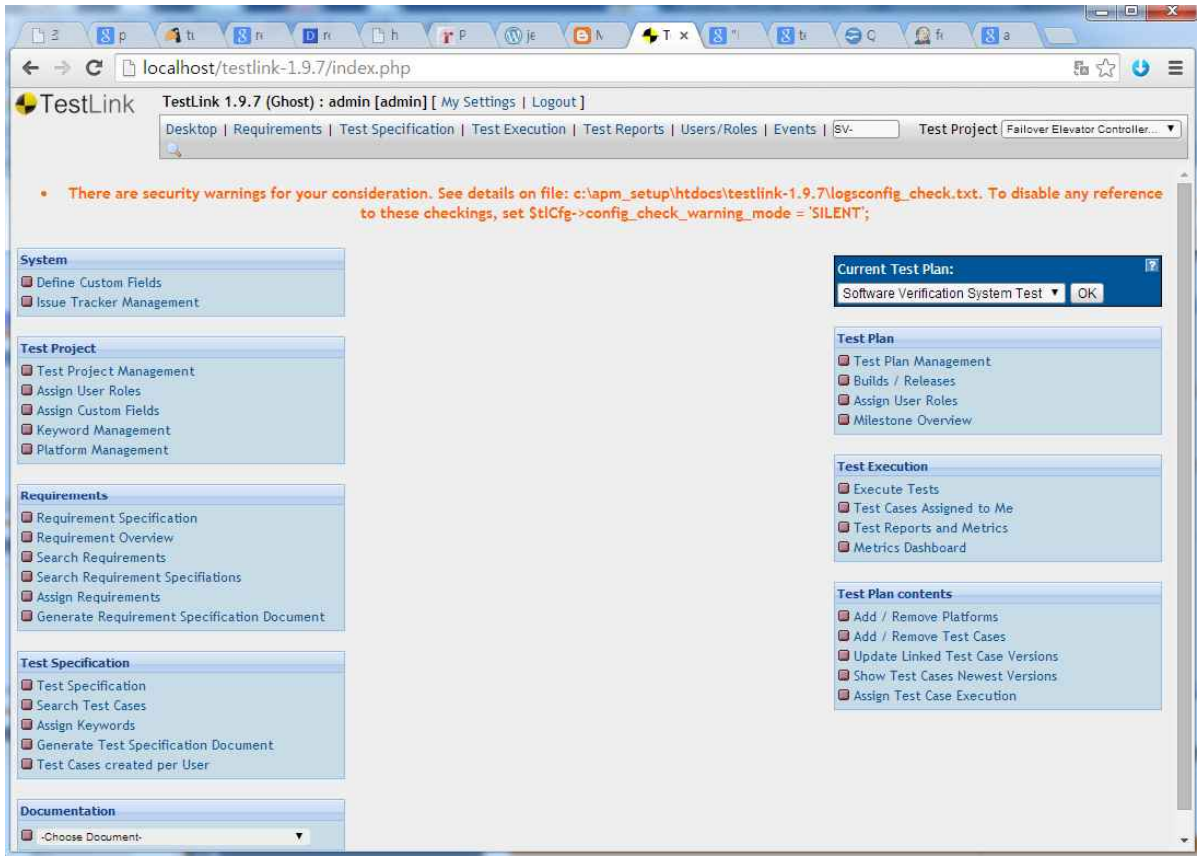
- Unzip the file and locate it in C:\APM_Setup\htdocs
- Rename the folder to ‘testlink’

How to install testlink



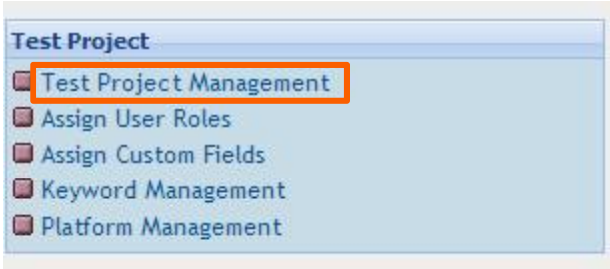
- http://localhost/testlink/install/index.php

How to install testlink



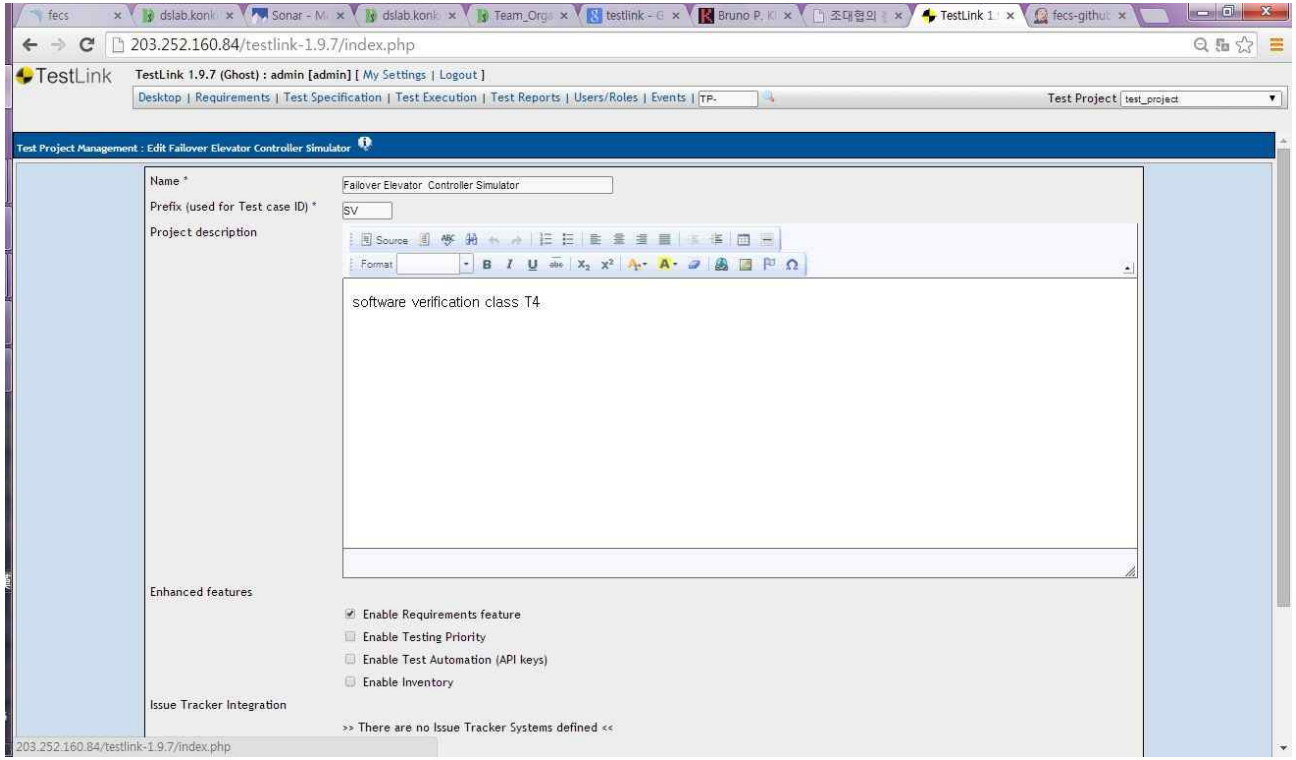
- http://localhost/testlink/index.php
- admin/admin

How to use testlink



- First, you have to make the test project.
- [Test project]-[Test project management]-[create]

How to use testlink



- You have to fill the Prefix…

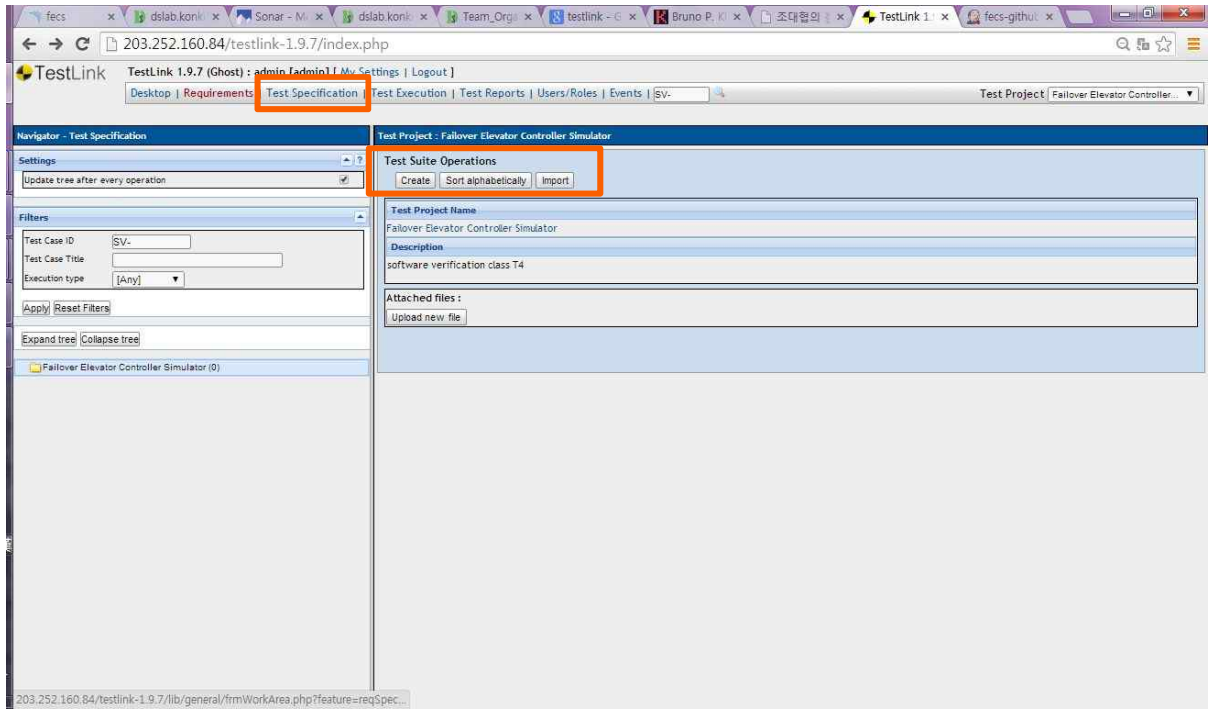
How to use testlink

01 jFeature

02 System Testing

03 Testlink

04 Static Analysis



- [Test Specification]-[Test suite operations]-[create]

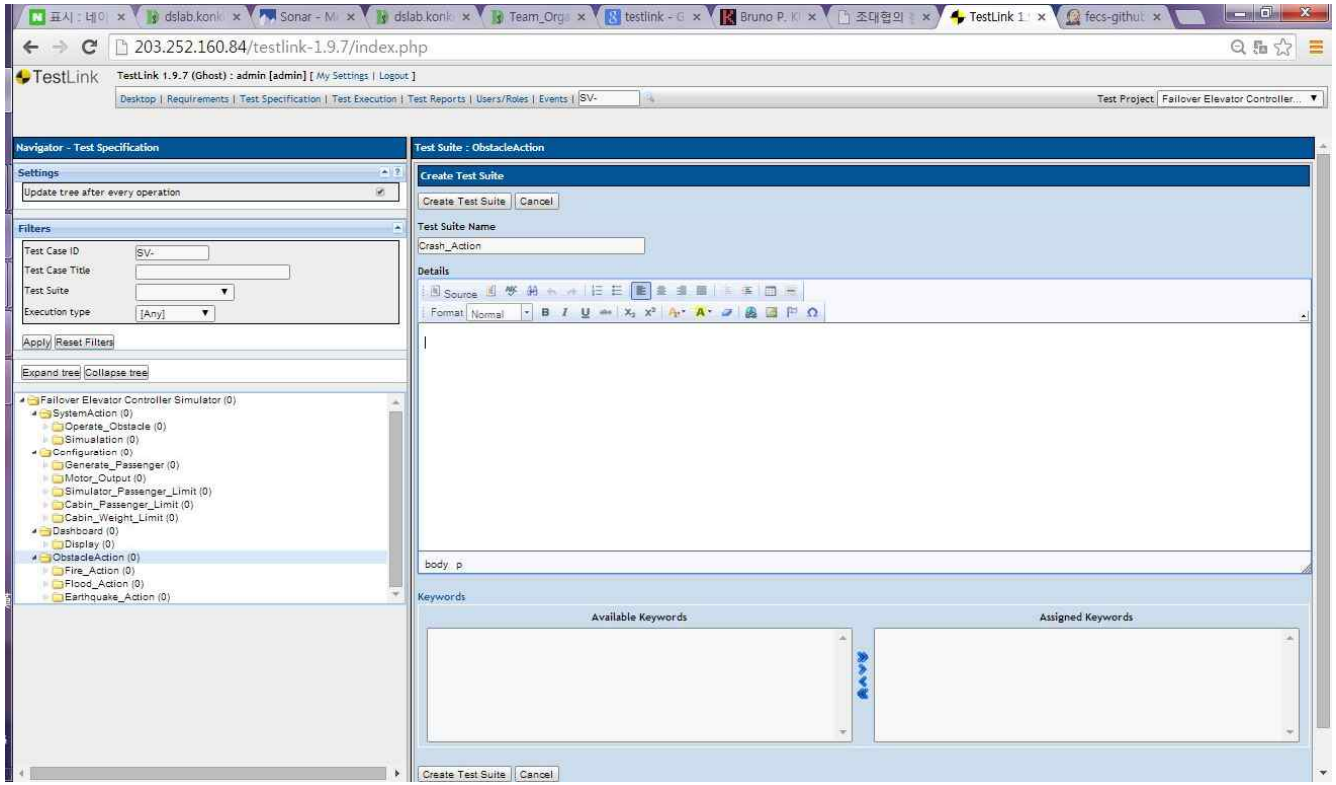
How to use testlink

01 jFeature

02 System Testing

03 Testlink

04 Static Analysis



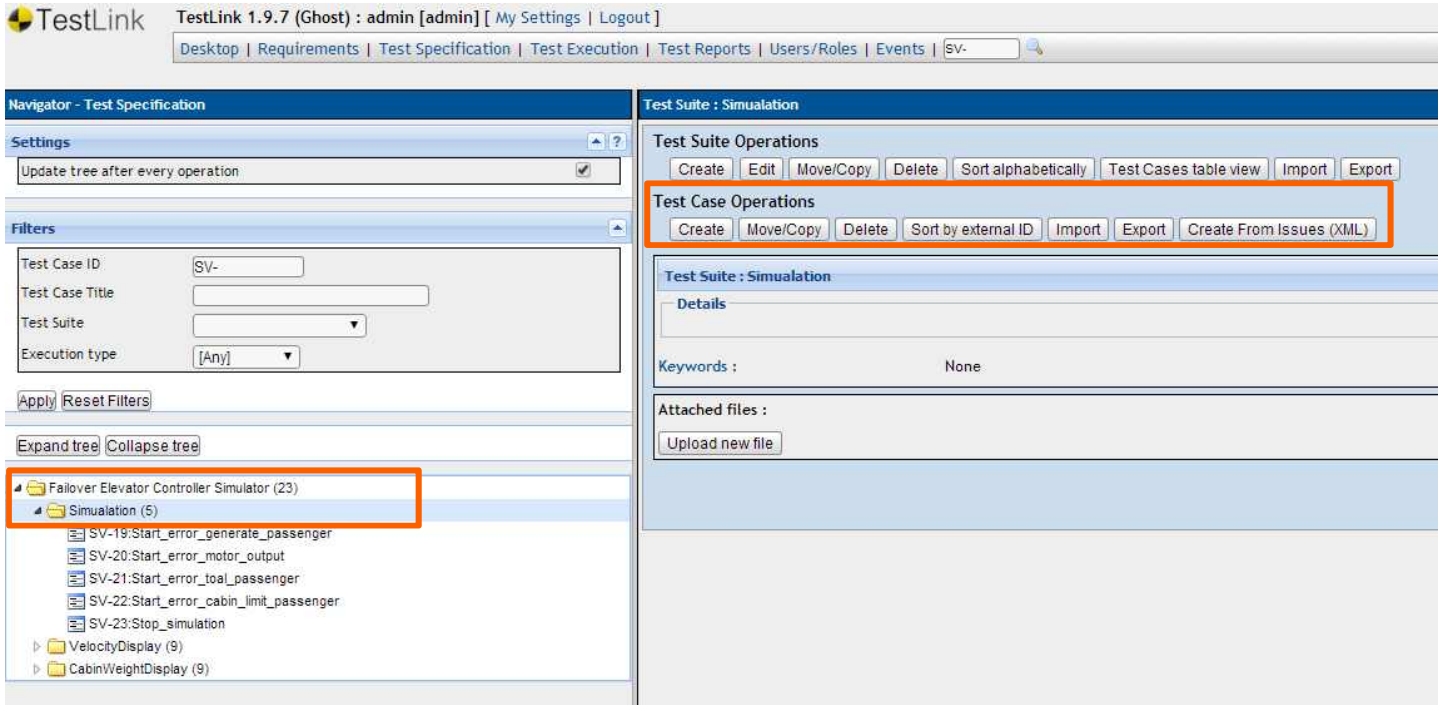
How to use testlink

01 jFeature

02 System Testing

03 Testlink

04 Static Analysis



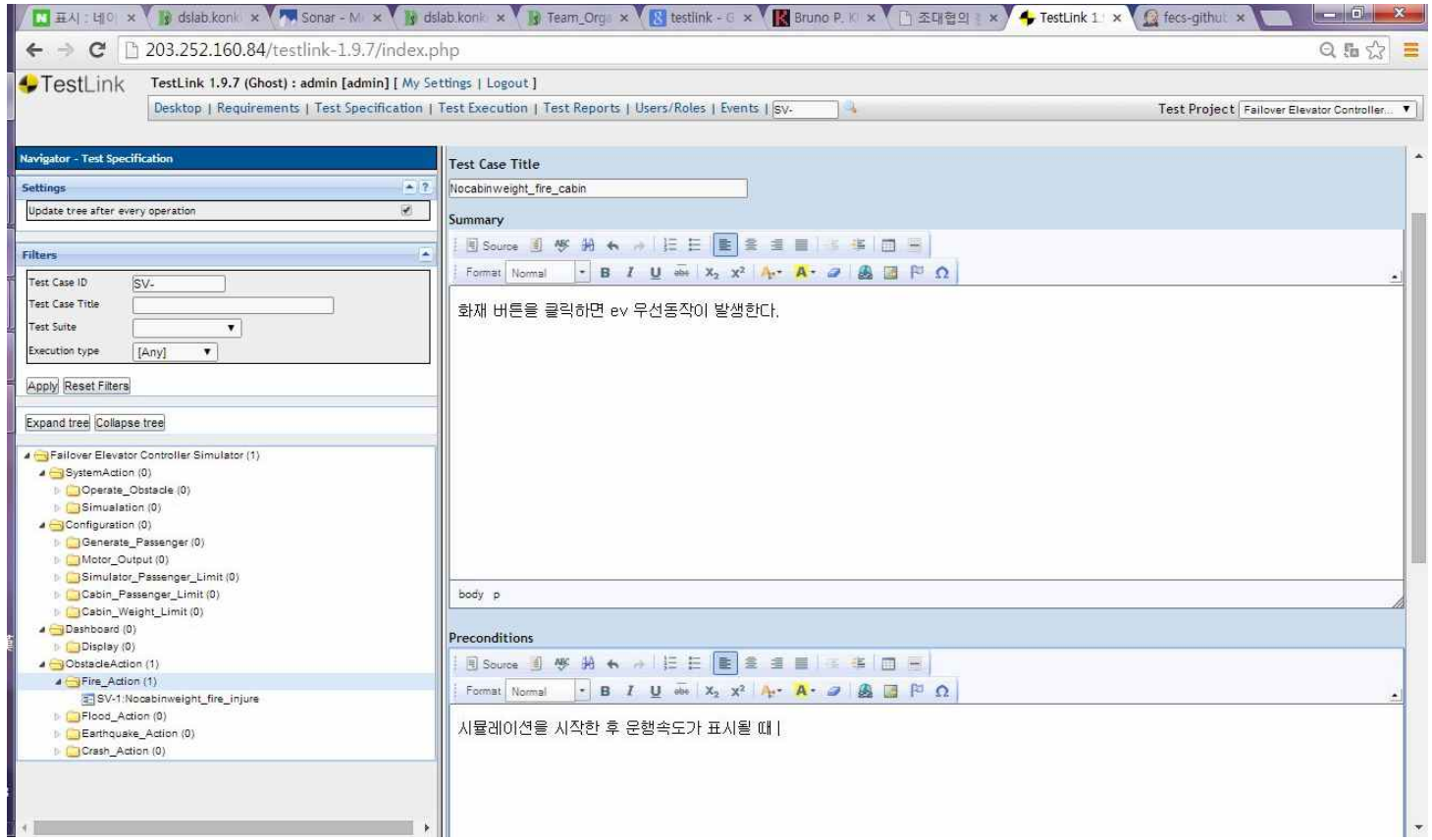
How to use testlink

01 jFeature

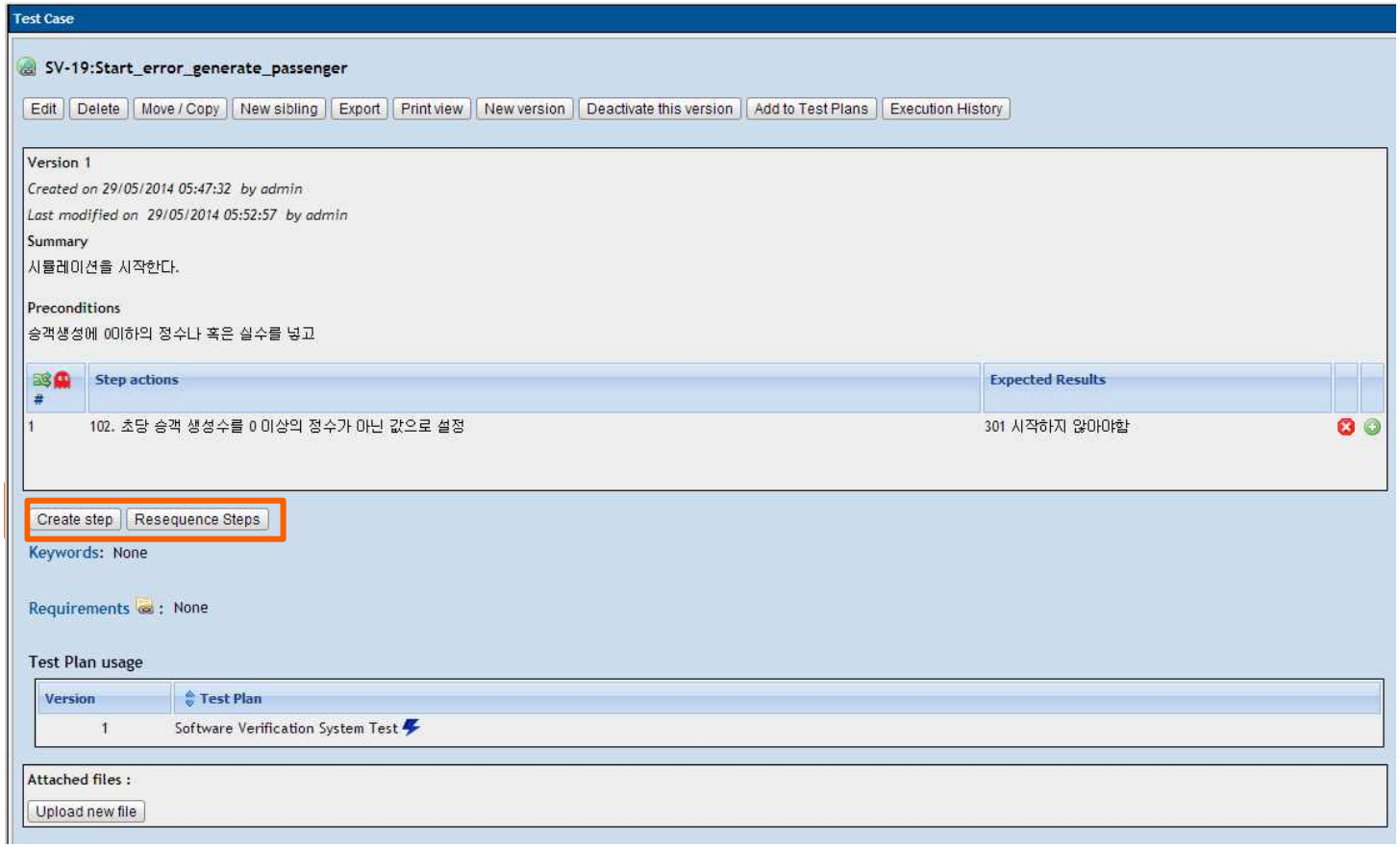
02 System Testing

03 Testlink

04 Static Analysis



How to use testlink



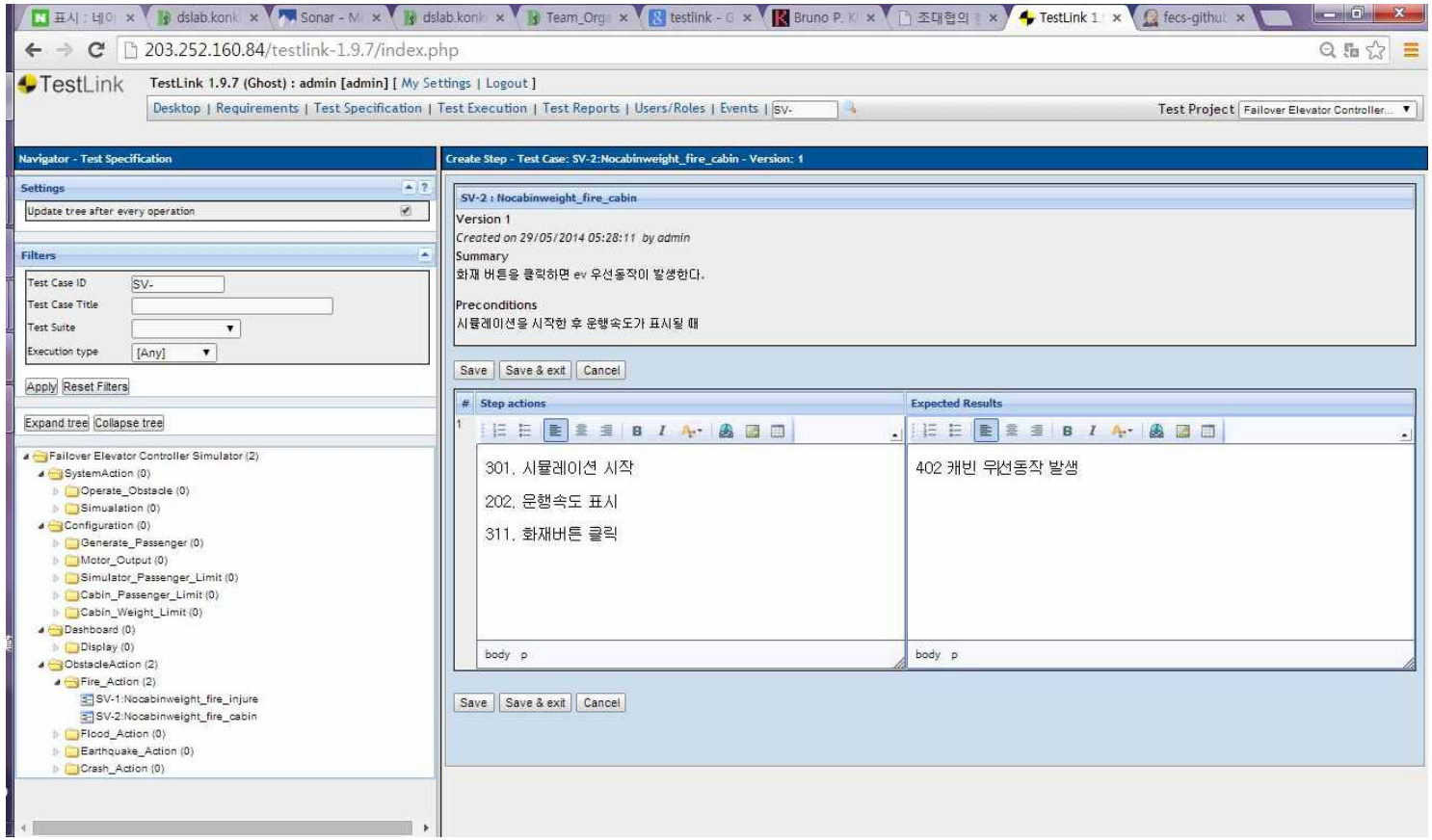
How to use testlink

01 jFeature

02 System Testing

03 Testlink

04 Static Analysis



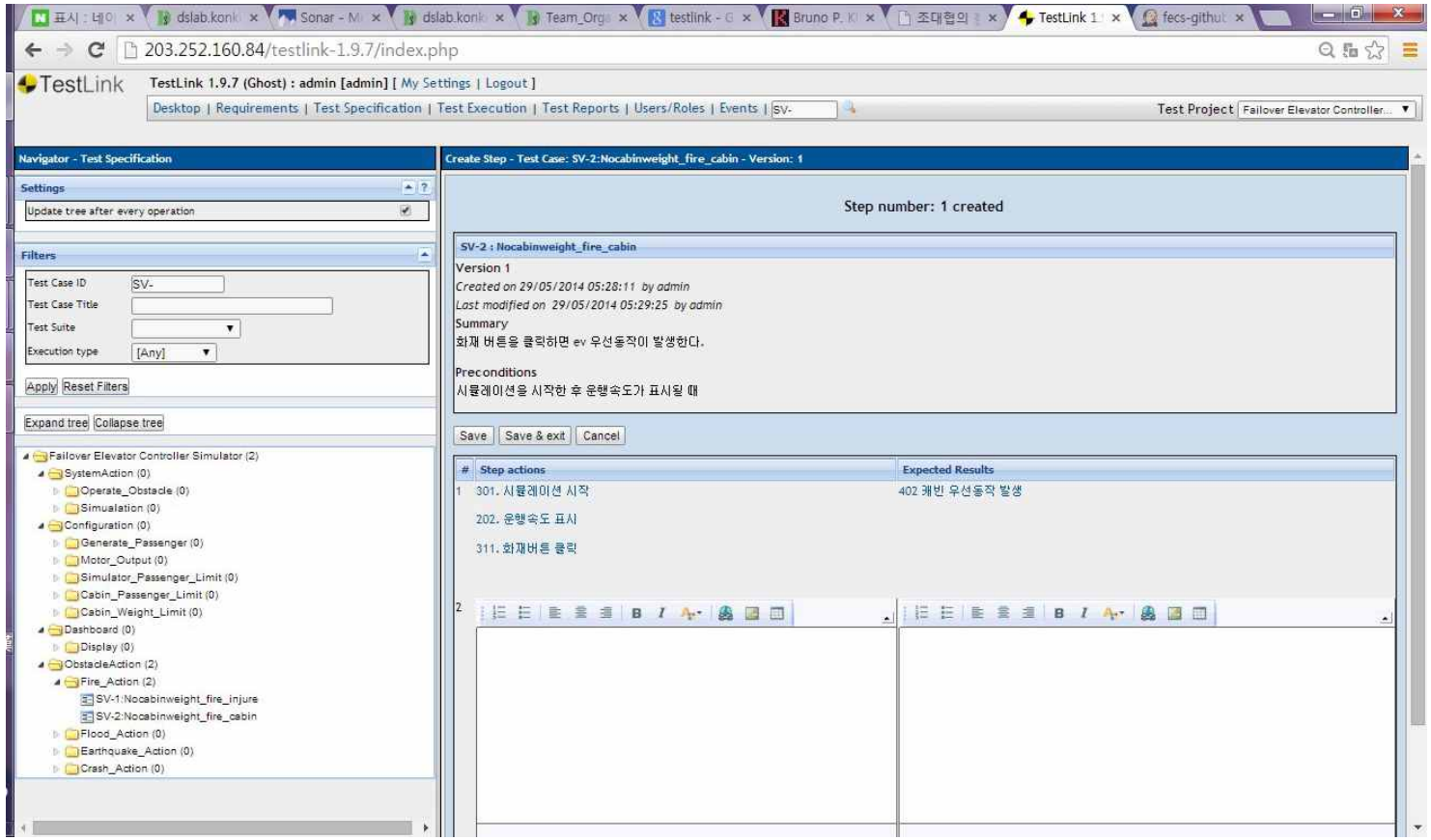
How to use testlink

01 jFeature

02 System Testing

03 Testlink

04 Static Analysis



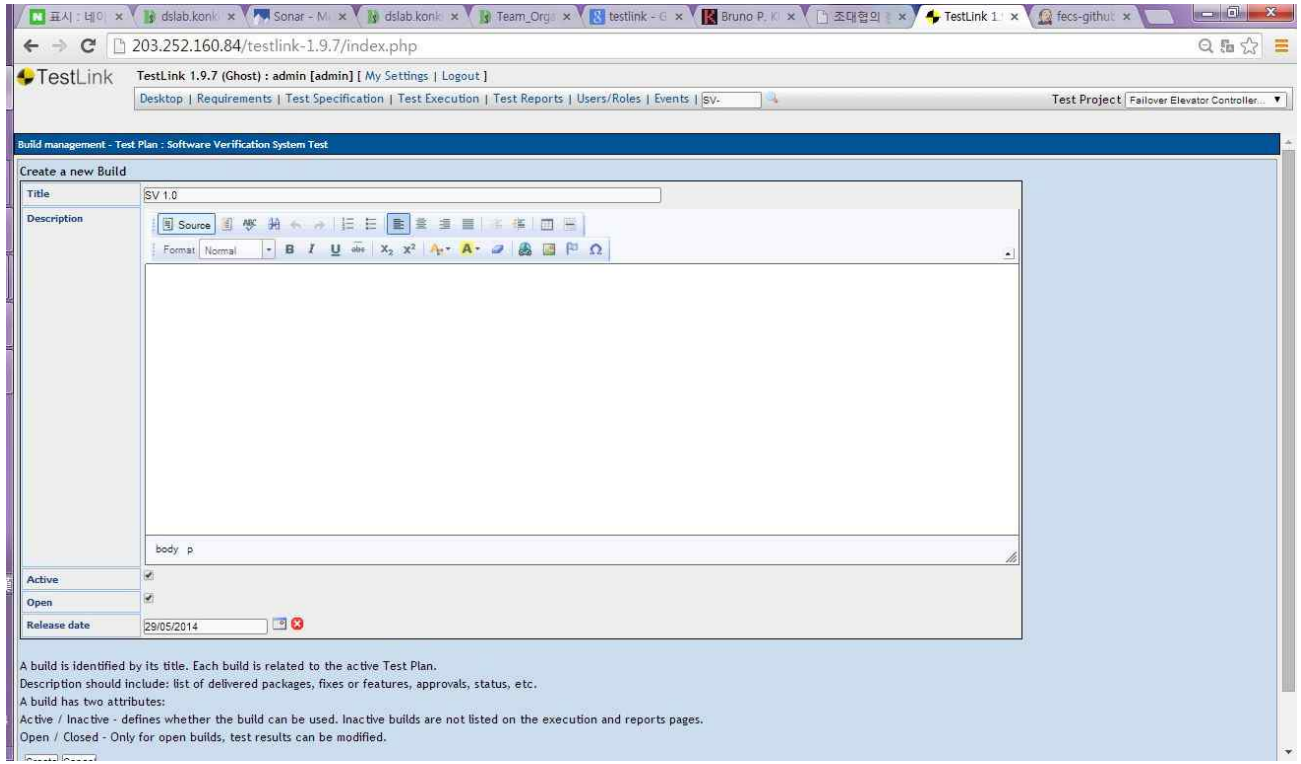
How to use testlink

01 jFeature

02 System Testing

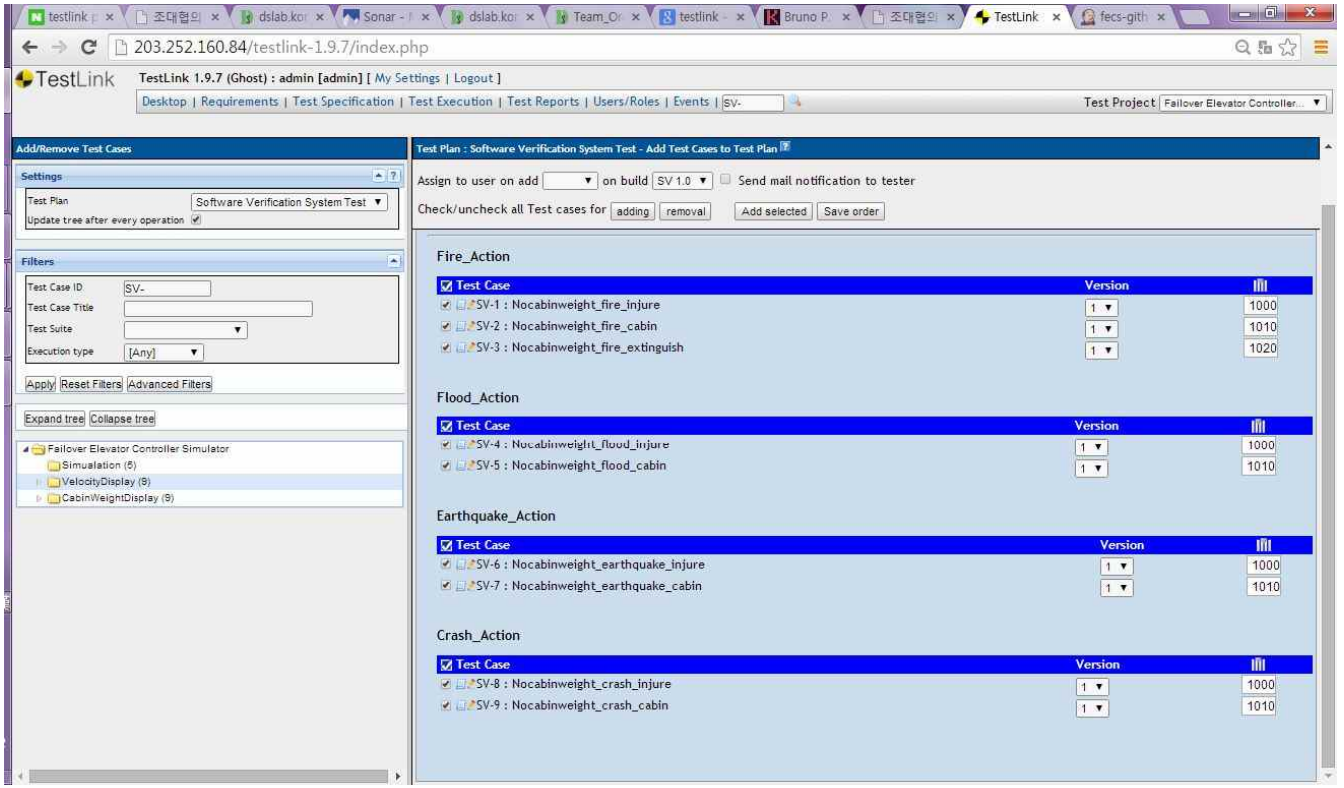
03 Testlink

04 Static Analysis



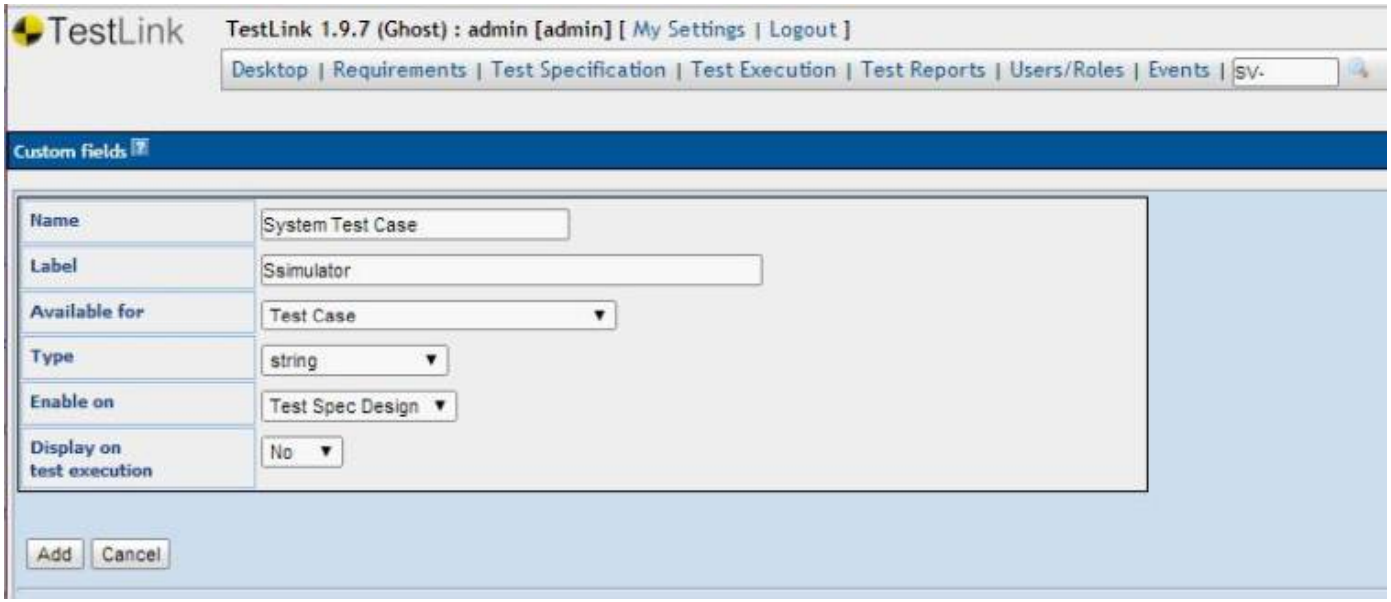
- [Build/releases]-[create]

How to use testlink



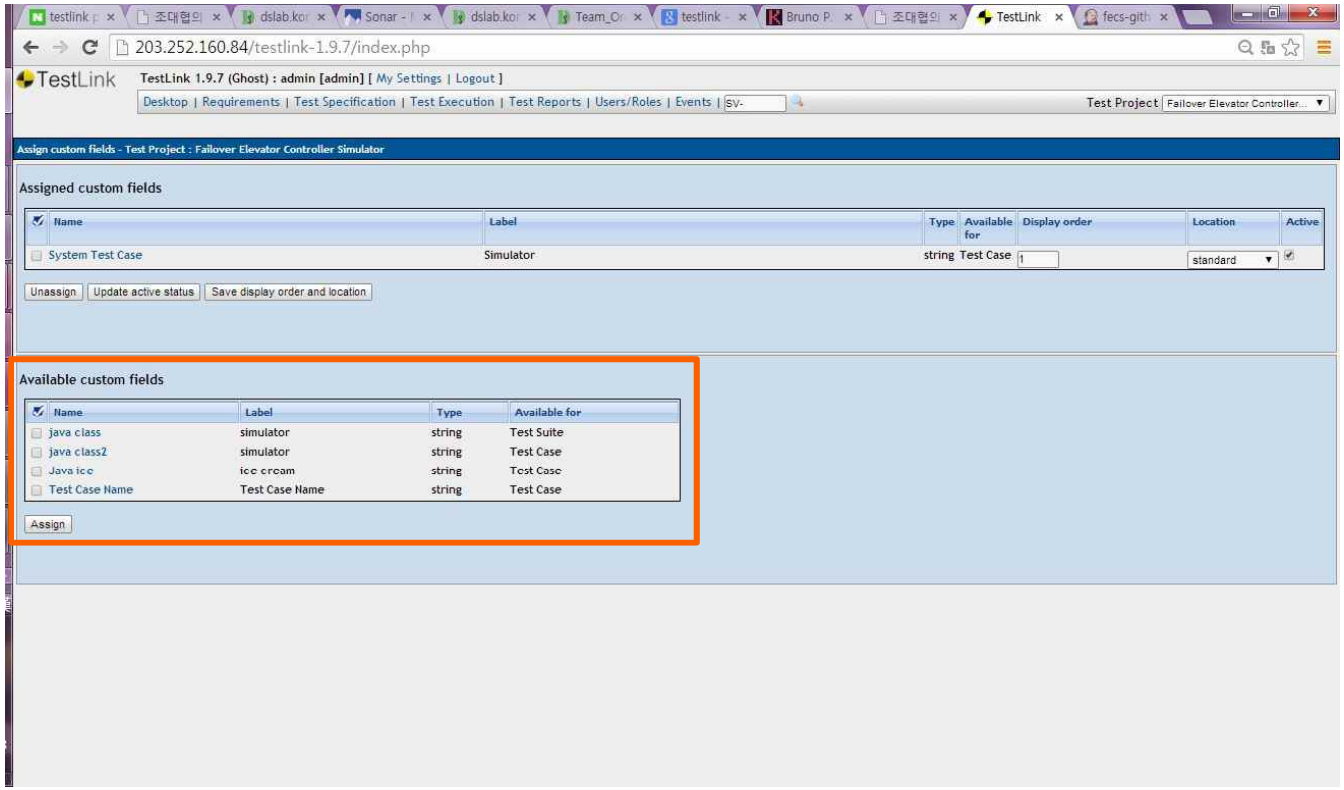
- [Test plan contents]-[add/remove test cases]

How to use testlink



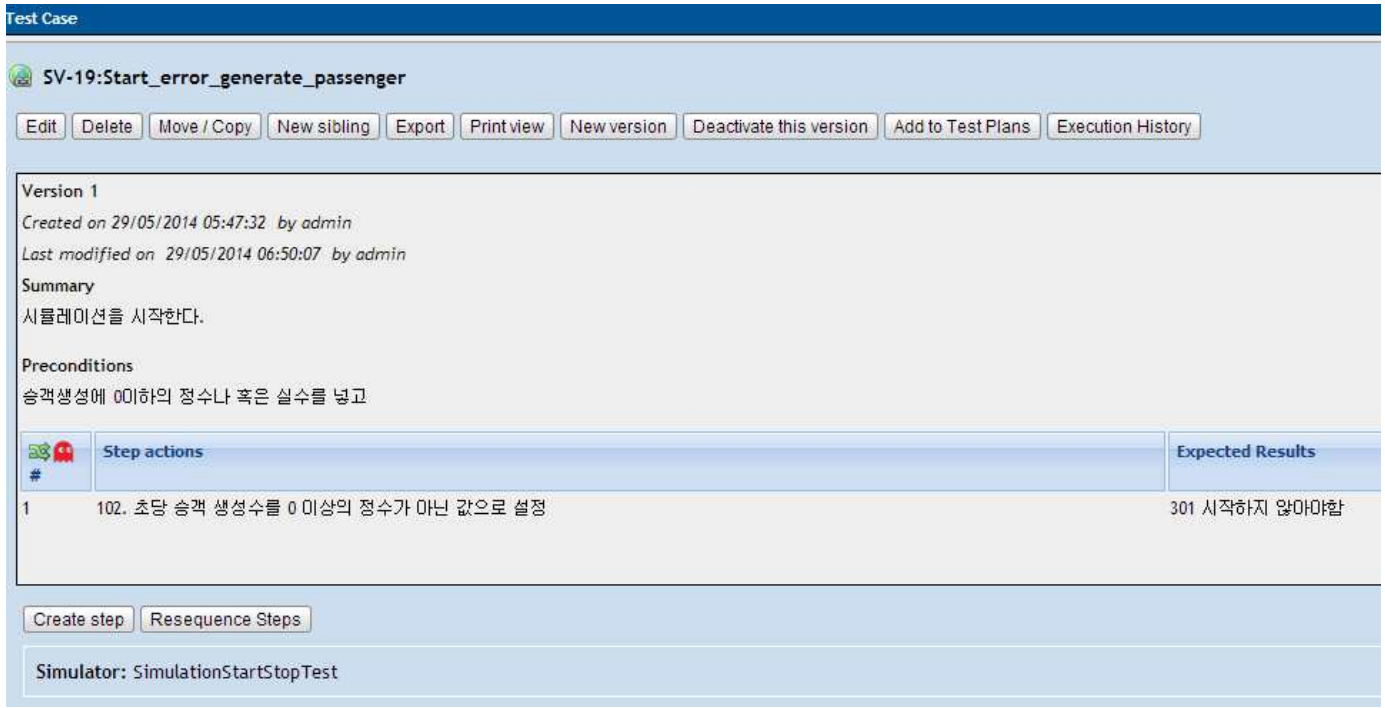
- [System]-[define custom field]

How to use testlink

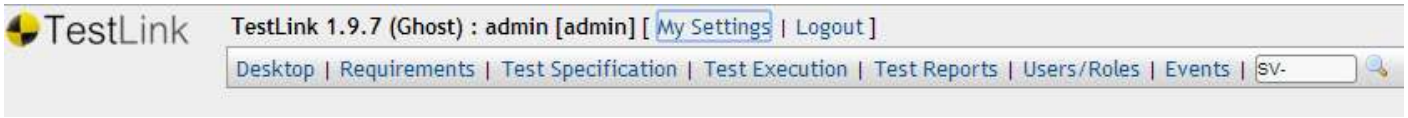


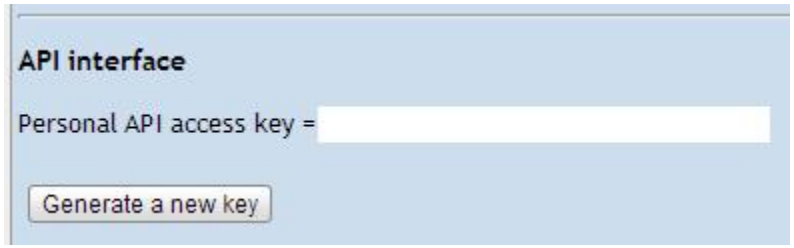
- [Test project]-[Assign user roles]

How to use testlink



How to integrate with Jenkins





- [My settings]-[Generate a new key]
- Install testlink plugin in jenkins

How to integrate with Jenkins

01 jFeature

02 System Testing

03 Testlink

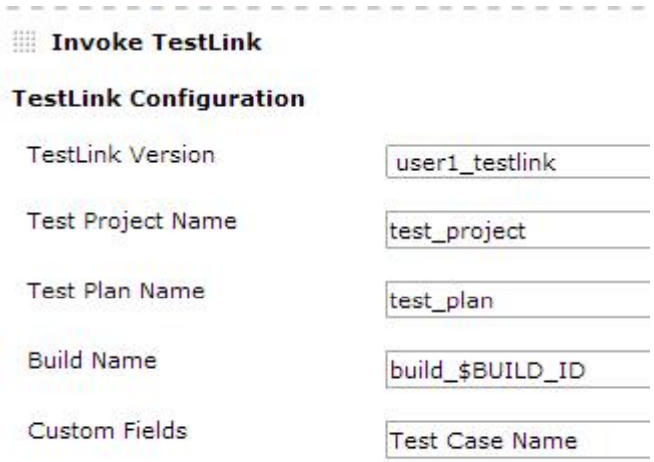
04 Static Analysis

- [Test project management]-[Test project]-check all of these
- [Test specification]-[Test case edit]-[execution type]

How to integrate with Jenkins

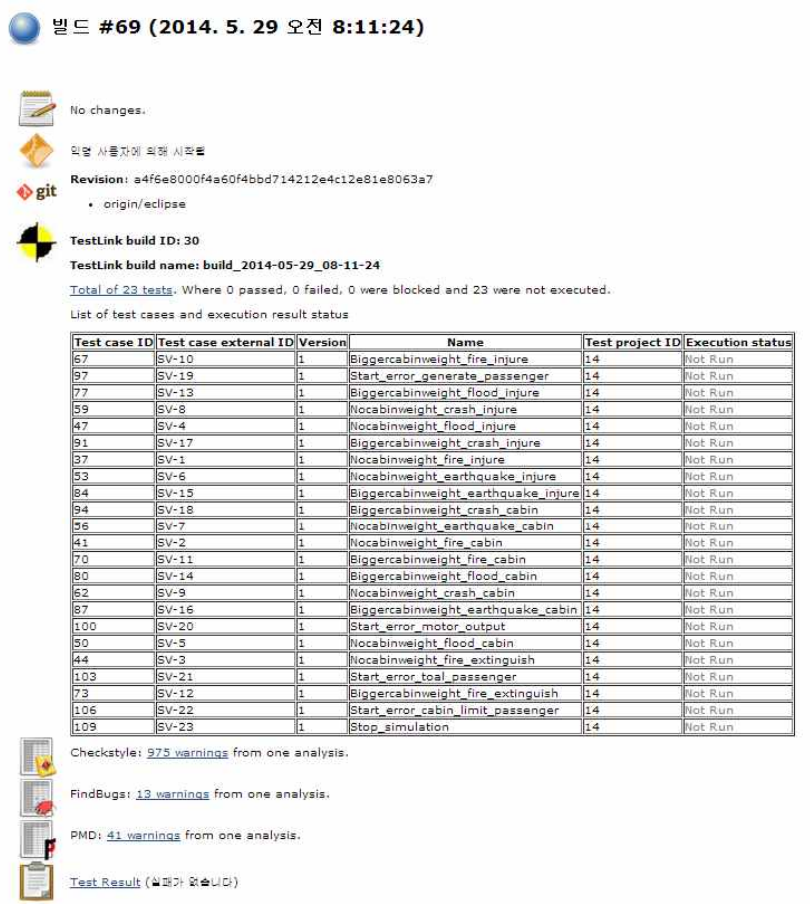
- **[jenkins 관리]-[시스템 설정]**

How to integrate with Jenkins



- [프로젝트 관리]-[Add build step]-[invoke testlink]

How to integrate with Jenkins



빌드 #69 (2014. 5. 29 오전 8:11:24)

No changes.

익명 사용자에 의해 시작됨

Revision: a4f6e8000f4a60f4bbd714212e4c12e81e8063a7

- origin/eclipse

TestLink build ID: 30

TestLink build name: build_2014-05-29_08-11-24

Total of 23 tests. Where 0 passed, 0 failed, 0 were blocked and 23 were not executed.

List of test cases and execution result status

Test case ID	Test case external ID	Version	Name	Test project ID	Execution status
67	SV-10	1	Biggercabinweight_fire_injure	14	Not Run
97	SV-19	1	Start_error_generate_passenger	14	Not Run
77	SV-13	1	Biggercabinweight_flood_injure	14	Not Run
59	SV-8	1	Nocabinweight_crash_injure	14	Not Run
47	SV-4	1	Nocabinweight_flood_injure	14	Not Run
91	SV-17	1	Biggercabinweight_crash_injure	14	Not Run
37	SV-1	1	Nocabinweight_fire_injure	14	Not Run
53	SV-6	1	Nocabinweight_earthquake_injure	14	Not Run
84	SV-15	1	Biggercabinweight_earthquake_injure	14	Not Run
94	SV-18	1	Biggercabinweight_crash_cabin	14	Not Run
56	SV-7	1	Nocabinweight_earthquake_cabin	14	Not Run
41	SV-2	1	Nocabinweight_fire_cabin	14	Not Run
70	SV-11	1	Biggercabinweight_fire_cabin	14	Not Run
80	SV-14	1	Biggercabinweight_flood_cabin	14	Not Run
62	SV-9	1	Nocabinweight_crash_cabin	14	Not Run
87	SV-16	1	Biggercabinweight_earthquake_cabin	14	Not Run
100	SV-20	1	Start_error_motor_output	14	Not Run
50	SV-5	1	Nocabinweight_flood_cabin	14	Not Run
44	SV-3	1	Nocabinweight_fire_extinguish	14	Not Run
103	SV-21	1	Start_error_toal_passenger	14	Not Run
73	SV-12	1	Biggercabinweight_fire_extinguish	14	Not Run
106	SV-22	1	Start_error_cabin_limit_passenger	14	Not Run
109	SV-23	1	Stop_simulation	14	Not Run

Checkstyle: 975 warnings from one analysis.

FindBugs: 13 warnings from one analysis.

PMD: 41 warnings from one analysis.

Test Result (실패가 없습니다)

- You can see the results.

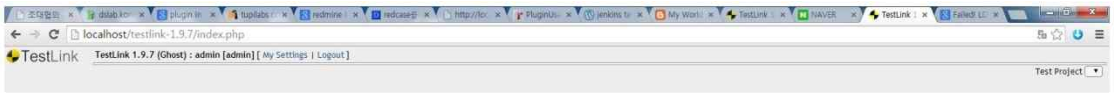
Trouble Shooting

01 jFeature

02 System Testing

03 Testlink

04 Static Analysis



Trouble Shooting


```
function getTypes()
{
  if( is_null($this->types) )
  {
    foreach($this->systems as $code => $spec)
    {
      $this->types[$code] = $spec['type'] . " (Interface: {$spec['api']})";
    }
  }
  return $this->types;
}
```

- Replace :: to ->

Trouble Shooting

01 jFeature

02 System Testing

03 Testlink

04 Static Analysis

```
Found 3 automated test cases in TestLink.
Sorting automated test cases by TestLink test plan execution order.
Executing single Build Steps.
Executing iterative Build Steps.
Looking for the test results of TestLink test cases.
Found 3 test result(s).
```

- I integrated with jenkins… but my tests were not integrated. It's has 'not run' state.

After using Testlink

- Testlink is quite good tool but hard to integrate with other tools.

- So many errors but there are no solutions cause this tool is made in japan.

- If I can use this well, it'll be strong tool.

04 Static Analysis

Static Analysis

- Static analysis is the analysis of computer software that is performed without actually executing programs. (⇔ Dynamic analysis)

- In most cases the analysis is performed on some version of the source code, and in the other cases, some form of the object code.

-You can identify and diagnose run-time errors such as overflows, divide by zero and then use the metrics produced by static analysis to measure and improve software quality.

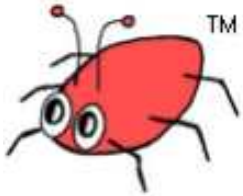
Static analysis tools that we used

01 jFeature

02 System Testing

03 Testlink

04 Static Analysis



Checkstyle

01 jFeature

02 System Testing

03 Testlink

04 Static Analysis

CheckStyle Result

Warnings Trend

All Warnings	New Warnings	Fixed Warnings
975	0	0

Summary

Total	High Priority	Normal Priority	Low Priority
975	975	0	0

Details

Packages | Files | Categories | Types | Warnings | Details

Package	Total	Distribution
fecs	103	
fecs.event	11	
fecs.interfaces	35	
fecs.model	38	
fecs.simulator	540	
fecs.ui	244	
src/main/java/fecs	4	
Total	**975**	

Packages | Files | **Categories** | Types | Warnings | Details

Category	Total
Blocks	24
Checks	86
Coding	104
Design	117
Imports	11
Javadoc	265
Modifier	12
Naming	9
Sizes	107
Whitespace	240
Total	**975**

Checkstyle – block category

Details

Packages | Files | **Types** | Warnings | Details

Type	Total
NeedBracesCheck	23
RightCurlyCheck	1
Total	**24**

Controller.java:64, NeedBracesCheck, Priority: High

'if' construct must use '{}'s.

```
else if (val < 1) val = 1.0;
```

Checks for braces around code blocks.

Controller.java:66, RightCurlyCheck, Priority: High

'}' should be on the same line.

```
try { val = Integer.parseInt(answer); }
```

Checks the placement of right curly braces ('}') for else, try, and catch tokens. The policy to verify is specified using property option.

Checkstyle – check category

Details

Packages | Files | **Types** | Warnings | Details

Type	Total
FinalParametersCheck	84
NewlineAtEndOfFileCheck	2
Total	**86**

Circumstance.java:49, **FinalParametersCheck, Priority: High**

```
public ElevatorCall(Passenger source, Vector pushed) {
```

Parameter name should be final.

Check that method/constructor/catch block parameters are final. Interface and abstract methods are not checked - the final keyword does not make sense for interface and abstract method parameters as there is no code that could modify the parameter.

Rationale: Changing the value of parameters during the execution of the method's algorithm can be confusing and should be avoided. A great way to let the Java compiler prevent this coding style is to declare parameters final.

Fecs.java:0, **NewlineAtEndOfFileCheck, Priority: High**

File does not end with a newline.

Checks whether files end with a new line.

Rationale: Any source files and text files in general should end with a newline character, especially when using SCM systems such as CVS. CVS will even print a warning when it encounters a file that doesn't end with a newline.

Checkstyle – coding category

01 jFeature

02 System Testing

03 Testlink

04 Static Analysis

Details

Packages | Files | **Types** | Warnings | Details

Type
AvoidInlineConditionalsCheck
EqualsHashCodeCheck
HiddenFieldCheck
MagicNumberCheck
MissingSwitchDefaultCheck
Total

CabinsController.java:17, **AvoidInlineConditionalsCheck, Priority: High**

Avoid inline conditionals.

```
cabin = Math.abs(dy_l) <= Math.abs(dy_r) ? left : right;
```

Detects inline conditionals. An example inline conditional is this:

String a = getParameter("a"); String b = (a==null || a.length<1) ? null : a.substring(1);

Rationale: Some developers find inline conditionals hard to read, so their company's coding standards forbids them.

Passenger.java:47, **EqualsHashCodeCheck, Priority: High**

Definition of 'equals()' without corresponding definition of 'hashCode()'.

```
@Override
public boolean equals(Object o) {
```

Checks that classes that override equals() also override hashCode().

Rationale: The contract of equals() and hashCode() requires that equal objects have the same hashCode. Hence, whenever you override equals() you must override hashCode() to ensure that your class can be used in collections that are hash based.

Passenger.java:26, **HiddenFieldCheck, Priority: High**

```
public void setStart(Integer start) {
    this.start = start;
```

'start' hides a field.

Checks that a local variable or a parameter does not shadow a field that is defined in the same class.

Checkstyle – coding category

Details

Packages | Files | **Types** | Warnings | Details

Type
AvoidInlineConditionalsCheck
EqualsHashCodeCheck
HiddenFieldCheck
MagicNumberCheck
MissingSwitchDefaultCheck
Total

PassengerMaker.java:19, MagicNumberCheck, Priority: High

```
private Integer now = 0, howMany = 1, max = 30;
```

'30' is a magic number.

Checks that there are no "magic numbers", where a magic number is a numeric literal that is not defined as a constant. By default, -1, 0, 1, and 2 are not considered to be magic numbers.

Engine.java:186, MissingSwitchDefaultCheck, Priority: High

switch without "default" clause.

Checks that switch statement has "default" clause.

Rationale: It's usually a good idea to introduce a default case in every switch statement. Even if the developer is sure that all currently possible cases are covered, this should be expressed in the default branch, e.g. by using an assertion. This way the code is protected aginst later changes, e.g. introduction of new types in an enumeration type.

Checkstyle – design category

Details

Packages | Files | **Types** | Warnings | Details

Type	Total
DesignForExtensionCheck	110
HideUtilityClassConstructorCheck	1
VisibilityModifierCheck	6
Total	**117**

Circumstance.java:34, **DesignForExtensionCheck, Priority: High**

Method 'trigger' is not designed for extension - needs to be abstract, final or empty.

Checks that classes are designed for extension. More specifically, it enforces a programming style where superclasses provide empty "hooks" that can be implemented by subclasses.

The exact rule is that nonprivate, nonstatic methods of classes that can be subclassed must either be

- abstract or
- final or
- have an empty implementation

Rationale: This API design style protects superclasses against beeing broken by subclasses. The downside is that subclasses are limited in their flexibility, in particular they cannot prevent execution of code in the superclass, but that also means that subclasses cannot corrupt the state of the superclass by forgetting to call the super method.

```
public class CabinsController {
```

CabinsController.java:10, **HideUtilityClassConstructorCheck, Priority: High**

```
public static void control(Cabin left, Cabin right, Floor floor) {
```

Utility classes should not have a public or default constructor.

Make sure that utility classes (classes that contain only static methods or fields in their API) do not have a public constructor.

Rationale: Instantiating utility classes does not make sense. Hence the constructors should either be private or (if you want to allow subclassing) protected. A common mistake is forgetting to hide the default constructor.

If you make the constructor protected you may want to consider the following constructor implementation technique to disallow instantiating subclasses:

public class StringUtils // not final to allow subclassing { protected StringUtils() { // prevents calls from subclass throw new UnsupportedOperationException(); } public static int count(char c, String s) { // ... } }

Circumstance.java:26, **VisibilityModifierCheck, Priority: High**

```
protected String name;
```

Variable 'name' must be private and have accessor methods.

Checks visibility of class members. Only static final members may be public; other class members must be private unless property protectedAllowed or packageAllowed is set.

Public members are not flagged if the name matches the public member regular expression (contains "^serialVersionUID$" by default). Note: Checkstyle 2 used to include "^f[A-Z][a-zA-Z0-9]+$" in the default pattern to allow CMP for EJB 1.1 with the default settings. With EJB 2.0 it is not longer necessary to have public access for persistent fields, hence the default has been changed.

Rationale: Enforce encapsulation.

Checkstyle – imports category

Details

Packages | Files | **Types** | Warnings | Details

Type	Total
AvoidStarImportCheck	8
UnusedImportsCheck	3
Total	**11**

Cabin.java:7, **AvoidStarImportCheck, Priority: High**

```
import java.util.*;
```

Using the '.*' form of import should be avoided - java.util.*.

Checks that there are no import statements that use the * notation.

Rationale: Importing all classes from a package or static members from a class leads to tight coupling between packages or classes and might lead to problems when a new version of a library introduces name clashes.

CabinsController.java:5, **UnusedImportsCheck, Priority: High**

Unused import - fecs.model.Vector.

Checks for unused import statements. Checkstyle uses a simple but very reliable algorithm to report on unused import statements. An import statement is considered unused if:

- It is not referenced in the file. The algorithm does not support wild-card imports like `import java.io.*;`. Most IDE's provide very sophisticated checks for imports that handle wild-card imports.
- It is a duplicate of another import. This is when a class is imported more than once.
- The class imported is from the `java.lang` package. For example importing `java.lang.String`.
- The class imported is from the same package.
- **Optionally:** it is referenced in Javadoc comments. This check is off by default, as it is considered bad practice to introduce a compile time dependency for documentation purposes only. As an example, the import `java.util.Date` would be considered referenced with the Javadoc comment `{@link Date}`. The alternative to avoid introducing a compile time dependency would be to write the Javadoc comment as `{@link java.util.Date}`.

The main limitation of this check is handling the case where an imported type has the same name as a declaration, such as a member variable.

For example, in the following case the import `java.awt.Component` will not be flagged as unused:

import java.awt.Component; class FooBar { private Object Component; // IMHO bad practise ... }

Checkstyle – javadoc category

01 jFeature

02 System Testing

03 Testlink

04 Static Analysis

Details

Packages | Files | **Types** | Warnings | Details

Type	Total
JavadocMethodCheck	141
JavadocPackageCheck	6
JavadocStyleCheck	1
JavadocTypeCheck	7
JavadocVariableCheck	110
Total	**265**

Circumstance.java:34, JavadocMethodCheck, Priority: High

Missing a Javadoc comment.

Checks the Javadoc of a method or constructor. By default, does not check for unused throws. To allow documented java.lang.RuntimeExceptions that are not declared, set property allowUndeclaredRTE to true. The scope to verify is specified using the Scope class and defaults to Scope.PRIVATE. To verify another scope, set property scope to a different scope.

Error messages about parameters and type parameters for which no param tags are present can be suppressed by defining property allowMissingParamTags. Error messages about exceptions which are declared to be thrown, but for which no throws tag is present can be suppressed by defining property allowMissingThrowsTags. Error messages about methods which return non-void but for which no return tag is present can be suppressed by defining property allowMissingReturnTag.

Javadoc is not required on a method that is tagged with the @Override annotation. However under Java 5 it is not possible to mark a method required for an interface (this was *corrected* under Java 6). Hence Checkstyle supports using the convention of using a single {@inheritDoc} tag instead of all the other tags.

Note that only inheritable items will allow the {@inheritDoc} tag to be used in place of comments. Static methods at all visibilities, private non-static methods and constructors are not inheritable.

For example, if the following method is implementing a method required by an interface, then the Javadoc could be done as:

```
/** {@inheritDoc} */
public int checkReturnTag(final int aTagIndex,
                          JavadocTag[] aTags,
                          int aLineNo)
```

The classpath may need to be configured to locate the class information. The classpath configuration is dependent on the mechanism used to invoke Checkstyle.

CabinsController.java:0, JavadocPackageCheck, Priority: High

Missing package-info.java file.

Checks that each Java package has a Javadoc file used for commenting. By default it only allows a package-info.java file, but can be configured to allow a package.html file.

An error will be reported if both files exist as this is not allowed by the Javadoc tool.

Checkstyle – javadoc category

Details

Packages | Files | **Types** | Warnings | Details

Type	Total
JavadocMethodCheck	141
JavadocPackageCheck	6
JavadocStyleCheck	1
JavadocTypeCheck	7
JavadocVariableCheck	110
Total	**265**

UserInterface.java:264, JavadocStyleCheck, Priority: High

Incomplete HTML tag found: * >>> IMPORTANT!! <<<

Validates Javadoc comments to help ensure they are well formed. The following checks are performed:

- Ensures the first sentence ends with proper punctuation (That is a period, question mark, or exclamation mark, by default). Javadoc automatically places the first sentence in the method summary table and index. With out proper punctuation the Javadoc may be malformed. All items eligible for the {@inheritDoc} tag are exempt from this requirement.
- Check text for Javadoc statements that do not have any description. This includes both completely empty Javadoc, and Javadoc with only tags such as @param and @return.
- Check text for incomplete HTML tags. Verifies that HTML tags have corresponding end tags and issues an "Unclosed HTML tag found:" error if not. An "Extra HTML tag found:" error is issued if an end tag is found without a previous open tag.
- Check that a package Javadoc comment is well-formed (as described above) and NOT missing from any package-info.java files.
- Check for allowed HTML tags. The list of allowed HTML tags is "a", "abbr", "acronym", "address", "area", "b", "bdo", "big", "blockquote", "br", "caption", "cite", "code", "colgroup", "del", "div", "dfn", "dl", "em", "fieldset", "h1" to "h6", "hr", "i", "img", "ins", "kbd", "li", "ol", "p", "pre", "q", "samp", "small", "span", "strong", "sub", "sup", "table", "tbody", "td", "tfoot", "th", "thread", "tr", "tt", "ul".

These checks were patterned after the checks made by the DocCheck doclet available from Sun.

```
public static enum State {
    WAIT,
    RIDING,
    NO_WAIT
```

Circumstance.java:53, JavadocTypeCheck, Priority: High

Missing a Javadoc comment.

Checks Javadoc comments for class and interface definitions. By default, does not check for author or version tags. The scope to verify is specified using the Scope class and defaults to Scope.PRIVATE. To verify another scope, set property scope to one of the Scope constants. To define the format for an author tag or a version tag, set property authorFormat or versionFormat respectively to a regular expression.

Error messages about type parameters for which no param tags are present can be suppressed by defining property allowMissingParamTags.

Fecs.java:26, JavadocVariableCheck, Priority: High

Missing a Javadoc comment.

Checks that variables have Javadoc comments.

Checkstyle – modifier category

Details

Packages | Files | **Types** | Warnings | Details

Type	Total
ModifierOrderCheck	3
RedundantModifierCheck	9
Total	**12**

Engine.java:26, **ModifierOrderCheck, Priority: High**

'public' modifier out of order with the JLS suggestions.

Checks that the order of modifiers conforms to the suggestions in the Java Language specification, sections 8.1.1, 8.3.1 and 8.4.3. The correct order is:

1. public
2. protected
3. private
4. abstract
5. static
6. final
7. transient
8. volatile
9. synchronized
10. native
11. strictfp

```
static public final int STATE_STOP = 0b00,
```

ICircumstance.java:9, **RedundantModifierCheck, Priority: High**

Redundant 'public' modifier.

Checks for redundant modifiers in:

1. interface and annotation definitions,
2. the final modifier on methods of final classes, and
3. inner interface declarations that are declared as static

```
public interface ICircumstance {
  public static final String DEFAULT = "Default"; //1
```

Rationale: The Java Language Specification strongly discourages the usage of "public" and "abstract" for method declarations in interface definitions as a matter of style.

Variables in interfaces and annotations are automatically public, static and final, so these modifiers are redundant as well.

As annotations are a form of interface, their fields are also automatically public, static and final just as their annotation fields are automatically public and abstract.

Final classes by definition can not be extended so the final modifier on the method of a final class is redundant.

Checkstyle – naming category

Details

Packages | Files | **Types** | Warnings | Details

Type	Total
ConstantNameCheck	5
LocalVariableNameCheck	2
MethodNameCheck	2
Total	**9**

ICircumstance.java:15, ConstantNameCheck, Priority: High

Name 'CircumstanceVector' must match pattern '^[A-Z][A-Z0-9]*(_[A-Z0-9]+)*$'.

No description available. Please upgrade to latest checkstyle version.

CabinsController.java:14, LocalVariableNameCheck, Priority: High

Name 'dy_l' must match pattern '^[a-z][a-zA-Z0-9]*$'.

No description available. Please upgrade to latest checkstyle version.

UserInterface.java:269, MethodNameCheck, Priority: High

Name '$$$setupUI$$$' must match pattern '^[a-z][a-zA-Z0-9]*$'.

No description available. Please upgrade to latest checkstyle version.

Checkstyle – size category

01 jFeature

02 System Testing

03 Testlink

04 Static Analysis

Details

Packages | Files | Warnings | Details

Package	Total
fecs	20
fecs.interfaces	3
fecs.model	1
fecs.simulator	27
fecs.ui	52
src/main/java/fecs	4
Total	**107**

UserInterface.java:376, LineLengthCheck, Priority: High

Line is longer than 80 characters (found 209).

Checks for long lines.

Rationale: Long lines are hard to read in printouts or if developers have limited screen space for the source code, e.g. if the IDE displays additional information like project tree, class hierarchy, etc.

Checkstyle – whitespace category

Details

Packages | Files | **Types** | Warnings | Details

Type	Total
NoWhitespaceBeforeCheck	1
WhitespaceAfterCheck	43
WhitespaceAroundCheck	196
Total	**240**

Engine.java:199, NoWhitespaceBeforeCheck, Priority: High

```
accel *= ((Math.abs(vector*0.5) < Math.abs(leftVector)) ? 1 : -1) ;
```

';' is preceded with whitespace.

Checks that there is no whitespace before a token. More specifically, it checks that it is not preceded with whitespace, or (if linebreaks are allowed) all characters on the line before are whitespace. To allow linebreaks before a token, set property allowLineBreaks to true.

Circumstance.java:19, WhitespaceAfterCheck, Priority: High

```
this.put(FLOOD,new FloodCircumstance());
```

',' is not followed by whitespace.

Checks that a token is followed by whitespace.

Circumstance.java:30, WhitespaceAroundCheck, Priority: High

```
pyCircumstance = Fecs.interp.get(name+"Circumstance");
```

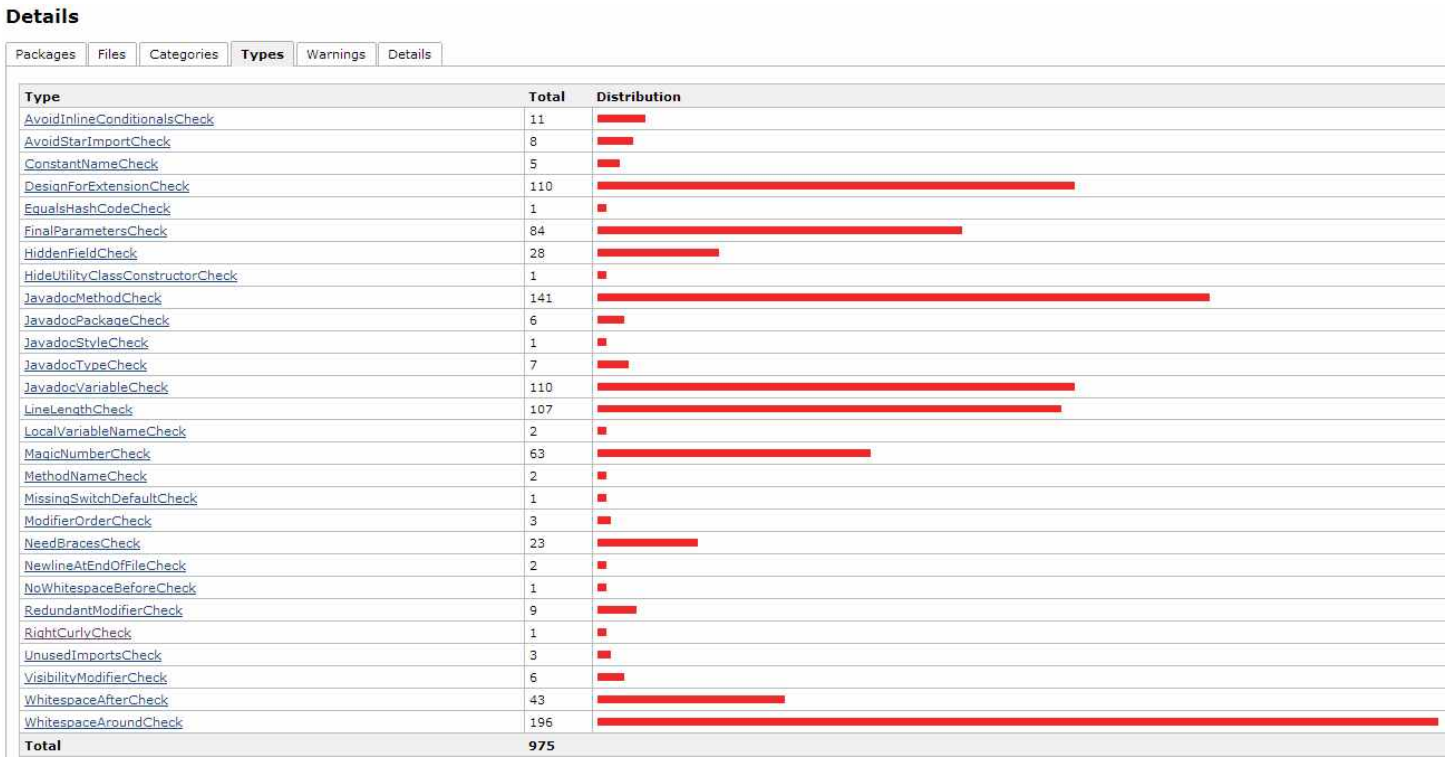
'+' is not preceded with whitespace.

Checks that a token is surrounded by whitespace. Empty constructor and method bodies (blocks) of the form

public MyClass() {} // empty constructor public void func() {} // empty method

may optionally be exempted from the policy using the allowEmptyMethods and allowEmptyConstructors properties.

Checkstyle



Details

Packages | Files | Categories | **Types** | Warnings | Details

Type	Total	Distribution
AvoidInlineConditionalsCheck	11	
AvoidStarImportCheck	8	
ConstantNameCheck	5	
DesignForExtensionCheck	110	
EqualsHashCodeCheck	1	
FinalParametersCheck	84	
HiddenFieldCheck	28	
HideUtilityClassConstructorCheck	1	
JavadocMethodCheck	141	
JavadocPackageCheck	6	
JavadocStyleCheck	1	
JavadocTypeCheck	7	
JavadocVariableCheck	110	
LineLengthCheck	107	
LocalVariableNameCheck	2	
MagicNumberCheck	63	
MethodNameCheck	2	
MissingSwitchDefaultCheck	1	
ModifierOrderCheck	3	
NeedBracesCheck	23	
NewlineAtEndOfFileCheck	2	
NoWhitespaceBeforeCheck	1	
RedundantModifierCheck	9	
RightCurlyCheck	1	
UnusedImportsCheck	3	
VisibilityModifierCheck	6	
WhitespaceAfterCheck	43	
WhitespaceAroundCheck	196	
Total	**975**	

- Checkstyle points at your coding style.

Checkstyle


```
<plugin>
    <artifactId>maven-checkstyle-plugin</artifactId>
    <version>2.12.1</version>
    <configuration>
        <configLocation>checkstyle.xml</configLocation>
    </configuration>
</plugin>
```

- To define checkstyle custom rules, you can edit build xml file like this.

PMD

01 jFeature

02 System Testing

03 Testlink

04 Static Analysis

PMD Result

Warnings Trend

All Warnings	New Warnings	Fixed Warnings
41	0	0

Summary

Total	High Priority	Normal Priority	Low Priority
41	0	41	0

Details

Packages | Files | Categories | Types | Warnings | Details

Package	Total	Distribution
fecs	2	
fecs.interfaces	9	
fecs.simulator	29	
fecs.ui	1	
Total	**41**	

Details

Packages | Files | **Categories** | Types | Warnings | Details

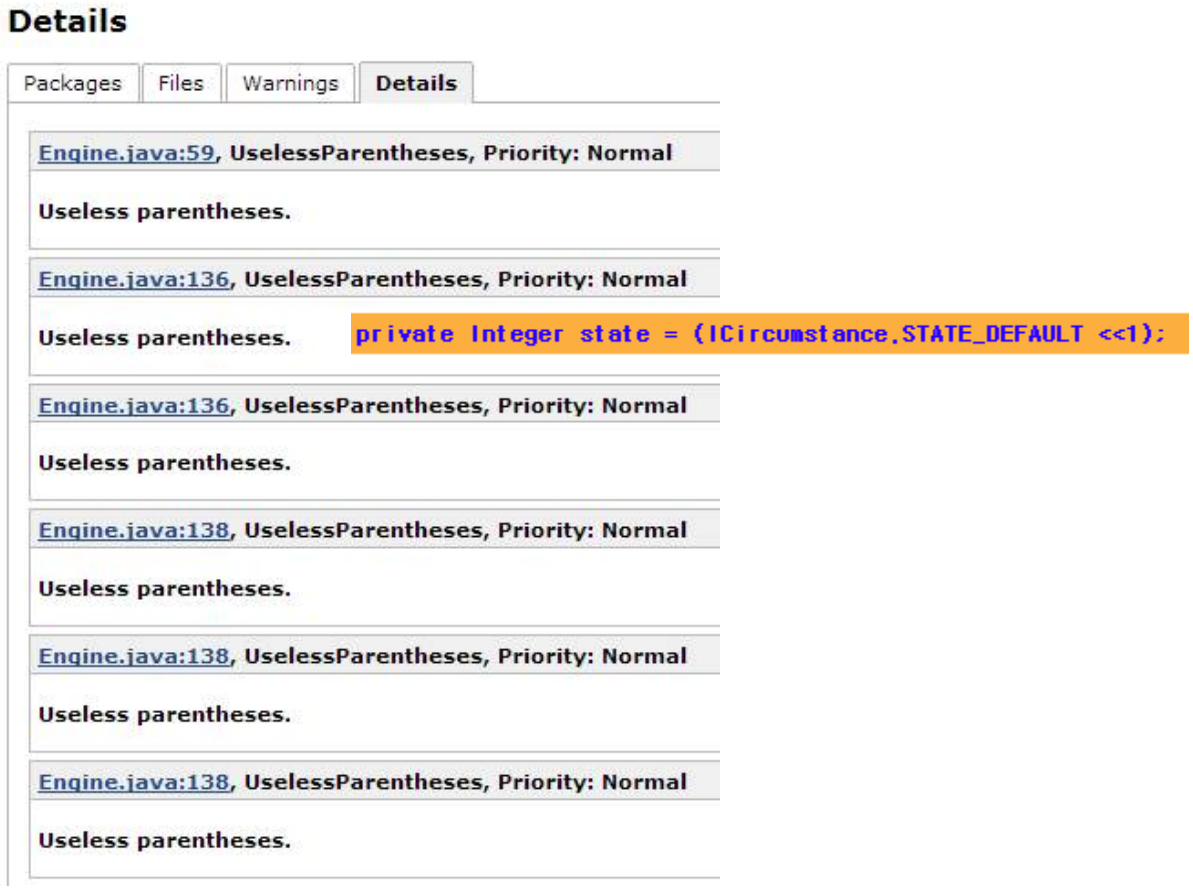
Category	Total	Distribution
Import Statements	3	
Unnecessary	21	
Unused Code	17	
Total	**41**	

PMD


```
<plugin>
    <groupid>org.apache.maven.plugins</groupid>
    <artifactid>maven-pmd-plugin</artifactid>
    <version>2.4</version>
    <configuration>
        <rulesets>
            <ruleset>rulesets/unusedcode.xml</ruleset>
            <ruleset>rulesets/coupling.xml</ruleset>
            <ruleset>rulesets/clone.xml</ruleset>
            <ruleset>rulesets/codesize.xml</ruleset>
            <ruleset>rulesets/braces.xml</ruleset>
            <ruleset>rulesets/naming.xml</ruleset>
            <ruleset>rulesets/optimizations.xml</ruleset>
            <ruleset>rulesets/imports.xml</ruleset>
            <ruleset>rulesets/logging-java.xml</ruleset>
            <ruleset>rulesets/strings.xml</ruleset>
        </rulesets>
        <sourceencoding>utf-8</sourceencoding>
        <targetjdk>1.6</targetjdk>
        <minimumtokens>10</minimumtokens>
        <excludes>
            <exclude>**/com/xenomity/common/**/*.java</exclude>
        </excludes>
    </configuration>
</plugin>
```

PMD – Import statements category

PMD – unnecessary category



PMD – unused code category

Details

Packages | Files | **Types** | Warnings | Details

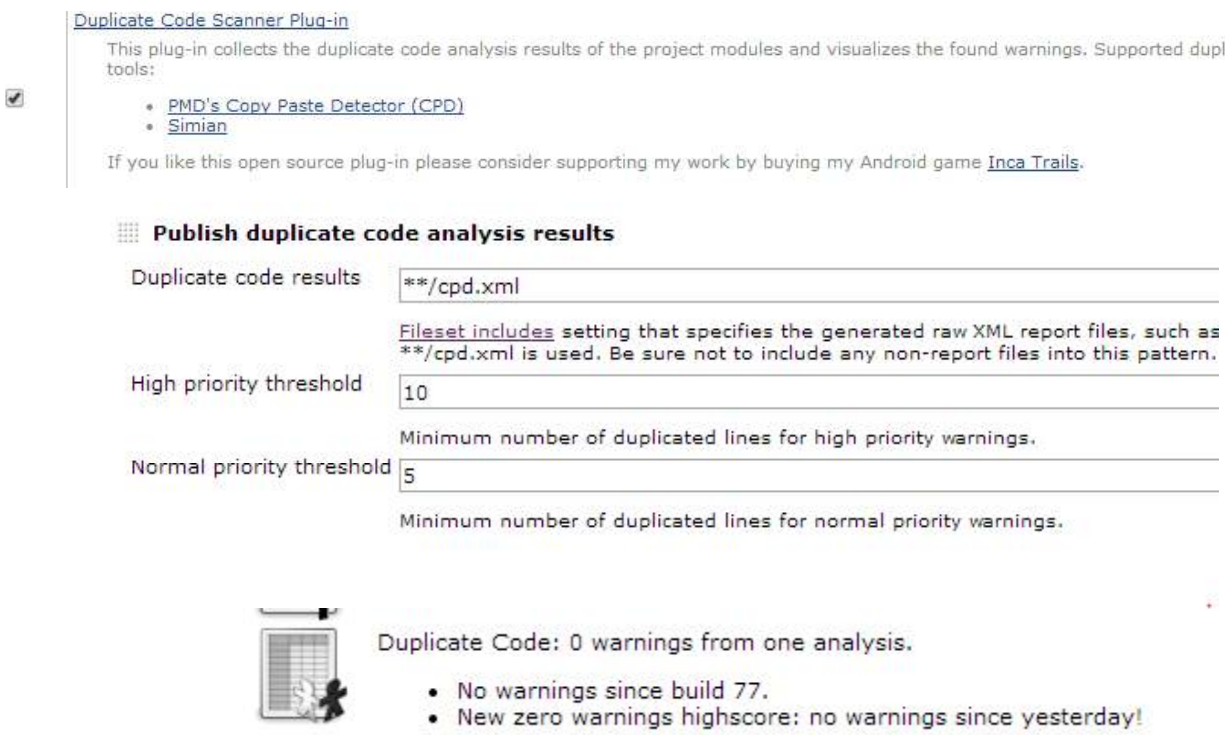
Type	Total
UnusedLocalVariable	1
UnusedModifier	9
UnusedPrivateField	7
Total	**17**

```
public static final String DEFAULT = "Default"; //1
```

```
private PythonInterpreter interp;
```

```
private static final double TARGET_WIDTH = 300;
```

PMD – CPD



- You have to ‘Duplicate code scanner plug-in’ for CPD.
- Edit the build script.

PMD – CPD


```
public void changeMoreEnterProbability(String moreEnterProbability) {
  try {
    Double val = Double.parseDouble(moreEnterProbability)/100;

    if (val >= 0 && val <= 1) {
      engine.setMoreEnterProbability(val);
    } else {
      throw new NumberFormatException();
    }

  } catch (NumberFormatException e) {
    displayError(e.getMessage());
    ui.getMoreEnterProbability().setText(engine.getMoreEnterProbability().toString());
  }
}

public void changeMotorOutput(String motorOutput) {
  try {
    Double val = Double.parseDouble(motorOutput);

    if (val > 0) {
      engine.setMotorOutput(val);
    } else {
      throw new NumberFormatException();
    }
```

- In eclipse, we can operate CPD. And you can see that upper codes are similar.

PMD – Cyclomatic reports

Location	Class or method	
src/main/java/fecs/CabinsController.java:10:	CabinsController	The class 'CabinsController' has a Cyclomatic Complexity of 11 (Highest = 10).
src/main/java/fecs/CabinsController.java:12:	CabinsController-contro	The method 'control' has a Cyclomatic Complexity of 10.
src/main/java/fecs/model/FloorType.java:6:	FloorType	The class 'FloorType' has a Cyclomatic Complexity of 5(Highest = 12).
src/main/java/fecs/model/FloorType.java:19:	FloorType-valueOf	The method 'valueOf' has a Cyclomatic Complexity of 12.
src/main/java/fecs/simulator/Engine.java:25:	Engine	The class 'Engine' has a Cyclomatic Complexity of 2 (Highest = 13).
src/main/java/fecs/simulator/Engine.java:183:	Engine-updateCabin	The method 'updateCabin' has a Cyclomatic Complexity of 13.
src/main/java/fecs/ui/Controller.java:21:	Controller	The class 'Controller' has a Cyclomatic Complexity of 4 (Highest = 13).
src/main/java/fecs/ui/Controller.java:54:	Controller-triggerFail	The method 'triggerFail' has a Cyclomatic Complexity of 13.
src/main/java/fecs/ui/UserInterface.java:19:	UserInterface	The class 'UserInterface' has a Cyclomatic Complexity of 2 (Highest = 20).
src/main/java/fecs/ui/UserInterface.java:63:	UserInterface-init	The method 'init' has a Cyclomatic Complexity of 20.

Number	Level of complexity
1~5	Low complexity
6~8	Moderate complexity
9~10	High complexity
11+	Very high complexity

Findbugs

01 jFeature

02 System Testing

03 Testlink

04 Static Analysis

FindBugs Result

Warnings Trend

All Warnings	New this build	Fixed Warnings
13	0	0

Summary

Total	High Priority	Normal Priority	Low Priority
13	4	9	0

Details

Packages | Files | Categories | Types | Warnings | Details | High | Normal

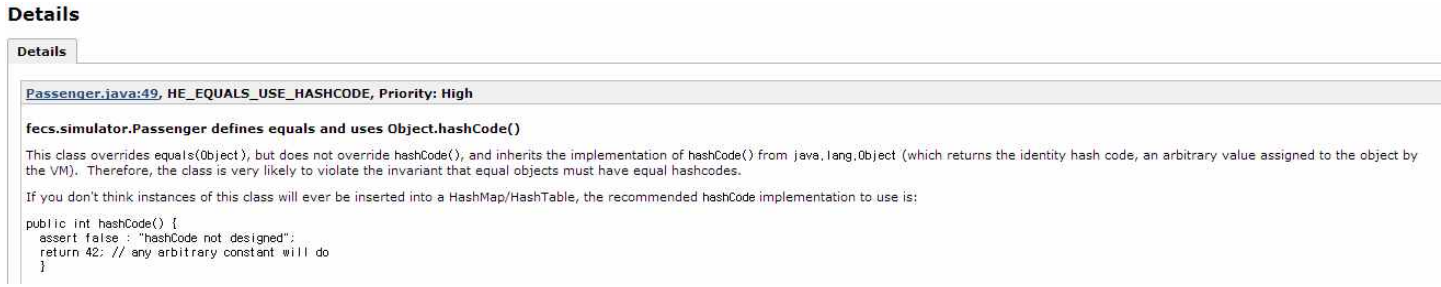
Package	Total	Distribution
fecs	4	
fecs.interfaces	1	
fecs.simulator	7	
fecs.ui	1	
Total	**13**	

Details

Packages | Files | **Categories** | Types | Warnings | Details | High | Normal

Category	Total	Distribution
BAD_PRACTICE	1	
CORRECTNESS	1	
MALICIOUS_CODE	2	
PERFORMANCE	4	
STYLE	5	
Total	**13**	

Findbugs – bad practice category



Details

Details

Passenger.java:49, HE_EQUALS_USE_HASHCODE, Priority: High

fecs.simulator.Passenger defines equals and uses Object.hashCode()

This class overrides equals(Object), but does not override hashCode(), and inherits the implementation of hashCode() from java.lang.Object (which returns the identity hash code, an arbitrary value assigned to the object by the VM). Therefore, the class is very likely to violate the invariant that equal objects must have equal hashcodes.

If you don't think instances of this class will ever be inserted into a HashMap/HashTable, the recommended hashCode implementation to use is:

```
public int hashCode() {
  assert false : "hashCode not designed";
  return 42; // any arbitrary constant will do
  }
```

```java
@Override
public boolean equals(Object o) {
    if (this == o) return true;
    if (o == null || getClass() != o.getClass()) return false;

    Passenger passenger = (Passenger) o;

    if (dest != null ? !dest.equals(passenger.dest) : passenger.dest != null) return false;
    if (start != null ? !start.equals(passenger.start) : passenger.start != null) return false;
    if (state != passenger.state) return false;

    return true;
```

Findbugs – correctness category

Details

Details

PassengerMaker.java:60, HE_USE_OF_UNHASHABLE_CLASS, Priority: Normal

fecs.simulator.Passenger doesn't define a hashCode() method but is used in a hashed data structure in fecs.simulator.PassengerMaker.killRandom()

A class defines an equals(Object) method but not a hashCode() method, and thus doesn't fulfill the requirement that equal objects have equal hashCodes. An instance of this class is used in a hash data structure, making the need to fix this problem of highest importance.

```
public void killRandom() {
  if (!passengers.isEmpty()) {
    int i = (int) (Math.random() * passengers.size());
    Passenger passenger = passengers.get(i);

    for (Cabin cabin : engine.getCabins().values()) {
      cabin.getPassengers().remove(passenger);
    }

    for (Floor floor : engine.getFloors().values()) {
      floor.getPassengers().remove(passenger);
    }

    this.passengers.remove(i);
    now--;
  }
}
```

Findbugs – malicious code category

Details

Packages | Files | **Types** | Warnings | Details

Type	Total
MS_MUTABLE_ARRAY	1
MS_SHOULD_BE_FINAL	1
Total	**2**

Fecs.java:50, MS_SHOULD_BE_FINAL, Priority: High

```
protected static PythonInterpreter interp= new PythonInterpreter();
```

fecs.Fecs.interp isn't final but should be

This static field public but not final, and could be changed by malicious code or by accident from another package. The field could be made final to avoid this vulnerability.

ICircumstance.java:15, MS_MUTABLE_ARRAY, Priority: High

fecs.interfaces.ICircumstance.CircumstanceVector is a mutable array

A final static field references an array and can be accessed by malicious code or by accident from another package. This code can freely modify the contents of the array.

```
public static final String[] CircumstanceVector = new String[] {DEFAULT,FIRE,FLOOD,CRASH,EARTH_QUAKE};
```

Findbugs – performance category

Details

Packages | Files | **Types** | Warnings | Details

Type	Total
DM_NEXTINT_VIA_NEXTDOUBLE	1
DM_NUMBER_CTOR	1
URF_UNREAD_FIELD	1
UUF_UNUSED_FIELD	1
Total	**4**

Circumstance.java:-1, **UUF_UNUSED_FIELD, Priority: Normal**

Unused field: fecs.Circumstance.interp

This field is never used. Consider removing it from the class.

Engine.java:126, **DM_NUMBER_CTOR, Priority: Normal**

fecs.simulator.Engine.draw() invokes inefficient new Integer(int) constructor; use Integer.valueOf(int) instead

Using new Integer(int) is guaranteed to always result in a new object whereas Integer.valueOf(int) allows caching of values to be done by the compiler, class library, or JVM. Using of cached values avoids object allocation and the code will be faster.

Values between -128 and 127 are guaranteed to have corresponding cached instances and using valueOf is approximately 3.5 times faster than using constructor. For values outside the constant range the performance of both styles is the same.

Unless the class must be compatible with JVMs predating Java 1.5, use either autoboxing or the valueOf() method when creating instances of Long, Integer, Short, Character, and Byte.

PassengerMaker.java:21, **URF_UNREAD_FIELD, Priority: Normal**

Unread field: fecs.simulator.PassengerMaker.lastUpdateTime

This field is never read. Consider removing it from the class.

```
private Long lastUpdateTime = null;
```

Controller.java:63, **DM_NEXTINT_VIA_NEXTDOUBLE, Priority: Normal**

fecs.ui.Controller.triggerFail(String) uses the nextDouble method of Random to generate a random integer; using nextInt is more efficient

If r is a java.util.Random, you can generate a random number from 0 to n-1 using r.nextInt(n), rather than using (int)(r.nextDouble() * n).

The argument to nextInt must be positive. If, for example, you want to generate a random value from -99 to 0, use -r.nextInt(100).

```
val = (int)(Math.random() * 10);
```

Comparisons between 3 tools

Tool	Comment
CheckStyle	- Points to user's coding style. - Requires custom setting cause catches too many warning.
PMD	- Has additional functions like CPD. - but, using CPD on Jenkins is not easy.
FindBugs	- Catches warnings to aim at improving efficiency.

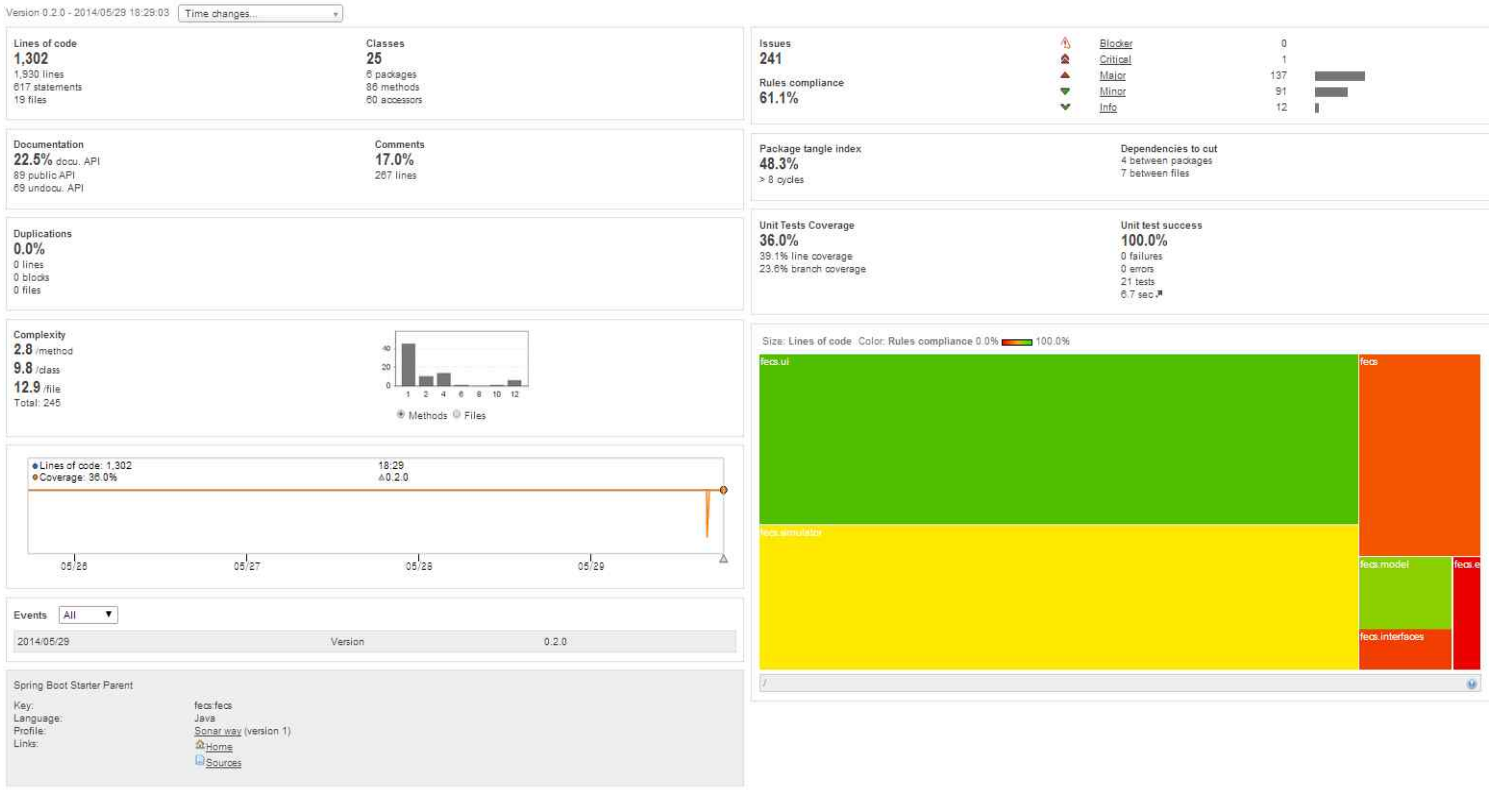
SONARQube - Dashboard

01 jFeature

02 System Testing

03 Testlink

04 Static Analysis



SONARQube – lines, documentation

Lines of code
1,302
1,930 lines
617 statements
19 files

Classes
25
6 packages
86 methods
60 accessors

Documentation
22.5% docu. API
89 public API
69 undocu. API

Comments
17.0%
267 lines

Language	Note
Java	Public classes, interfaces, methods, constructors, annotations and attributes. The following objects are excluded: • static final attribute • method with @Override annotation • empty constructor with no parameters • accessors (getters and setters) if the "sonar.squid.analyse.property.accessors" property is set to "true" (default value)

Public documented API (%)	public_documented_api_density	Density of public documented API = (**Public API** - **Public undocumented API**) / **Public API** * 100
Public undocumented API	public_undocumented_api	**Public API** without comments header.

SONARQube - complexity

01 jFeature

02 System Testing

03 Testlink

04 Static Analysis

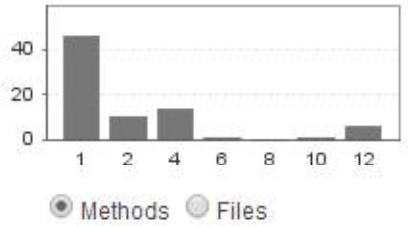
Duplications

0.0%

0 lines
0 blocks
0 files

Complexity

2.8 /method
9.8 /class
12.9 /file
Total: 245



Complexity /method

2.8

fecs.model	6.8
fecs.ui	3.9
fecs.simulator	2.9
fecs	1.9
fecs.event	1.0
fecs.interfaces	0.0

SONARQube - complexity

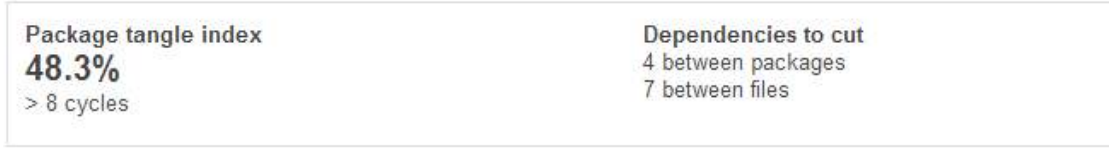
Java

- Keywords incrementing the complexity: `if`, `for`, `while`, `case`, `catch`, `throw`, `return` (that is not the last statement of a method), `&&`, `||`, `?`
- Notes:
  - `else`, `default`, and `finally` keywords do not increment the complexity.
  - a simple method with a switch statement and a huge block of case statements can have a surprisingly high complexity value (still it has the same value when converting a switch block to an equivalent sequence of if statements).
  - accessors are not considered as methods and so do not increment the complexity

Example: the following method has a complexity of 5

```
public void process(Car myCar){          // +1
    if(myCar.isNotMine()){               // +1
        return;                          // +1
    }
    car.paint("red");
    car.changeWheel();
    while(car.hasGazol() && car.getDriver().isNotStressed()){   // +2
        car.drive();
    }
    return;
}
```

SONARQube – Issue, coverage



SONARQube – Treemap, timeline

01 jFeature

02 System Testing

03 Testlink

04 Static Analysis

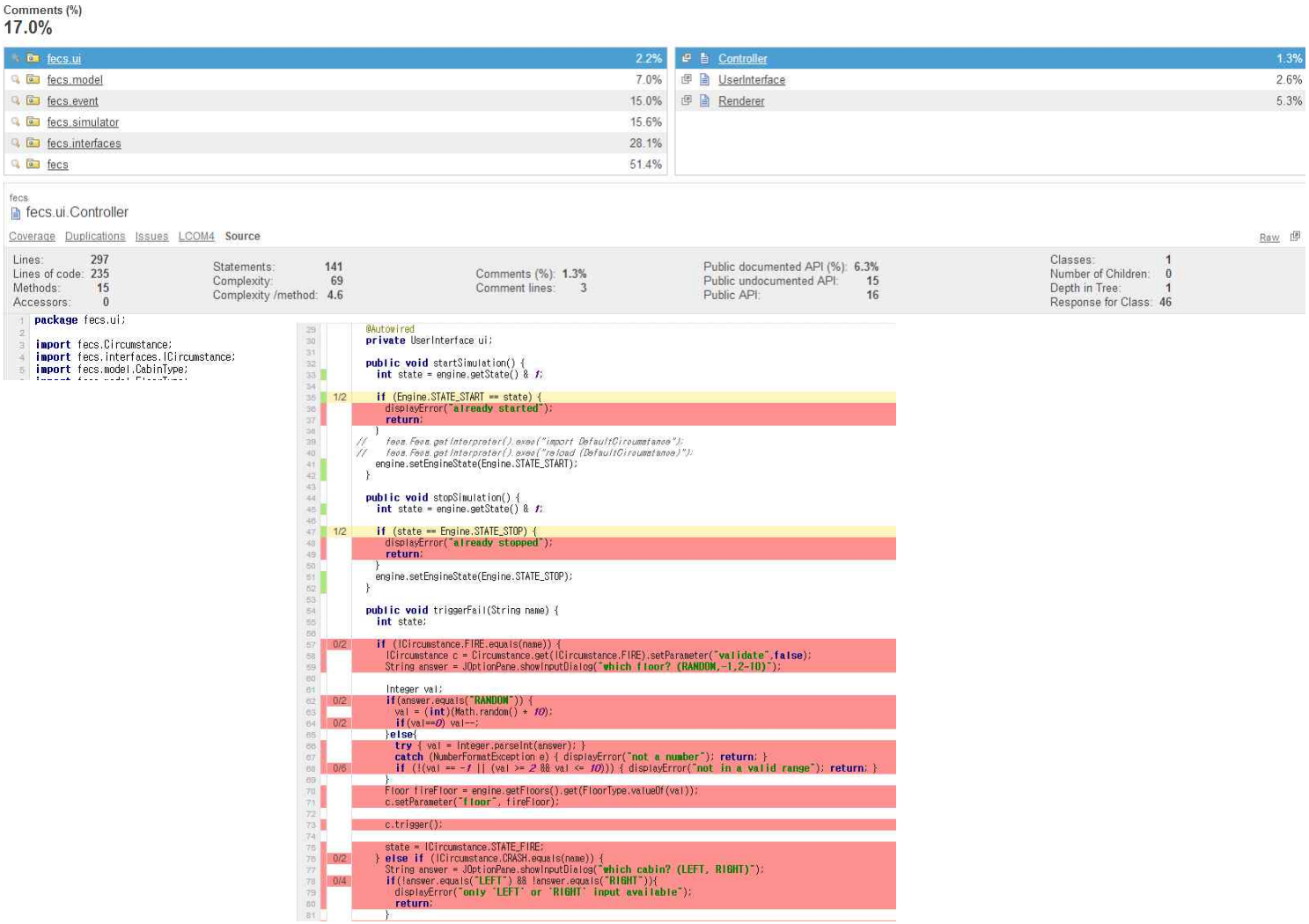
SONARQube - File

01 jFeature

02 System Testing

03 Testlink

04 Static Analysis



SONARQube - Cloud

01 jFeature

02 System Testing

03 Testlink

04 Static Analysis

- Quick Win : Lines
- Top Risk : Coverage

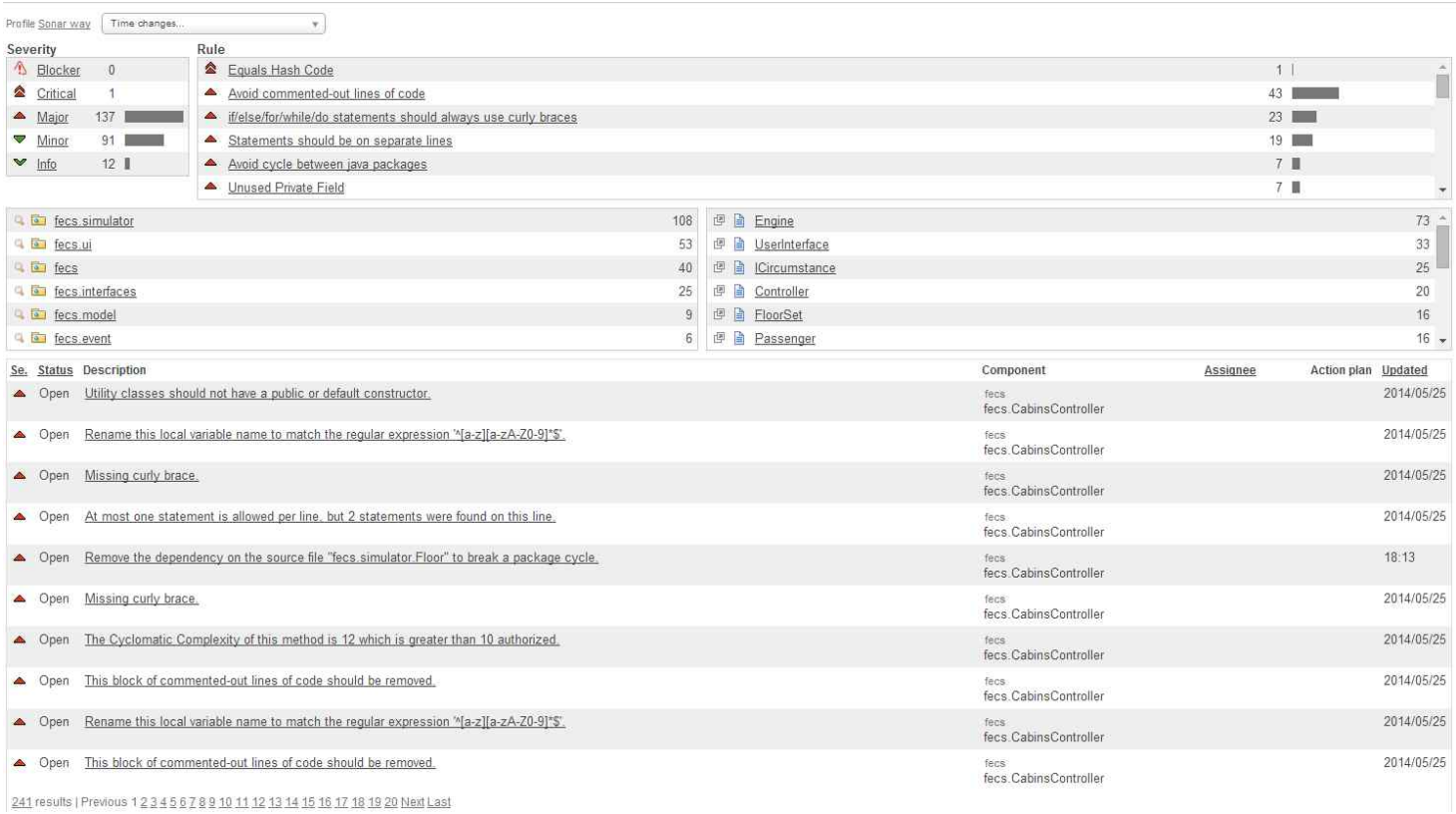
SONARQube - Issues

01 jFeature

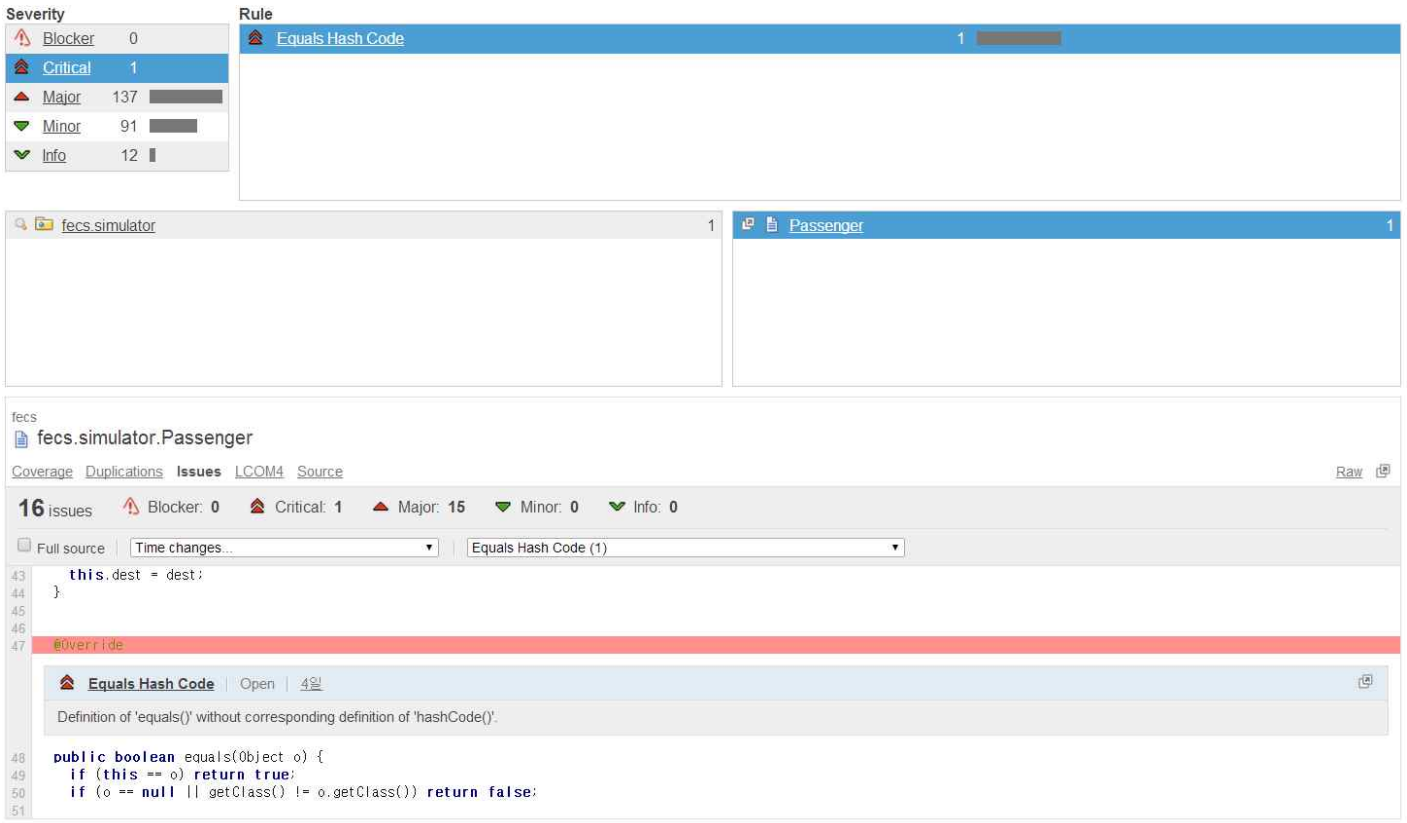
02 System Testing

03 Testlink

04 Static Analysis



SONARQube – Critical issues



SONARQube – Major issues

Issues	Exp
Avoid commented-out lines of code	일반 주석이 아닌 **code**에 주석을 친 경우 발생
if/else/for/while/do statements should always use curly braces	**If/else/for/while/do statement**에 괄호를 붙이지 않은 경우
Statements should be on separate lines	**Statement**를 줄을 떼지 않은 경우
Unused Private Field	사용되지 않은 **private field**
Avoid cycle between java packages	자바 패키지간 **cycle**이 발생한 경우
Avoid too complex method	메소드가 너무 복잡한 경우
Visibility Modifier	**Private**로 사용해야하는 변수를 다른 가시성으로 사용한 경우

SONARQube – Minor issues

Issues	Exp
Magic Number	직접적 숫자를 사용한 경우**(final** 등의 가시자를 붙이지 않고**)**
Trailing Comment	주석을 코드 바로 뒤에 붙여썼을 경우
Redundant Modifier	굳이 가시성을 선언하지 않아도 당연한데 가시한 경우
Modifier Order	**Modifier** 순서를 바꾸어 쓴 경우

Reference

http://bcho.tistory.com/832
http://kinoshita.eti.br/2012/10/11/jenkins-testlink-and-gtest-in-5-minutes-or-so.html
http://blog.naver.com/yarinii1?Redirect=Log&logNo=10154165439
https://gitorious.org/testlink-ga/testlink-code/commit/4842b189af26fbe81c4c8e6330fe17114ff58986
https://code.google.com/p/gallio-testlink-adapter/wiki/ApiKey
http://blog.naver.com/PostView.nhn?blogId=chloesso&logNo=110049101483
http://lfestivalet.wordpress.com/tag/jenkins-testlink-plugin/
http://mr100do.tistory.com/137
http://en.wikipedia.org/wiki/TestLink
http://www.mathworks.co.kr/discovery/static-code-analysis.html
http://en.wikipedia.org/wiki/Static_program_analysis
http://maven.apache.org/plugins/maven-checkstyle-plugin/examples/custom-checker-config.html
https://www.google.co.kr/url?sa=t&rct=j&q=&esrc=s&source=web&cd=4
&ved=0CEUQFjAD&url=http%3A%2F%2Fforum.magnolia-cms.c
om%2Fforum%2Fthread.html%3FthreadId%3D0490d
34e-2e01-41f7-9128-3a35a7337aed&ei=Y-eG
U7jzNoSrkgW6kYG4Bw&usg=AFQjCNFIpAvH2pa2oqo3x-3sY0ZkyPoamA&sig2=b_FePN6ujC
DLsNTCUPhpsA&bvm=bv.67720277,d.dGI&cad=rjt
http://www.xenomity.com/93