

SW 개발 품질관리 매뉴얼



미래창조과학부
Ministry of Science, ICT and
Future Planning

정보통신산업진흥원 부설
nipa SW공학센터

I. SW Visualization 배경

1. SW의 비가시성	3
2. SW의 복잡도 증가	4
3. 국내 중소기업의 개발환경	4

II. SW Visualization 정의

1. SW 공학 프로세스	9
가. 요구사항	10
나. 구현	13
다. 테스트	15
라. 형상관리	17
2. SW Visualization 이란?	18
가. 시각화	19
나. 문서화	20

III. SW Visualization의 기대효과

1. 요구사항의 구현 상태에 대한 실시간 파악	23
2. 품질 지표에 의한 객관적이고 정량적인 분석	24
3. 개발 진척상황에 대한 투명성 보장	24
4. 자동화를 통한 SW 관리의 편이성 증가	25
5. SW 개발관리 문서화 작업의 간소화	25
6. 품질 기반의 미래형 SW개발 문화를 위한 기반	25

CONTENTS

IV. 프로세스의 시각화

1. SW Visualization 품질지표	29
가. 개발 SW에 따른 지표 설정	29
2. SW Visualization 시스템	40
가. SW Visualization 프로세스	40
나. SW Visualization 시스템 구성요소	42
다. 시스템 구성 절차	51
3. SW Visualization 구성원 역할	63
가. Role & Responsibility 정의	63
나. SW 개발단계별 주요 Activity	64

V. 소스코드의 시각화

1. 역공학	71
가. 역공학 기법의 개요	71
2. 소프트웨어의 내부 설계	73
가. 개요	73
나. 품질 지표	74
3. 소스 코드의 시각화	76
가. Source Navigator 설치	77
나. Graphviz 설치	78
다. Code Visualization Tool 동작	79

VI. 소스코드의 문서화

1. JavaDoc	84
가. JavaDoc 주석 작성방법	84
나. JavaDoc 명령어를 통한 생성방법	86
다. 이클립스에서의 JavaDoc 생성	87
2. Doxygen	93
가. Doxygen 주석 작성방법	94
나. Doxygen 문서 생성방법	96
다. Doxygen 출력 문서 확인	100

VII. 프로세스의 문서화

1. Redmine 및 Impasse 설정	103
가. Redmine 설정	103
나. Impasse 설정	106
2. 프로젝트 산출물	108
가. 요구사항 명세서	108
나. 테스트 시나리오 및 결과서	109
다. 요구사항 주제표	111
3. Redmine table 분석	112
가. issues	113
나. journals	115
다. journal_details	115
라. custom_values	116

CONTENTS

마. projects	116
바. enumerations	117
4. Impasse table 분석	118
가. impasse_nodes	118
나. impasse_node_type	119
다. impasse_test_cases	119
라. impasse_test_steps	120
마. impasse_test_suites	120
바. impasse_test_plans	121
사. impasse_test_plan_cases	121
차. impasse_executions	122
5. 프로젝트 산출물 생성용 SQL	124
가. 요구사항 명세서	124
나. 테스트 시나리오 및 결과서	126
다. 요구사항 추적표	128

VIII. 사례

1. 요구사항 명세서	133
2. 테스트 명세서	170
3. SW아키텍처 설계 문서 양식	194

I

SW Visualization 배경

1. SW의 비가시성
2. SW의 복잡도 증가
3. 국내 중소기업의 개발환경

I

SW Visualization 배경

2002년 이후 글로벌 시장에서 SW의 시장규모가 급속하게 성장하고 있다. 실제로 전체 IT 산업에서 SW 시장의 규모는 1조 달러로 전체 IT 산업의 30% 수준이며, 8000억

달러 수준인 하드웨어 시장보다 큰 것으로 나타났다. 또한 IT 용·복합화가 진행되면서, 최종제품에서 SW의 역할과 기능이 점차 확대되고 있으며, SW가 최종제품의 부가가치를 높이는 역할을 수행할 뿐만 아니라 최종제품의 경쟁력을 좌우하는 핵심요소가 되었다.



이처럼 SW의 중요성이 날이 갈수록 강조되는 가운데에서도 SW의 비가시성이라는 특징과, 복잡도 증가, 국내 중소기업의 SW 개발환경은 SW 품질 관리를 어렵게 하는 요소로 작용하고 있다.

1. SW의 비가시성

SW 개발 관리에 있어서 중요한 문제점은 가시성을 확보하기 어렵다는 것이다. SW 생산성에 있어 가장 중요한 요소는 SW 실무자 개인의 역량(Rubey Raymond, 1978)이며, 그 개발 영역은 공개되지 않은 채 실질적으로 닫혀져 있는 SW 개발 과정의 결과로서 나온 제품만을 테스트할 수 있을 뿐이다. 이에 대한 방안으로서 여러 가지 프로

세스와 개발 방법론들이 만들어지고 있지만 이는 복잡성을 증가시킬 뿐 본질적인 문제는 전혀 해결되지 않고 있다.

SW 개발 과정 전반에 있어서의 가시성의 확보는 개발 후반에 발생하는 다양한 문제에 대하여 그 문제가 개발의 문제인지 더 나아가서는 요구사항 정의에 대한 문제인지에 대한 파악을 가능하게 한다. 이를 위해서는 요구사항 정의에서부터 개발, 테스트에 이르는 전체의 과정이 유기적으로 연결이 되어야 하며 이는 상호 추적성의 확보에서 시작된다. 따라서 이러한 추적성을 확보할 수 있는 개발 과정 전반의 연계가 필요하다.

2. SW의 복잡도 증가

SW의 개발은 그 과정이 순조롭게 진행된 것처럼 보여도 잠재되어있는 버그가 드러나거나 예상하지 못한 위험요소에 의해 납기지연, 비용초과, 품질저하 등의 문제가 발생되게 된다. 양적으로는 대규모의 SW가 요구되고, 질적으로는 SW의 복잡성이 증가되고 있는 SW산업현장에서 이런 문제들을 해결하기 위해서는 체계적이고 정량적인 접근 방법이 필요하다.

3. 국내 중소기업의 개발환경

SW에서 품질은 누구나 인식하듯이 그 중요성이 매우 높다. 이러한 SW의 품질 확보를 위하여 SW 공학에서는 요구사항, 형상관리, 테스트, 프로젝트 관리 등의 다양한 방법론을 통하여 SW의 품질을 확보하기 위한 방안을 제시하고 있다. 그러나 이러한 방법론은 매우 전문적이며 실제 SW를 개발하고 있는 국내 중소기업의 경우 이러한 방법론에 따라 SW의 개발을 진행하기에는 인력, 비용 등의 여러 측면에서 어려움이 존재한다. 또한 발주자와 수주자가 구분되어 있거나 개발자와 관리자가 구분되어 있으며, SW의 복잡도가 매우 높아지고 있는 개발 환경의 특성상 개발 과정 전반을 특정 인원 혹은 조직이 모두 파악하기는 어렵다.

또한 SW 개발 과정 전반에 있어서의 가시성을 확보하기 위해서는 블랙박스 테스트

와 화이트박스 테스트, 그리고 자동화 툴을 통한 소스 코드의 자동 진단을 통한 SW 구축 단계에서의 세부적인 측량이 필요하며, 이를 위한 SW공학 툴들이 개발되고 있으나 아직 소스 코드를 이해할 수 있는 SW 엔지니어 차원에서의 접근만이 이루어지고 있는 실정이다. 이를 가시화하여 SW 개발 경험이 없더라도 그 진척상황과 품질 수준을 쉽게 파악할 수 있도록 하고 요구 사항을 추적할 수 있는 통합 시스템이 필요한 시점이다.

II

SW Visualization 정의

1. SW 공학 프로세스

- 가. 요구사항
- 나. 구현
- 다. 테스트
- 라. 형상관리

2. SW Visualization 이란?

- 가. 시각화
- 나. 문서화

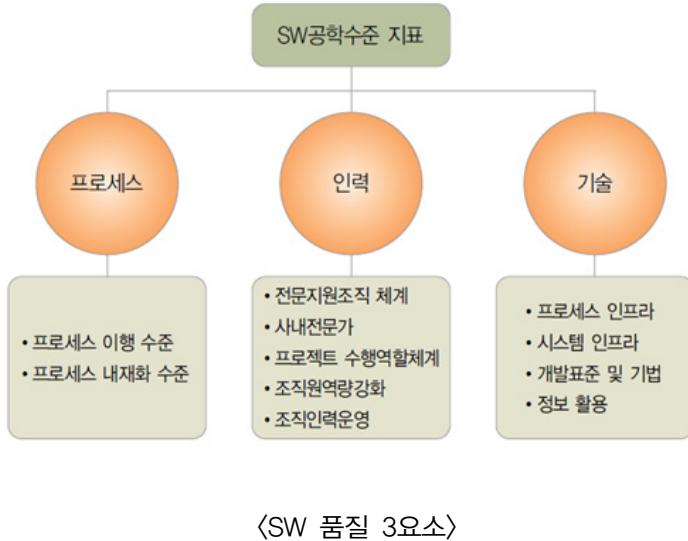
II

SW Visualization 정의

1. SW 공학 프로세스

다기능화 및 대규모화 되는 SW를 성공적으로 개발하기 위해서는 요구사항 분석에서부터 유지보수에 이르기까지 전 과정에 걸쳐 예상되는 어려움을 해결하기 위한 체계적인 관리와 효율적 업무수행을 지원해 주는 기술, 기법 등을 제공하는 SW 공학기술의 적용이 무엇보다 필요하다.

효과적인 SW 공학기술 적용을 위해서는 체계적인 업무방식 및 흐름의 정의와 이를 적용할 수 있는 프로세스, 전문적인 지식을 갖춘 조직 및 인력의 구성, 정의된 업무방식과 조직인력이 효율적으로 운영되기 위한 기반 인프라나 기술의 3가지 핵심요소를 균형 있고, 조화롭게 갖추고, 이를 유지하기 위한 지속적인 노력이 필요하다.



또 다른 측면에서는 SW를 개발하고 폐기하기까지의 과정인 요구사항, 분석/설계, 구현, 테스트, 유지보수 관련 기술과 이를 효과적으로 운영, 관리하기 위한 기술인 형상관리, 프로젝트 관리, 프로세스 관리, 도구 및 기법, SW 품질 관리 관련 기술도 균형 있고, 조화롭게 갖춰야 할 것이다.

성공적인 SW 개발의 목표는 주어진 일정과 비용을 준수하면서 고객이 요청하는 품질수준의 제품을 만들어내는 것이다. 이런 목표를 달성하기 위해서는 앞서 언급한 핵심요소(프로세스, 인력, 기술)에 대한 투자와 노력이 소요된다. SW를 개발하는 기업 입장에서는 얼마나 많은 품질비용을 투입해야 원하는 성과를 얻을 수 있는지 결정하는 문제는 어려운 문제이다. 따라서 적절한 품질비용의 투자와 이에 대한 성과에 대한 관리도 필수적인 요소이다.

따라서 SW 품질 향상을 위한 활동과 더불어 그에 따른 비용이 투입되었는지를 측정하고, 이들 활동들에 따른 성과를 지속적으로 분석하고, 문제점을 보완하려는 노력이 필요하다. 이런 노력과 분석을 통해 어디에 투자하는 것이 가장 바람직한가를 결정해야 하며, 현실적으로 품질이 가장 취약한 부분에 적극 투자할 수 있도록 해야 한다.

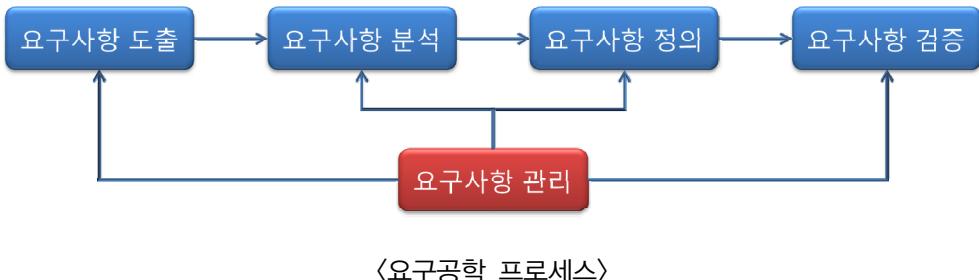
가. 요구사항

SW 공학을 적용하여 SW를 개발할 때 가장 먼저 개선되어야 할 영역중의 한 분야가 요구사항 관리 분야이다. 이는 해당 SW가 수행해야 할 역할이나 기능들에 대한 명확한 정의 없이는 사용자들을 만족시킬 수 있는 SW 제품을 기대할 수 없기 때문이다. 하지만 이런 명확한 사실에도 불구하고 많은 SW 개발 프로젝트들이 요구사항 관리의 실패로 인해서 어려움을 겪고 있다.

프로젝트 초기단계에서 수행되는 요구사항의 도출, 분석, 명세화 및 검증을 포함한 효율적인 요구사항 정의는 프로젝트 진행 중의 재작업을 감소시키고 개발 작업의 속도를 증가시켜 비용 및 시간 절약효과를 가져다준다. 이를 위한 공학적 접근방법을 요구공학이라고 한다.

요구공학은 고객의 요구 분석, 요구사항의 개발, 명세, 검증을 위한 매커니즘을 제공하고, 요구사항 개발 및 관리의 수행을 위한 체계적이고 총괄적인 접근법을 제공한다. 요구공학에서 중요한 이슈 중 하나는 요구사항의 변경을 관리하는 것이다. 요구사항의 변경은 이후에 진행되는 SW 개발단계에 많은 영향을 미치게 되고, 궁극적으로 SW 제품의 배포 이후 많은 비용을 요구한다. 따라서 요구공학은 요구사항 변경의 원인과 처리방법을 이해하고 완전성과 정확성을 갖고 품질을 만족하는 요구사항 관리를 통해

SW 프로젝트 실패를 최소화하는 것을 목표로 한다.



요구공학 프로세스는 요구사항 개발 프로세스와 요구사항 관리 프로세스로 구분한다. 요구사항의 개발은 도출/분석/명세화/검증 및 확인의 순서로 진행하여 개발자의 바람을 이끌어 내는 것이다. 요구사항 관리는 요구사항을 개발 후에 명세 된 요구사항의 변경 이력을 관리하여 이해관계자들이 동일한 요구사항을 공유할 수 있도록 하여 다른 공학 영역과의 연계를 통한 품질향상을 실현하는 것이다.

요구사항의 개발에서 요구사항 도출 단계는 이해관계자들이 제시하는 추상적 요구에 대한 관련 정보를 수집하여 식별하고, 구체화하는 단계이다. 프로젝트 관리자는 다양한 요구사항 도출 방법을 프로젝트의 특성에 맞추어 적절한 선정해야 한다. 그리고 요구사항의 누락을 방지하기 위해서는 관련법규를 포함한 다양한 소스를 빠짐없이 식별해야 한다. 그리고 관련된 이해관계자들이 모두 참여하여 기술적인 구현 가능성, 예산, 범위 등에 구애받지 않고 다양한 요구사항을 수집하는 단계이다.

요구사항 분석 단계에서는 도출된 요구사항들을 중요도, 구현가능성, 상호연관성 등을 고려하여 정제한다. 이렇게 정제된 요구사항들은 구현 가능한 요구사항과 불가능한 요구사항으로 구분하고, 해당영역에 따라 분류한다. 분류된 요구사항은 중요도에 따라 우선순위화하고 프로젝트에 반영 여부를 결정한다.

요구사항 명세화 단계에서는 정제된 요구사항들을 최종 요구사항으로 확정하기 위해서 범위와 기술을 객관화하여 명세서를 작성한다. 그리고 요구사항 명세서는 어느 누가 읽어도 오해의 소지가 없도록 명확하고 일관성 있게 작성되어야 하며, 구현 방법은 일반적으로 설계의 영역이기에 기술하지 않는다.

요구사항 검증 및 확인 단계에서는 작성된 요구사항 명세서가 오류 없이 작성되었

는지 검증하고, 고객의 요구사항이 요구사항 명세서에 빠짐없이 반영이 되었는지 확인하는 과정을 거치게 된다. 검증 후 고객과의 협의를 통해서 요구사항을 최종 확정하게 되며, 이를 베이스 라인이라 한다.

이와 같은 요구사항 개발 프로세스를 기반으로 이 문서에서는 요구사항 문서 사례를 8장에서 정리하고 있다. 8장은 크게 2개의 Part로 구성되어 있다. Part I은 테스트 케이스와 테스트 결과 정리서가 통합된 형태와 Part II는 테스트 케이스와 테스트 결과 정리서가 분리된 형태의 예를 보여 주고 있다.

Part I은 개발자가 개발할 때 사용하는 시스템 요구사항 내용이 주로 정리되어 있으면 Case Tool의 특성에 따라 요구사항 관리에 필요한 내용이 추가되어 있다. 여기서는 이슈관리 Tool인 Redmine을 요구사항 관리 Tool로 사용하고 있으면 Redmine에서 요구사항 관리로 필요한 내용을 추가로 정리되어 있다.

Part II는 과제에 관련된 모든 이해관계자의 의견을 반영하고 이해할 수 있는 형태의 예로 작성된 요구사항으로 비즈니스 요구사항, 사용자 요구사항, 그리고 시스템 요구사항까지 작성한 예이다. 특히 여기서는 사용자 요구사항과 시스템 요구사항을 더욱 구체화 위하여 Use Case Modeling, QAW 방법론을 사용하였다.

그리고 프로젝트를 진행하다보면, 환경 및 조직의 변화, 혹은 기술, 시간, 비용적 문제 등 여러 요인으로 인해 확정된 요구사항일지라도 수시로 변경이 발생하는데, 프로젝트 참여자들이 동일한 요구사항을 보고 프로젝트를 수행하기 위해서는 이러한 변경에 대한 체계적인 관리가 필요하다. 승인절차에 의해 최초로 확정된 요구사항이 베이스라인에 변경이 발생하게 되면, 반드시 정해진 절차를 통해 변경하여 모든 이해관계자들이 동일하게 변경된 베이스라인을 보고 프로젝트를 수행할 수 있도록 해야 하며, 이를 변경관리라 한다. 그리고 프로젝트 진행에 있어 설계, 구현, 테스트는 모두 요구사항을 기반으로 수행되며, 요구사항의 변경은 모든 단계에 영향을 미치므로 이들의 연결정보가 관리되어야 하는데, 이것을 추적관리라고 한다.

나. 구현

SW 구현은 SW 공학의 가장 기본적인 활동으로, 코드 작성, 디버깅, SW 통합, 개발자 테스팅 활동들로 이루어진다.

SW를 구현할 때 고려해야 할 4가지의 주요 원칙이 있다. 첫 번째는 코딩의 복잡성을 최소화시키는 것이다. SW 구현의 복잡성을 줄이는 문제는 복잡한 정보를 유지하는 인간 능력의 한계와 관련이 있다. 이러한 한계로 인해 SW 디버깅과 같은 복잡한 문제를 해결하는 능력은 제한적일 수밖에 없으며 이는 결국 SW 코드는 단순하고, 가독성이 있어야 한다는 것을 의미한다.

둘째로 향후에 일어날 변화를 고려해야 한다. 시간이 지남에 따라 대부분의 SW 시스템은 변하기 때문에 이런 변화를 예상하고 SW를 구현해야 한다. 대부분의 경우 향후 일어날 일정부분은 예상이 가능하며, 코드는 이런 변화요소들에 대응할 수 있어야 한다.

셋째는 SW 구현은 검증이 가능해야 한다는 것이다. 이는 구현되는 SW가 테스팅이나 다른 검증 기술을 통해 여러의 발견이 용이한 형식으로 구현되어야 한다는 것을 의미한다.

마지막으로 표준을 활용하는 것으로, 앞서 언급한 세 가지의 원칙들을 표준 프로그램 언어 및 외부 프로그램과의 인터페이스를 제공하는 표준 API를 선택함으로써 용이하게 적용될 수 있다. 규모가 큰 프로젝트에서 프로그래머들이 하나의 형식을 따른다면 앞서 언급한 세 가지의 원칙들을 쉽게 달성할 수 있게 된다. 이런 경우, 적용하는 표준은 모범 사례를 제시하는 것이 아니라 임의의 가능한 선택으로부터 프로그래머가 일관된 선택을 하도록 요구하게 된다. 이 경우, 외부 표준화 기구의 표준보다는 일반적으로 기업 내부에서 자체적으로 만들어진 표준이 활용된다.

다수의 개발자가 참여하여 공동으로 SW를 구현하는 경우, 개발자의 코딩 스타일은 프로그램의 논리에 영향을 주는 것은 아니지만, 일의 효율에 큰 영향을 미칠 수 있기 때문에 코딩 규칙이나 표준을 준수하여 작성해야 한다. 그렇게 완성된 소스 코드는 마치 한 명의 개발자가 하나의 세션에서 이 코드를 작성한 것처럼 일관적인 스타일을 유지해야 한다.

SW 개발 회사들이 자체적으로 코딩 규칙이나 표준을 갖추고 있어야 하는 중요한 이유는 다음과 같다.

1. SW가 제작되면서부터 폐기 시까지 발생하는 비용의 80%가 유지보수 단계에서 발생하지만
2. SW를 최초 구현한 개발자가 그 SW를 유지보수 하는 경우는 거의 없다.
3. 그래서 코딩 규칙이나 표준은 가독성을 높여 다른 엔지니어들이 빨리 이해할 수 있도록 해 주기 때문이다.

이렇듯 SW 개발 업체들은 SW 개발 프로젝트를 성공적으로 수행하고 관리할 수 있도록 코딩 규칙이나 표준을 수립하여 모든 개발자가 프로젝트에서 조화롭게 작업할 수 있도록 해야 한다. 이렇게 해서 좋아진 소스 코드의 가독성은 개발자가 SW 시스템을 이해하는 데 영향을 미치게 되고, 가독성이 향상되면 이해가 용이해지고 유지보수가 수월해지며 일반적으로 코드의 품질도 좋아진다.

또한 코딩 규칙이나 표준을 지키게 되면 새로운 기능을 추가하고 기존 기능을 수정하기 위해 SW 시스템을 쉽게 변경할 수 있는 코드 유지 관리성도 좋아지게 된다. 가독성과 유지 관리성은 여러 가지 요인에 의해 좌우되지만, 모든 개발자가 영향을 미치는 SW 개발의 한 측면은 코딩 기술이다. 개별 개발자들의 코딩 기술은 개발자별로 상이하기 때문에 개발 팀에서 우수한 코드를 작성할 수 있도록 유도하는 가장 쉬운 방법은 코딩 표준을 설정한 다음 이 표준을 루틴 코드 검사에서 적용하는 것이다.

고급 코드를 작성하기 위해 완벽한 코딩 기술과 뛰어난 프로그래밍 기술을 사용하면 SW 품질과 성능을 향상시키는 데 중요한 역할을 할 수 있다. 또한 명확하게 정의된 코딩 표준과 올바른 코딩 기술을 일관적으로 적용하고, 이어서 루틴 코드 검사를 적용하면 보다 쉽게 SW 시스템을 이해하고 관리할 수 있다.

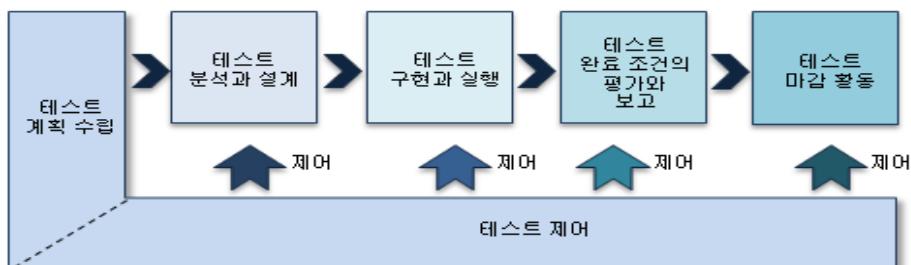
다. 테스트

SW 테스트란 프로그램에 있는 오류를 찾기 위해 프로그램을 실행하여 실행 결과와 예상 결과를 비교하고 검토하는 과정이다. 또한 응용 프로그램 또는 시스템의 동작과 성능, 안정성이 요구 수준을 만족하는지 확인하는 과정이며, 시스템의 결함을 발견하기 위한 방법이다.

SW 테스트와 평가는 품질인증의 중요한 요소이고, 명세서, 설계, 코드 생성의 최종 검토를 의미하기도 한다. 그러므로 올바른 테스트를 수행하고 수행결과를 검토하며 분석하는 것은 SW의 품질을 보증하고 향상시키는데 매우 중요한 역할을 한다. 즉, SW 테스트는 프로그램에 존재하는 오류와 결함을 검출하여 수정하고 요구사항에 부합되는지를 확인하며 SW의 품질을 향상시키는 것을 목적으로 한다.

일반적으로 테스트는 개발된 SW에 포함된 결함을 찾아내고 이를 수정하거나 제거하여 SW의 품질을 개선하려는 목적으로 수행된다. SW 테스트는 SW에 존재하는 결함을 찾아내는 것을 주목적으로 하며, 그 외에도 SW 테스트를 바라보는 SW 결함의 관점, 전체 시스템의 관점, SW 품질의 관점, 마케팅의 관점에 따라 각각의 필요성과 목적에 의해 테스트를 수행한다.

SW 테스트 프로세스는 SW 테스트를 수행하기 위한 전반적인 활동을 나타낸다. SW 테스트 프로세스는 크게 테스트 계획과 제어, 테스트 분석과 설계, 테스트 구현과 실행, 테스트 완료 조건의 평가와 리포팅, 테스트 마감 활동으로 이루어진다.



〈SW 테스트 프로세스〉

SW 테스트 구분의 SW개발생명주기상에서의 단계별 테스트와 각 SW의 특성을 고려한 테스트로 나눌 수 있습니다. SW개발생명주기상에서의 테스트는 개발순서에 따라 유닛테스트, 통합테스트, 시스템테스트 그리고 인수 테스트의 단계로 순차적으로 수행됩니다. 유닛테스트란 개발자 스스로가 본인이 개발한 소프트웨어 모듈이 개별적으로 잘 기능하는지를 테스트하는 것이고, 통합테스트는 유닛테스트에서 확인된 SW모듈들이 통합되어 돌아갈 때 모듈간 인터페이스가 잘 되어 이상 없이 동작하는지를 확인하는 것이다. 시스템 테스트는 통합된 모듈들이 원래 계획했던 대로 동작하는지, 시스템이 실제 동작과 원래 의도했던 것과 차이가 없는지 확인하는 과정이다. 그리고 인수 테스트란 구현된 시스템이 사용자 입장에서 원래의 요구사항들을 얼마나 만족시키는지를 평가하는 과정으로 알파테스트와 베타테스트로 구분된다.

다른 테스트의 관점으로 SW특성을 고려한 테스트가 있으며 개발된 SW의 특성에 따라 어떤 테스트를 어느 수준에서 할 것인지를 결정하는 것이 중요하다. 예를 들어 항공기 또는 자동차용 임베디드 SW는 인명손실이나 경제적 피해가 크기 때문에 신뢰성을 보장하기 위한 높은 수준의 테스트가 필요하고, 사무용 SW인 경우 높은 신뢰성보다는 사용자가 쉽고 편하게 사용하도록 사용성, 호환성을 확보하기 위해 노력하는 것이 현명할 수 있다. SW특성을 고려한 테스트에는 SW 프로그램의 표현상 오류를 찾아내는 정적 테스트와 SW가 동작하면서 발생하는 오류를 찾아내는 동적테스트가 있다. 정적테스트는 프로그램 작성 시 구문의 표현 방법, 문법, 규칙에서 벗어난 오류를 찾거나 SW구조상의 잠재적 문제를 찾아내어 개선하기 위한 테스트로 SW개발자가 정적분석 도구를 돌려 찾아낸다. SW동작시의 오류를 찾아내는 동적 테스트는 블랙박스 테스트와 화이트박스 테스트로 나누어지는데, 블랙박스 테스트는 기능적인 측면에서 시스템이 제대로 동작하고 있는지를 확인하는 것으로 입력과 출력 값만을 비교하여 비교적 빠른 시간에 수행할 수 있다. 화이트박스 테스트는 프로그램 소스코드들의 논리적인 모든 경로를 검사하여 동작의 유효성뿐만 아니라 실행되는 과정을 살펴봄으로써 정확성을 높일 수 있다. 성공적인 테스트를 위해서는 두 테스트간의 장단점을 잘 활용할 필요가 있다.

위 내용을 기반하여 이 문서에서는 시스템 테스트 케이스 명세서 사례를 8장에서 정리하고 있다. 8장은 크게 2개의 Part로 구성되어 있다. Part I은 테스트 케이스와 Test 결과 정리서가 통합된 형태와 Part II는 테스트 케이스와 Test 결과 정리서가 분

리된 형태의 예를 보여 주고 있다.

Part I는 테스트 케이스와 테스트 결과 정리서가 하나의 형태로 테스트 케이스의 관리보다는 빠른 결과에 대한 피드백이 필요한 과제에서 사용되는 것으로 많은 문서를 만들지 않아도 되는 장점이 있다. 그러나 테스트 케이스가 지속적으로 관리되지 않기 때문에 제품을 만들면서 발생할 수 있는 특성들이 테스트 케이스에 반영하는데 어려움이 있을 수 있다.

Part II은 테스트 케이스와 테스트 결과 정리서가 분리한 형태로 동일한 제품에 대하여 동일한 내용의 테스트를 자주 반복할 때 사용하는 사례로 테스트 케이스를 지속적으로 관리할 수 있는 장점이 있다. 그러나 테스트 케이스 작성시 작성해야 할 내용이 많아 테스트 케이스 작성시 많이 시간이 소요되는 단점이 있다.

SW테스트는 그 기능과 사용성, 신뢰성, 성능, 호환성 등의 특성을 고려하여 어떤 테스트를 어느 수준에서 수행할지를 경영적 차원에서 결정하여야 하며, 채택된 목표에 따라 전문성을 가지고 효과적으로 테스트되어야 한다.

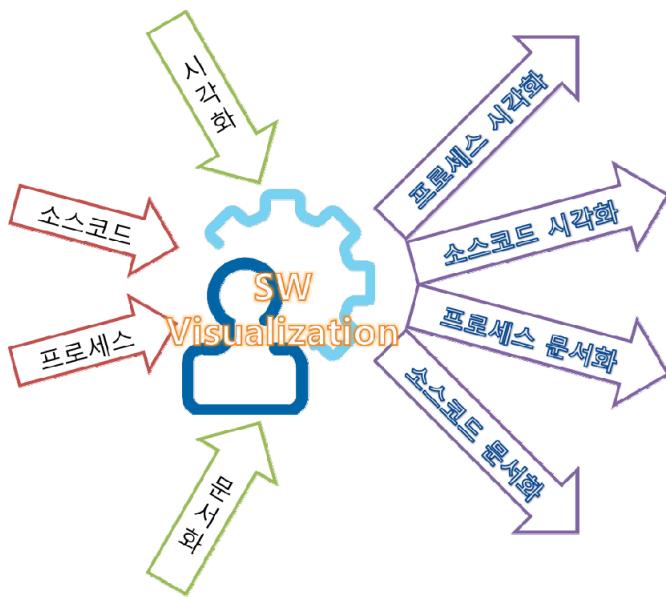
라. 형상관리

SW 형상관리는 SW생명주기 동안 발생하는 변경사항을 통제하기 위한 관리방법이다. 형상관리를 통하여 고객 또는 이해당사자들의 요구사항을 만족시키도록 변경관리를 하며, 변화하는 비즈니스 환경에서 SW가 변화에 대응할 수 있도록 SW 개발 및 진화를 통제하고 관리하는 활동이다.

SW의 변경은 근본적으로 네 가지의 주요 요인이 있다. 첫째, 제품 요구사항이나 비즈니스의 변화에 따른 새로운 비즈니스 및 시장 창출에 의한 변경 요청이 있고, 둘째, SW에 대한 이해당사자들의 변경 요청이 있으며, 셋째, SW품질 관련 팀의 요구에 의한 변경 요청이 있고, 넷째, 예산이나 일정의 변경이 주요 원인이다. SW 형상관리는 이러한 주요 원인을 SW 생명 주기 동안 효과적으로 관리하는 활동으로써, 세부 활동은 여러 연구를 통해서 다양하게 제시되었지만, 일반적으로 형상 식별, 변경 관리 및 통제, 버전 및 배포 관리, 감사 및 보고 활동으로 이루어져 있다.

SW 형상식별은 형상항목을 식별하고, 버전에 대한 식별 스키마를 수립하며, 항목을 획득하고 관리하기 위한 도구 및 기법을 정의한다. 형상항목 식별은 SW에 대한 변화를 제어하기 위한 첫 단계로, 이는 시스템 형상에서 SW 형상을 파악한 후, SW 형상 항목을 식별하여 SW 형상항목에 대한 라벨링 전략을 개발하고, 항목들 사이의 관계를 정의한다. SW 변경 관리 및 통제는 SW 형상 통제를 위한 활동을 뜻하며, 변경 및 통제는 변경 수행자와 지원 도구로 관리된다.

2. SW Visualization 이란?



좋은 결과를 위해서는 과정에 대한 엄밀한 관리가 필요하다. 마찬가지로 좋은 SW를 얻기 위해서는 SW개발 과정 전반에 대한 관리가 중요하며 이는 곧 SW개발 프로세스로 표현될 수 있다. 따라서 성공적인 SW 개발 관리를 위해서는 SW 자체, 즉 소스 코드와 SW개발 프로세스에 대한 관리가 필요하다 할 것이다.

SW 공학 프로세스는 이러한 관리를 위한 전통적인 방법이다. 그러나 SW 공학 프로

세스에 의한 SW개발 품질관리를 수행하기에 국내의 중소기업은 인력 및 비용이 부족하며, 그 방법이 매우 전문적이다. 이러한 현실에서 국내 중소기업이 SW개발 품질관리를 수행하기 위한 현실적인 방안이 필요한 시점이다.

SW Visualization은 이러한 방안으로 시각화와 문서화를 제안한다. 첫째, 시각화는 SW 개발의 가장 어려운 점인 SW 비가시성을 극복함으로써 SW개발의 전체 과정을 파악할 수 있도록 하며, 이를 통하여 SW개발 품질 관리를 실현하고자 한다. 둘째, 문서화는 기업의 개발 노하우 관리 및 내부 인력간의 업무 이해도 향상과 특정 상황에서 외부와의 의사소통을 위한 방안이다.

즉, SW Visualization은 소스코드와 개발 프로세스를 관리하는 것을 목적으로 하고, 시각화와 문서화를 그 방안으로 하여 SW개발 품질관리를 수행하기 위한 것이다. 이에 따라 본 문서에서는 ①개발 프로세스의 시각화, ②소스코드의 시각화, ③소스코드의 문서화, ④개발 프로세스의 문서화라는 4방향으로 SW개발 품질관리를 정리하고자 한다.

가. 시각화

경영관리체계 중 Plan, Do, See라는 개념이 있다. 이는 합리적인 계획과 그 계획에 따른 통제에 기초한 관리 방안으로 경영관리 뿐 아니라 다양한 분야에서 널리 사용되는 개념이다. SW Visualization은 SW개발 관리를 위하여 이러한 Plan, Do, See의 개념을 활용한다.

가치 있는 SW의 생산을 위해서는 전체 SW 개발 과정의 효율적인 관리가 필요하다. 성공적인 SW 개발 관리를 위해서는 명확한 목표수립과 효율적인 수행, 지속적 모니터링 및 통제 활동이 필요하다. SW Visualization은 ①지표 설정에 따른 명확한 목표수립과 ②시스템 기반의 효율적인 개발 활동 및 ③시각화를 통한 지속적 모니터링 및 통제를 가능토록 하여 성공



적인 SW 개발 관리의 기반을 제공한다.

SW 개발 기업은 SW Visualization을 활용함으로써 SW의 비가시성을 극복하고 SW 개발 과정의 투명성을 확보할 수 있으며, SW 개발 과정의 투명화는 SW 품질 확보 및 SW 개발 문제의 조기 검출을 통한 개발비용 절감으로 기업 경쟁력 확보로 이어질 수 있을 것이다.

나. 문서화

SW Visualization은 지표 설정에 따른 명확한 목표수립과 시스템 기반의 효율적인 개발 활동 및 시각화를 통한 지속적 모니터링 및 통제라는 큰 흐름에서 SW 개발 관리를 수행한다. 이 과정에서 SW 개발 과정 전반의 모든 데이터는 유기적으로 연결된다.

따라서 개발 후반부에 발생하는 문제의 빠른 분석과 개발 후반에 발생할 문제에 대한 조기 예측이 가능하다. 이러한 구성은 SW개발의 품질을 향상시키고 기업 경쟁력 확보로 이어질 수 있을 것이다.

SW 개발의 효율성과 SW개발 과정의 추적성 확보를 위해 개발 전반의 활동을 시스템으로 연계하는 것은 매우 중요하며, 이러한 활동에 대한 추가적인 문서는 필요치 않다.

그러나 기업의 산출물 관리, 발주자의 요구, 기타 인증 및 심사 과정에서의 필요에 따라 개발 산출물은 시스템 그 자체가 아닌 문서화된 형태로 작성되어야 할 수 있다. 이러한 경우 관련된 문서를 다시 생성하는 것은 시스템 기반의 개발 활동과 중복되는 이중의 작업이 될 수 있으며, SW개발 활동에 부담이 될 수 있다.

따라서 본 문서에서는 개발 과정에서 이미 시스템 내부에 적재되어 있는 다양한 산출물을 문서화할 수 있는 방안을 제시함으로써, SW Visualization 활용으로 인한 업무 부담을 최소화하고, 활용성을 극대화하고자 한다.

본 문서에서 문서화는 소스코드의 문서화, 개발 과정의 문서화로 구분하였다. 소스코드의 문서화는 개발자의 인력 이동 및 업무 이동으로 인한 업무 공백을 최소화할 수 있는 방안으로 활용하고자 한다. 개발 과정의 문서화는 SW공학적 측면에서 개발 과정의 검증에 활용되는 산출물을 시스템에서 자동으로 생성할 수 있도록 지원하고자 한다.

III

SW Visualization의 기대효과

1. 요구사항의 구현 상태에 대한 실시간 파악
2. 품질 지표에 의한 객관적이고 정량적인 분석
3. 개발 진척상황에 대한 투명성 보장
4. 자동화를 통한 SW 관리의 편이성 증가
5. SW 개발관리 문서화 작업의 간소화
6. 품질 기반의 미래형 SW개발 문화를 위한 기반

III

SW Visualization의 기대효과



〈SW Visualization 기대효과〉

1. 요구사항의 구현 상태에 대한 실시간 파악

대부분 발주자와 수주자가 분리되어 있으며 SW는 완성물이 나오기 전까지는 그 모습을 볼 수 없다는 비가시성이라는 특성에 따라 개발이 특정 단계까지 진행되기 전에는 발주자는 SW의 개발 상태를 확인할 수 없으며 이로 인하여 요구사항의 잣은 변경이 발생하고 요구사항의 잣은 변경으로 인하여 개발에 많은 어려움이 발생한다. 그러

나 SW Visualization 에서는 발주자가 요구사항의 구현 상태를 실시간으로 확인할 수 있도록 지원함으로써, 개발 일정 및 SW 품질에 대한 문제를 조기에 인지하고 그에 따른 대처가 가능하도록 지원한다.

2. 품질 지표에 의한 객관적이고 정량적인 분석

SW 개발이 일정 수준 이상 진행되면 SW의 품질에 따라 개발 완료를 결정하고, 제품의 Release를 결정하여야 한다. 그러나 100% 완벽한 SW는 존재할 수 없으며, 모든 SW는 잠재적으로 어느 정도의 결함을 내재하고 있다. 따라서 현재의 SW 품질이 Release를 할 수 있는 상태인지에 대한 결정이 필요하다. 그러나 이에 대한 명확한 기준은 없는 상태이며, 실제 대부분의 개발에서 과거의 기준에 의거하여 이러한 판단이 이루어지고 있다. 따라서 SW Visualization 에서는 시각화 된 지표가 어느 정도의 품질을 보일 때 인정될 수 있는 품질 수준인지에 대한 기준을 제시함으로써 SW 품질에 대한 기준을 제공한다. 이는 일정 수준 이상의 SW 시장 품질 확보를 가능하게 함으로써 기업의 제품 경쟁력 확보에 기여할 수 있다.

3. 개발 진척상황에 대한 투명성 보장

SW 결함에 대한 조치 비용은 그 결함이 발견되는 시점이 늦을수록 커진다. 따라서 SW 개발 관리자의 입장에서 관리에 가장 중요한 부분은 SW 개발 과정의 투명성 확보이다. SW Visualization은 SW 개발의 전 과정을 연결하고 요구사항 분석 단계에서부터 제품 Release 시점까지 각 시스템에서 산출된 데이터를 기반으로 개발의 진행 상태 및 품질을 시각화함으로써 SW 개발의 투명성을 확보하고, 이를 통하여 SW 개발에서 발생할 수 있는 결함을 미연에 방지하고, 발생한 결함을 조기에 조치할 수 있도록 지원할 수 있다.

4. 자동화를 통한 SW 관리의 편이성 증가

SW 공학에서는 SW 품질 확보를 위한 여러 가지 방법론을 제시하고 있다. 그러나 이러한 방법론에 따라 SW를 개발하는 것은 중소기업 입장에서는 비용과 인력 측면에서 많은 부담으로 작용한다. 따라서 이러한 SW 공학적인 방법을 시스템으로 자동화함으로써 SW 공학에 대한 전문적인 지식이 없거나, 지식이 있어도 비용 및 인력이 부족하여 그 방법에 따라 개발을 진행할 수 없는 중소기업이 쉽게 SW 공학적인 방법에 따라 SW 개발을 진행할 수 있도록 지원한다. 이는 결국 SW의 품질 향상으로 연결될 것이다.

5. SW 개발관리 문서화 작업의 간소화

SW의 품질 확보와 추후 개발의 편의를 위하여 SW 개발 과정의 많은 산출물과 개발 과정은 문서화된 상태로 관리되는 것이 바람직하다. 그러나 이러한 문서화 작업은 때로 개발 과정에서 부담으로 작용할 수 있으며 개발 일정의 지연을 초래할 수 있다. 그러나 SW Visualization에서는 개발 과정 전체를 시스템화 함으로써 문서 작업을 대폭 줄이고 시스템 자체가 문서를 대신하도록 함으로써 개발 편의성의 확보를 지원한다.

6. 품질 기반의 미래형 SW개발 문화를 위한 기반

최근의 SW는 SW의 융·복합화가 이루어지면서 점차 대형화, 복잡화되고 있는 추세이며, 전체 제품에서 SW가 차지하는 비중은 점차 커지고 있는 실정이다. 이에 따라 최종 제품에서 SW 품질은 그 중요성이 점차 증대되고 있으며, 이러한 SW 품질의 중요성은 앞으로 더욱 커질 전망이다. SW Visualization은 이렇게 점차 그 중요성이 커지고 있는 SW 품질 확보에 기여하고, 자동화된 시스템 기반의 편의성을 제공함으로써 품질 기반의 미래형 SW 개발 문화를 위한 기반을 마련한다.

IV

프로세스의 시각화

1. SW Visualization 품질지표
 - 가. 개발 SW에 따른 지표 설정
2. SW Visualization 시스템
 - 가. SW Visualization 프로세스
 - 나. SW Visualization 시스템 구성요소
 - 다. 시스템 구성 절차
3. SW Visualization 구성원 역할
 - 가. Role & Responsibility 정의
 - 나. SW 개발단계별 주요 Activity

IV | 프로세스의 시각화

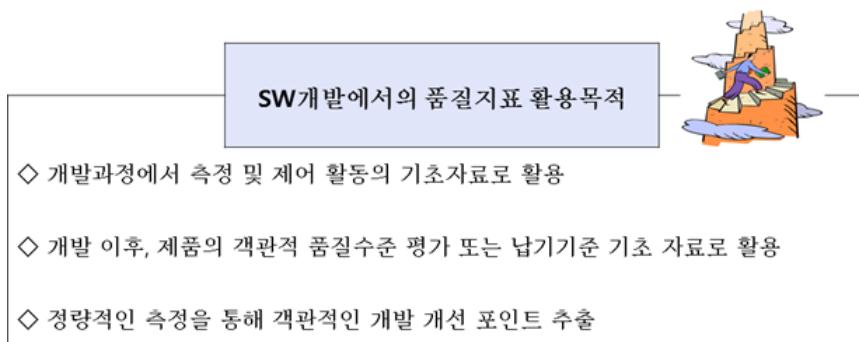
SW개발 프로세스의 시각화는 SW 공학에서 제안하는 개발 단계에 따른 방법론 각각에 대한 부분을 시스템으로 구성하고 이러한 시스템의 활용 과정에서 발생하는 각종 데이터를 종합하여 시각화함으로써, 개발 과정 전반의 상태를 파악할 수 있도록 지원 하며, 개발 과정 전반에 대한 유기적인 연계를 통하여 현재의 개발 상태에 대한 원인을 파악할 수 있도록 하고, 품질 지표를 통하여 현재의 상태가 어느 정도의 품질을 담보할 수 있는지에 대한 정보를 제공하는 것을 목적으로 한다.

1. SW Visualization 품질지표

가. 개발 SW에 따른 지표 설정

➤ 지표 활용목적

SW개발과정을 효과적(Effective)이고 효율적(Efficient)으로 관리하기 위해서는 개발하는 과정과 개발SW에 대한 정량적인 측정을 통해 현 상황에 대한 객관적인 인식과 제어 기초자료가 필요하다.



이는 개발SW유형에 적절한 지표를 선별하고 적용함으로써 개발SW의 객관적인 현 품질수준을 평가 또는 납기기준의 기초자료로써의 활용할 수 있게 됨은 물론, 품질목표 달성을 위한 SW개발 품질관리와 SW개발 개선 포인트를 도출할 수 있게 된다.

이러한 지표는 최대한 간단히 수집이 가능하고 쉽게 이해관계자가 받아들여 손쉽게 품질관리에 활용할 수 있는 지표를 선택하는 것이 무엇보다 중요하다.

※ 도구를 활용한 지표수집 및 현황 분석방법은 ‘SW Visualization 도구 활용가이드’에서 상세히 설명하며 본 챕터는 개발SW유형별 활용가능 지표 위주로 설명한다.

➤ 체계적인 품질관리를 위한 SW Visualization 주요 지표

SW개발과정을 효과적(Effective)이고 효율적(Efficient)인 관리하기 위해서는 개발하는 과정과 개발SW에 대한 정량적인 측정을 통해 품질 현 상황에 대한 객관적인 인식과 제어 기초자료가 필요하다.

이는 SW개발과정에 있어서 파악하기 어려운 SW개발과 품질 현황을 다양한 이해관계자(관리자, 발주자, 개발자, QA 등)가 이해하기 쉬운 형태의 정량적 수치로 표현하는 것으로 SW Visualization하기 위한 주요 지표 아래와 같이 개발과정과 품질검증 영역으로 구분하여 제시한다.

구분	항 목	지 표	개 요	주요 활용용도
개발 과정	요구사항	요구사항 추적성	<ul style="list-style-type: none">개발공정별로 요구사항의 일관성 유지여부를 확인	요구사항 개발관리 프로젝트 진척도 관리
		요구사항 달성을	<ul style="list-style-type: none">정의된 요구사항이 시스템으로 구현되고 있는 정도	
		요구사항 커버리지	<ul style="list-style-type: none">시스템이 요구사항을 커버하는 정도(요구사항이 개발되고, 테스트되고 있는지의 정도를 평가)	
	형상관리	형상목록	<ul style="list-style-type: none">관리 대상항목은 SW유형 및 특성에 따라 상이함	개발 산출물의 무결성 유지 체계적인 유지보수
		항목 등록율	<ul style="list-style-type: none">관리할 내용이 형상목록으로 설정, 관리되는 지 여부	

IV. 프로세스의 시각화

구분	항 목	지 표	개 요	주요 활용용도
		항목 변경율	<ul style="list-style-type: none"> 형상항목 변경 발생 정도로, 형상항목 관리 안정도를 파악 	
동적 테스트	요구 사항 검증	기능 검증	<ul style="list-style-type: none"> 요구된 기능 구현의 완전성 및 정확성 여부를 확인 ※ 적용하는 테스트 설계수준에 따라 확인 수준이 상이함 	요구사항 결합 검출 정량적 품질목표 달성여부 확인(지각된 품질 확보)
		비기능 검증	<ul style="list-style-type: none"> 요구된 정량적 품질목표 달성을 여부를 확인 ※ 비기능 검증항목과 품질목표는 SW유형 및 특성에 따라 상이 	
		정형 검증	<ul style="list-style-type: none"> 높은 수준의 SW신뢰성이 요구되는 경우, 테스트로는 모든 조건의 확인이 현실적으로 불가능함으로 수학적, 논리적 모델을 기반으로 	SW신뢰성 확보
		사용자 검증	<ul style="list-style-type: none"> 고객의 시스템 인수 또는 불특정 잠재고객으로부터의 품질피드백 수행 (알파/베타테스트) 	고객 만족도 극대화
품질 검증	구조 검증	구조적 커버리지	<ul style="list-style-type: none"> 요구되는 신뢰성(테스트) 수준에 따라 Basic Path, Statement, Branch, MD/DC Coverage 등을 선택, 적용 ※ 실제 동작하는 코드의 구조적 커버리지 만으로는 기능·비기능 요구사항 간 연계성 확보여부의 측정이 어려움으로 요구사항 커버리지를 함께 사용하는 것이 바람직함 	코드 결합 검출 SW 신뢰성 확보 테스트 케이스 적절성 판단
정적 테스트		코딩 표준 준수율	<ul style="list-style-type: none"> SW개발에 적용하는 코딩 표준에 대한 준수율 여부를 확인 ※ 미리 정의된 Coding 표준(Rule)에 따른 코드 작성여부 점검 	SW의 잠재적 결합 검출
		메트릭 만족율	<ul style="list-style-type: none"> 자동화 도구 등을 활용, 메트릭을 측정, 메트릭 기준의 만족여부 확인 	SW코드의 가독성, 유지보수성, 테스트 가능성 향상
		정적분석 이행율	<ul style="list-style-type: none"> SW를 실행하지 않은 상태에서 SW의 잠재적 결합을 검출하는 분석 	
결합수준		결합 조치율	<ul style="list-style-type: none"> 결합처리 상태 파악으로 프로젝트의 가시적 문제해결 추진 	현 품질수준에 대한 지각
		결합 밀도	<ul style="list-style-type: none"> 산출물별/영역별/전체 등의 결합분포도를 확인, 결합예방 및 통제 	릴리즈 의사결정 데이터 활용

➤ 개발SW 유형별 SW Visualization 활용지표 사례

개발SW유형은 크게 (1) SW R&D형 프로젝트, (2) SI형 프로젝트, (3) 패키지 SW형 프로젝트, (4) 고신뢰SW형 프로젝트로 구분하여 사례를 제시한다.

이러한 지표와 지표별 목표 수준은 실제 적용 프로젝트 특성을 고려하여 설정할 필요가 있으며 고려해야 하는 프로젝트 특성으로는 계획하였거나 시장에서 요구되는 품질목표 수준, 개발SW의 목표시장 등이 있을 수 있다.

◎ SW R&D형

- ◊ SW기초기술, 플랫폼, 新시장 창출을 위한 서비스 구축 등의 분야에서 창의력을 기반으로 한 목표기술 수준 달성을 위한 활동으로
- ◊ 연구개발의 목표가 상용화 수준을 달성하는 것은 아니라 연구개발로써의 목표 달성을 위한 품질관리 가 필요.
- ◊ R&D 현장의 창의력을 저감시키지 않는 수준에서 R&D 목표달성을 점검하고 지속적인 R&D 추진동력을 마련해야 하며
- ◊ 이를 위해 ‘R&D 컨셉→도출컨셉에 따른 요구사항 도출→기술 개발기술→테스트’에 대한 품질관리를 위한 지표활용이 필요함



◎ SI형

- ◇ 대다수, 명확한 사용자 및 환경에서 한정된 기간 내 개발완수하고 지속적인 유지보수 기반을 마련해야 한다는 특징이 있음
- ◇ 다양한 시스템 연동에 따른 상호운영성 확보, 다수 사용자 및 대용량 정보처리능력, 비즈니스 데이터의 무결성, 사용자 오·조작 방지와, 발생가능 다양한 운영조건에서의 안정성은 물론 대내외 침입으로의 보안성을 확보해야 함
- ◇ 발주자 만족도 극대화를 위해 요구사항에 대한 Consensus는 물론 실 운영환경에서의 인수테스트 수행여부가 중요



◎ 패키지 SW형

- ◇ 특정 또는 매스마켓을 대상으로 다양한 사용자 환경에서 동작하는 SW로 신규제품의 경우 기능 충분성과 사용자 만족도가, 기존 보유 제품은 CSR(Customer Service Request) 및 다양한 제품/버전 형상관리를 통한 품질유지와 다양화 능력이 중요함
- ◇ 불특정 다수 고객 사용으로 인한 오·조작 방지, 다양한 환경 적응



성이 중요하고 기존 SW기반으로 확장되는 경향이 높음으로 SW코드 자체의 품질 또한 중요한 요소임

- ◇ 또한 다양한 배포(온라인, 패키지)경로를 통한 설치/유지보수, 다양한 잠재 사용자를 통한 베타 테스트 등을 적극 활용이 요구됨

◎ 고신뢰SW형

- ◇ 고신뢰 SW(항공, 자동차, 의료, 철도SW 등) 수요가 급격히 증대하고 있으나 SW 특성상 결함은 최소화되어야 하며 결함이 발생하는 경우에도 사람의 생명이나 재산상 피해 등을 입힐지 않도록 안전요구사항이 필수적으로 구현되어야 함
- ◇ 매우 높은 품질요구에 따라 전 개발과정에서 철저한 개발 산출물의 추적관리 및 검증, SW코드 신뢰성 확보, 개발단계별 테스트는 물론 SW코드구조에 대한 테스트도 수행해야 하며 실제 운영환경에서의 철저한 검증이 매우 중요함



* 이해관계자별 구체적인 활동과 관련 지표의 상세 내용은 “SW Visualization 구성원 역할”에서, 각 지표기준 설정방법과 SW Visualization 도구 활용방법은 “SW Visualization 시스템”에서 제시한다.

상기에서 언급한 4개 개발SW유형별 SW Visualization 지표의 활용사례는 아래와 같으며 이는 예시를 위한 것으로 프로젝트에서의 SW Visualization을 위한 활용지표와 지표별 목표수준은 프로젝트 특성을 고려하여 설정하여야 한다.

〈품질관리 지표의 SW유형별 활용사례〉

구분	항목	지표	SW유형			
			원천 SW R&D	SI	Package	고신뢰SW
개발 과정	요구사항	요구사항 추적성	SRS, HLD, System 테스트 케이스	SRS, HLD, Integration/Sy stem 테스트 케이스	SRS, HLD/LLD, Integration/Sy stem 테스트 케이스	SRS, HLD/LLD, Code, Integration/Syste m 테스트 케이스
		요구사항 달성을			100%	
		요구사항 커버리지			100%	
	형상관리	형상항목	SRS, HLD, Code, 테스트 케이스	SRS, HLD, Code, 테스트 케이스, 운영가이드, 사용자 가이드	SRS, HLD, Code, 테스트 케이스, 매뉴얼	SRS, HLD, LLD, Code, Object, 테스트 케이스
		항목 등록율			100%	
		항목 변경율	20%	15%	10%	3%
품질 검증	동적 테스트	기능 검증	100% (주요기능)	100%(주요/부가기능, Positive/Negative Cases)		100%(주요/부가 기능) + Fault Injection Testing
		비기능 검증	성능	성능, 볼륨, 스트레스, 보안, 복구, 상호운영성	성능, 사용성, 구성, 설치, 공존성	성능, 보안, 상호운영성, 유지보수성, 설치
		정형 검증	N/A	N/A	N/A	주요상태 100%
		사용자 검증	N/A	인수 테스트	베타 테스트 (CBT, OBT)	운영환경 테스트(OT, FT)
	구조 검증	구조적 커버리지	N/A	N/A	Statement Coverage > 60%	Statement Coverage > 80% && MC/DC > 60%
	정적 테스트	코딩 표준 준수율	N/A	발주자(전자정부 프레임워크) 및 자체 코딩표준	자체 코딩표준 100%	자체 코딩표준 100% && 산업표준 100%

구분	항목	지표	SW유형				
			원천 SW R&D	SI	Package	고신뢰SW	
		메트릭 만족율	N/A	순환 복잡도 < 30	순환 복잡도 < 25	순환 복잡도 < 15 && 선형복잡도 < 10	
		정적분석 이행율	N/A	주요 모듈 100% && 신규/변경모듈	주요 모듈 100% && 신규/변경모듈	전체 모듈 100%	
결합수준		결합 조치율	치명적인 결합=0. 주요 결합=0	치명적인 결합/주요결합=0, 일반결합=0	치명적인 결합/주요결합/ 일반결합=0, 사소한 결합=0	치명적인 결합/ 주요결합/일반결합/ 사소한 결합=0, 개선사항=0	
		결합 밀도	N/A	0.4/FP	0.15/FP	0.008/FP	

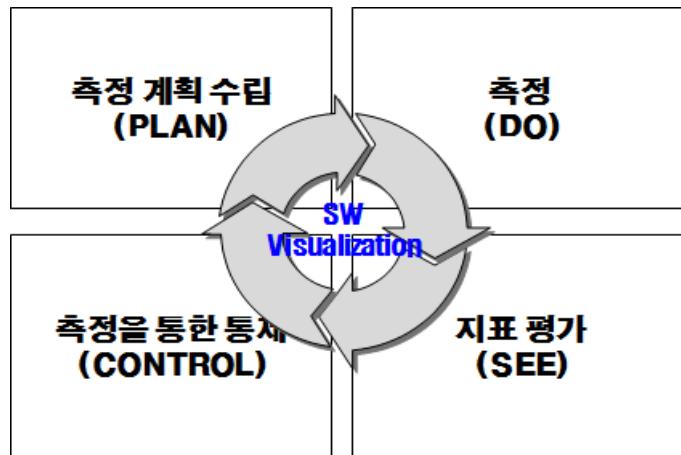
▶ SW개발과정에서의 품질지표 활용절차

지표는 SW개발 초기(착수단계에서)부터 필요 지표를 선별하고 수집방법, 주기 등을 사전에 계획하고 지속적으로 진행 상태를 추적하여야 한다.

측정 시에는 수집하고 싶은 것이 아니라 명확한 사전 목표에 따라 필요한 타당한 Data를 수집하고 어떤 상황을 해석 또는 분석함에 있어서는 하나 이상의 지표를 활용하여 의미 있게 타당한 해석을 수행하는지를 점검할 필요가 있다. 이 때, SW개발상의 약점을 식별하기 위해서는 문제점들을 그룹화 하는 것이 도움이 될 수 있다.

이러한 지표를 체계적으로 활용함으로써 SW개발과정에서의 평가와 통제 영역의 문제를 해결을 위한 기술로서의 활용이 가능하다.

- ※ SW개발과정에서의 이해관계자별 품질지표 활용절차 및 활용방안은 ‘절차 및 활용방안은 SW 개발단계별 주요 Activity’에서 기술한다.



〈품질지표의 활용절차〉

단계별 고려사항은 아래와 같다.

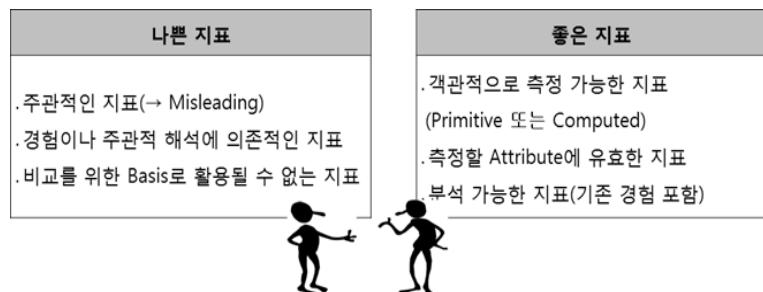
단계	고려사항
계획	<ul style="list-style-type: none"> · 품질활동 관련지표 측정방법(필요지표 선별, 수집방법, 주기 등) 결정 · 6하 원칙을 준수
측정	<ul style="list-style-type: none"> · 계획을 준수하기 위한 관련 개발활동을 실시 · 선별 지표를 통해 준수여부 및 오차발생을 측정
통제	<ul style="list-style-type: none"> · 계획대비 오차가 발생 시, 문제점을 파악하고 개선방안을 도출 · 개선방안 반영을 통한 개발활동 재추진
평가	<ul style="list-style-type: none"> · 활동실적 및 오차를 측정 · 활동실적 및 오차에 대한 분석을 실시

➤ 지표 활용원칙

구분	Principles
1	<ul style="list-style-type: none"> · ‘무엇을 측정한다’가 아니라 ‘왜 측정하는가’를 가지고 이해 관계자를 설득 · 요구사항에 부합되는 Metric을 선정
2	<ul style="list-style-type: none"> · 가장 중요하고 효과가 높은 소수 Metric을 선정, 효과를 본 후 확산
3	<ul style="list-style-type: none"> · 한정된 Metric의 Set으로 시작하고, 천천히 수립
4	<ul style="list-style-type: none"> · 되도록이면 간결한 Metrics 사용
5	<ul style="list-style-type: none"> · 간단히 수집 가능하여, 쉽게 받아들일 수 있는 Metric을 선택(SW Visualization 사용 권장)
6	<ul style="list-style-type: none"> · Metric 표현이 복잡한 통계기법 또는 모델 사용은 되도록 회피 · Distribution Chart, Aging Chart, Trend Chart 적극 활용
7	<ul style="list-style-type: none"> · 무조건적으로 1개의 지표를 가지고 해석, 판단해서는 절대 안 됨
8	<ul style="list-style-type: none"> · 정확한 Data 측정, 분석을 위해 측정대상, 시기 및 방법을 정확히 해야 함
9	<ul style="list-style-type: none"> · 지표를 통한 구성원에 대한 평가를 절대 금물 ! 평가순간 데이터는 왜곡된다.
10	<ul style="list-style-type: none"> · Metric 평가 시, Metric 수집대상 조직을 참여시켜라
11	<ul style="list-style-type: none"> · 목표 만족 또는 개선동기 유발을 위해서는 지표를 활용한 개선을 함께 시도

➤ 지표 선택 시의 고려사항

지표는 되도록 간결하며 목적이 명확해야 하며 많은 수의 지표를 한꺼번에 활용하기보다는 초기에는 한정된 지표 Set으로 효과를 본 후 다수의 Metric으로 확산, 적용하는 것이 바람직하다.



➤ 관련 용어



- **Attribute(속성)** : 객체에 대한 측정 가능한 물리적 또는 추상적 성질
(예) Reliability, Usability, Understandability……
- **Measurement(측정)** : 객체 속성에 척도(Scale)를 배정하기 위하여 메트릭을 사용
- **Metric(메트릭, 지표)** : SW 기능 또는 품질과 관계된 특성(characteristic)이나 특징(feature)을 표현하기 위한 정량적 지표로, 측정방법과 척도(Scale)를 정의
(예) Number of errors found per person hours expended
- **Measure(측정값)** : 측정에 의하여 객체 속성에 배정된 수나 범주
(예) Number of errors
- **SRS** : SW 요구 사양서(SW Requirement Specification)를 의미
- **HLD** : 상위 수준의 설계(Hight Level Design)를 의미, (예) UML Design
- **LLD** : 하위 수준의 설계(Low Level Design)를 의미, (예) Screen Shot of Design

2. SW Visualization 시스템

SW Visualization 시스템은 SW 개발에 있어, 앞의 “SW Visualization 품질지표”에서 정의한 품질지표를 목표로 하여 SW Visualization 시스템의 활용을 SW 개발에서의 수행 활동으로 정의함으로써 SW 품질을 확보하고자 한다.

SW Visualization은 SW 공학 프로세스를 기초로 각 품질지표를 정의하고 시스템을 구성하며, 각 단계별 개발 방법론 및 Tool 사용에 관한 가이드를 제공한다.

가. SW Visualization 프로세스

SW Visualization의 구성은 SW 공학 프로세스에 기반을 둔다.

SW 개발 프로세스는 개발 방법론에 따라 여러 가지로 구분될 수 있으나 SW Visualization에서는 요구사항 관리, 구현, 테스트, 형상관리의 4가지로 구분하여 개발 프로세스를 구성한다.

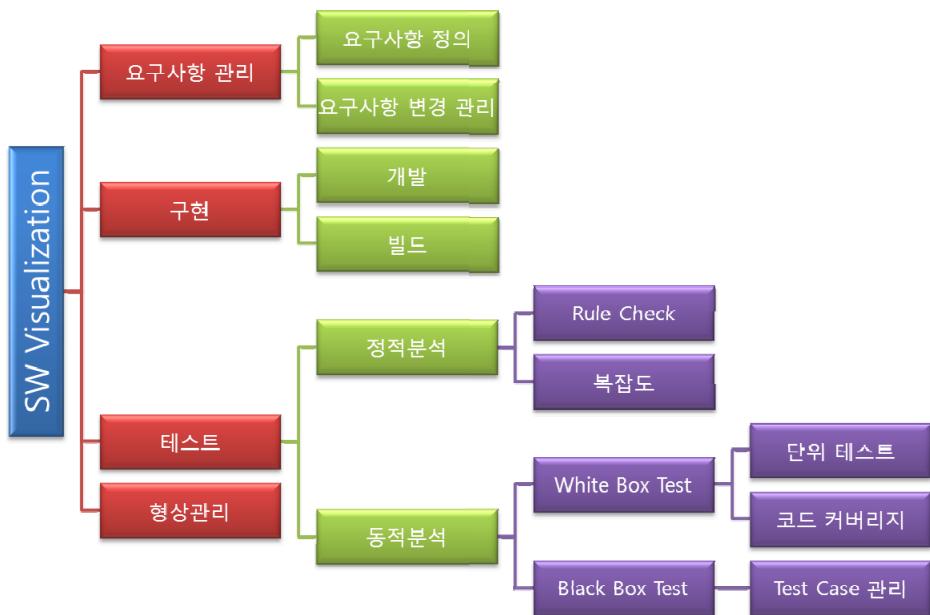


요구사항 관리 단계는 요구사항을 정의하고 요구사항의 변경을 관리하는 과정이 포함된다.

구현 단계는 다시 개발과 빌드의 두 단계로 구분하여 구성한다. 개발은 개발자가 실제로 SW를 구현하는 과정으로 이 과정에서는 개발자의 편의성을 최우선으로 고려한다. 빌드는 개발된 SW의 빌드를 수행하는 과정으로 이 과정은 자동화된 절차에 따라

수행될 수 있어야 하며, 따라서 개발사 각각의 개발 환경에 따라 스크립트와 같은 형태의 자동화된 빌드를 수행할 수 있는 절차가 정의되어야 한다.

테스트 단계는 정적 분석과 동적 분석의 두 단계로 구성하며, 동적 분석은 다시 White Box Test와 Black Box Test의 두 단계로 구분한다. 정적 분석은 개발사에 미리 정의된 코딩 가이드 등의 규칙에 따라 소스 코드를 분석하는 과정과 소스 코드 복잡도를 계산하는 과정이 포함된다. 동적 분석의 첫 번째 구성요소인 White Box Test는 소스 코드의 단위 테스트 과정과 단위 테스트 수행 범위에 대한 코드 커버리지 분석 과정이 포함된다. Black Box Test는 이전의 요구사항 정의 단계에서 정의된 요구사항 각각에 대한 테스트 케이스를 정의하고 테스트 케이스 각각에 대한 테스트를 수행하여 요구사항의 만족도를 확인하며, 필요한 경우 테스트 케이스를 관리하는 과정이 포함된다.



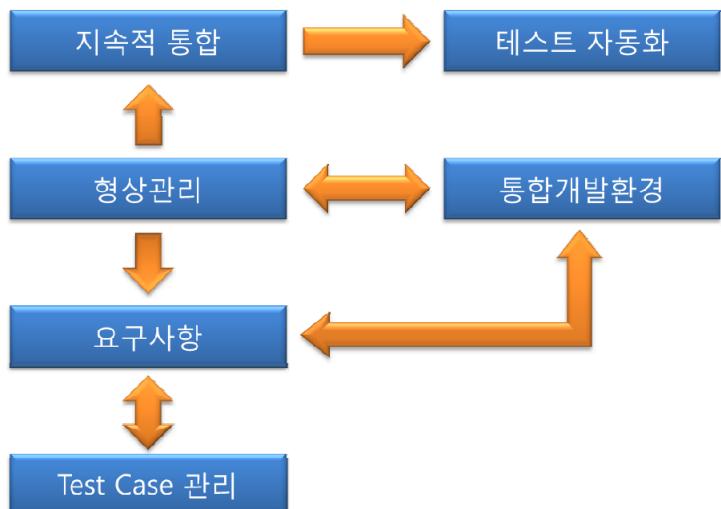
〈SW Visualization 프로세스별 세부 활동〉

끝으로 형상관리 단계는 이전의 요구사항, 구현, 테스트와는 달리 개발 과정 전반에 걸쳐 수행되는 단계로, 개발 산출물을 관리하고 변경을 통제한다. 또한 모든 개발 산출물을 시스템 레벨에서 유지하여 요구사항 관리 단계에서는 정의된 요구사항의 구현과 실제 구현 작업을 연결하는 역할을 수행하고, 개발 단계에서는 개발자의 개발 산출물에 대한 변경을 통제하는 역할을 하며, 빌드 단계에서는 시스템화 된 개발 산출물의 자동화 빌드를 수행할 수 있도록 지원한다.

나. SW Visualization 시스템 구성요소

SW Visualization System은 지속적 통합, 형상관리, 요구사항, 테스트 케이스 관리, 통합개발환경, 테스트 자동화의 각 부분이 상호 연결되어 동작한다. 요구사항 관리 시스템으로 입력된 요구사항에 따라 개발된 산출물은 형상관리 시스템을 통하여 관리되고 형상관리 시스템의 개발 산출물은 지속적 통합을 통하여 지속적으로 빌드된다. 이렇게 빌드된 결과는 테스트 자동화 시스템에 의하여 정해진 시점에 자동으로 테스트되어 현재 SW의 품질을 측정하고 그 결과를 시각화한다. 또한 테스트 케이스 관리 시스템에 의하여 요구사항 대비 구현상태의 검증을 위한 테스트 케이스가 관리되고 검증 결과에 따라 요구사항의 구현률이 시각화된다.

이러한 SW Visualization 시스템이 상호 연결되어 동작함에 따라 SW 품질 시각화의 결과에 대한 원인을 파악하고, 빠른 결함 조치가 이루어질 수 있도록 지원한다.



〈SW Visualization 시스템 구성 요소〉

➤ 지속적 통합 (Continuous Integration)

지속적인 통합(Continuous Integration)은 프로젝트 구성원들이 자신들의 작업 내용을 자주 통합하는 방법이다. 통합이 수행될 때마다 테스트를 포함한 자동화된 빌드 절차에 의해 통합내용은 자동 검증 되며, 이로 인해 통합 시 발생하는 에러를 조기에 발견할 수 있다.

❖ Jenkins

Jenkins 는 반복된 작업의 수행을 모니터하는 응용 프로그램으로 SW 프로젝트의 build 작업이나 cron 으로 동작하는 작업 등을 관리할 수 있다. hudson 개발자들에 의해 개발된 오픈 소스 프로젝트로서 SD Times에서 2012년의 CI 툴로 선정된 바 있다.

Project SWV1

- Language : Java
- Build : Maven
- Test : Sonar

Build History

- #8 2013. 10. 22 오후 2:27:10
- #7 2013. 10. 10 오후 5:09:19
- #6 2013. 8. 28 오전 11:23:34
- #5 2013. 8. 28 오전 11:21:50
- #4 2013. 7. 18 오후 1:19:06
- #3 2013. 7. 16 오후 6:43:57
- #2 2013. 4. 9 오후 2:23:42
- #1 2013. 4. 9 오후 2:22:28

Recent Changes

주요 링크

- Last build, (#8) 6 days 6 hr 전
- Last stable build, (#8) 6 days 6 hr 전
- Last successful build, (#8) 6 days 6 hr 전
- Last failed build, (#3) 3 mo 14 days 전
- Last unsuccessful build, (#5) 2 mo 1 day 전

작성된 글이자: 2013. 10. 28 오후 9:24:43 REST API Jenkins ver. 1.502

localhost:8080/jenkins/job/SWV1/changes

➤ 요구사항 관리 (Require Management)

SW 개발에 있어서 가장 어려운 부분은 무엇을 개발할 것인가를 정확히 결정하는 것이다. 이러한 SW 요구분석은 SW 개발의 실제적인 첫 단계로 사용자의 요구에 대하여 이해하는 단계라 할 수 있다. SW 요구분석 단계는 전체 개발 과정에서 개발 비용을 감소시킬 수 있는 결정적인 단계로, 초기에 요구사항을 잘 분석하여, 정의하고 관리하기 위해 투자한다면 전체 SW 개발 기간과 비용의 초과, 품질저하를 미연에 방지할 수 있다.

❖ Redmine

Redmine은 웹기반의 프로젝트 관리와 버그 추적 기능을 제공하는 오픈 소스 도구이다. 화면 기반의 프로젝트 관리에 도움이 되도록 간트 차트를 통한 달력과 일정관리 기능 및 프로젝트 관리 기능과 이슈추적, 여러 가지 형상관리 기능을 제공한다.



➤ 테스트 케이스 (Test case)

테스트 케이스란 정의된 요구사항대로 SW가 구현되었는지를 확인하기 위하여 테스트 되어야 하는 항목들을 정의한 문서이다. 테스트 케이스에 명시된 사항들에 대한 테스트를 수행함으로써 SW의 품질을 측정할 수 있다.

❖ TestLink

요구사항을 등록하고 등록된 요구사항에 대한 테스트 케이스를 관리할 수 있는 툴이다. 테스트 케이스를 작성하고 Test Plan에 따라 테스트를 수행함으로써 요구사항을 확인할 수 있으며, 각 요구사항에 대한 커버리지를 확인할 수 있으며, 실패한 테스트 케이스에 대하여 결합 내용을 다시 Redmine에 등록할 수 있다.

❖ Impasse

TestLink의 기능을 Redmine에서 사용할 수 있도록 만들어진 Redmine의 plugin이다. 기능은 TestLink와 유사하나 별도로 요구사항을 입력하지 않고 Redmine에 등록된 요구사항에 대한 테스트 케이스를 관리하는 방식으로 동작한다.

➤ 형상관리 (Configuration Management)

SW를 개발하거나 유지 보수하는 과정에서 변경을 통제하는 절차는 SW 개발 과정의 산출물들을 관리하고, 고 품질의 SW를 얻기 위해 매우 중요하다. SW 형상관리는 SW의 개발 및 유지보수 과정에서 발생하는 각종 산출물에 대한 계획, 개발, 운용 등을 종합하여 시스템의 형상을 만들고, 이에 대한 변경을 체계적으로 관리, 제어하기 위한 활동이다.

❖ Subversion

Subversion 은 오픈 소스 버전관리 시스템으로 2000년부터 CVS의 대체를 위해 개발되었으며 현재 아파치 최상위 프로젝트로서 전 세계 개발자 커뮤니티와 함께 개발되고 있다.

➤ 테스트 자동화

지속적 통합은 정해진 특정 시점에 자동으로 소프트웨어를 빌드하여 바이너리를 생성한다. 이렇게 생성된 바이너리는 다시 테스트 자동화 시스템에 의하여 자동으로 테스트 되며, 그 결과는 시각화된 형태로 표현되어 현재 SW의 상태에 대한 명확한 인지와 개선을 위한 정보를 제공하는 것을 목적으로 한다. 테스트 자동화의 방법은 크게 적정분석과 동적분석으로 구분할 수 있다.

정적분석은 SW를 실행시키지 않고 소스코드 자체를 분석하는 것이다. 프로그램에 내재한 논리적 오류는 보통 SW를 실행하여 확인하지 않으면 찾기가 힘들지만, 정적분석은 이러한 오류를 찾아내는 데 도움을 줄 수 있다. PMD, CPPCheck 등을 활용한 Rule Check와 NSIQ Collector를 활용한 코드 복잡도 체크는 대표적인 정적분석의 예이다.

Rule Check는 미리 정의된 방법에 따라 소스코드가 작성되었는지를 검사하는 방법이며, Complexity는 소스코드의 복잡도를 계산함으로써 SW가 지나치게 복잡해짐으로써 발생할 수 있는 오류를 미연에 방지하고, 이후 유지보수를 쉽게 할 수 있는 방법이다.

동적 분석은 프로그램에 여러 가지 입력을 주며 실행시켜서 그 결과 값을 이용하여 프로그램의 정상 동작 여부를 판단하는 방법으로, 프로그램을 실행시키지 않고 테스트하는 정적 분석(Static Analysis) 과 대비되는 테스트 방법이다. 동적분석의 대표적인 방법은 단위테스트 및 단위테스트에 대한 커버리지 확인이 있다.

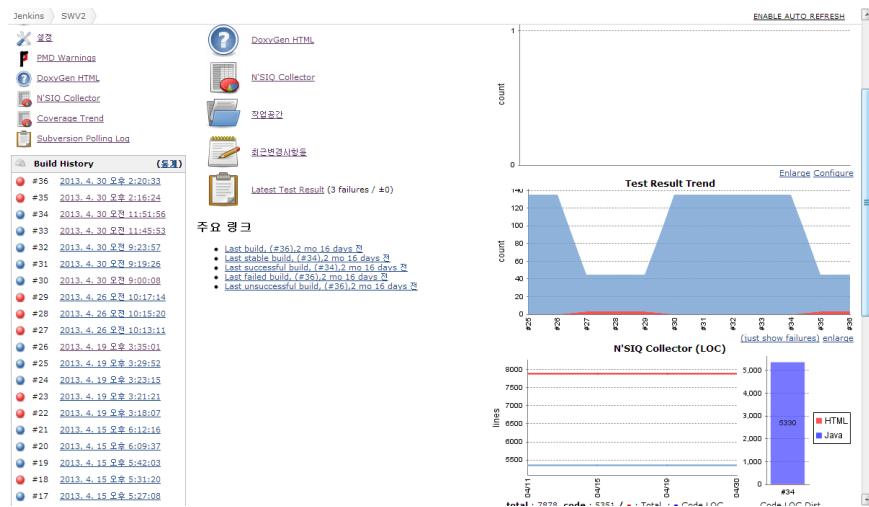
단위 테스트는 테스트를 위한 코드를 개발자가 직접 작성하여 해당 코드를 실행함으로써 테스트 대상이 되는 코드에 대한 특정 영역을 실행해보는 테스트 방법이다. 코드

커버리지는 단위 테스트를 통하여 수행된 테스트가 전체 코드에 대하여 어느 정도의 테스트를 수행했는지를 확인함으로써 단위 테스트의 완성도를 측정하는 방법이다.

❖ Jenkins

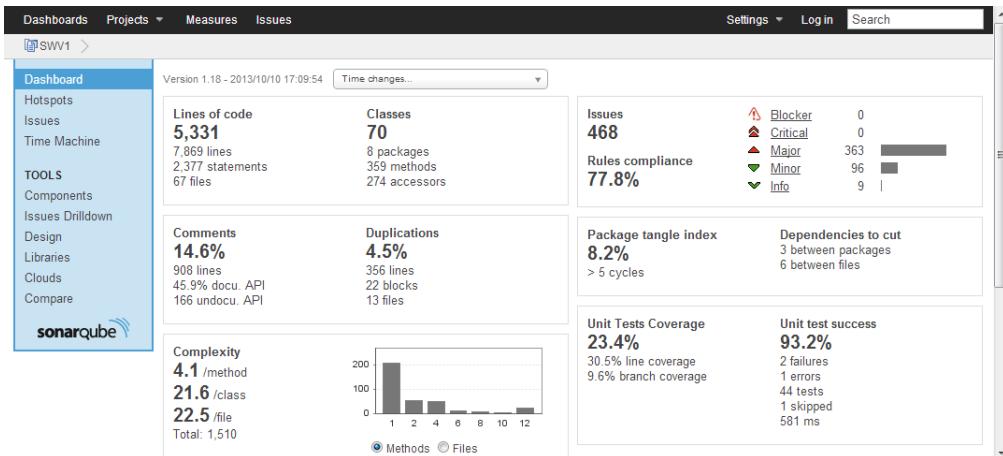
Jenkins는 앞서 빌드를 위한 지속적 통합으로 언급하였다. 그러나 Jenkins에 테스트를 위한 개별 plugin을 설치함으로써 정해진 시점(일반적으로 빌드가 완료된 직후)에 정해진 테스트가 수행되도록 구성이 가능하다.

			System
정적분석	Java 환경	Rule Check	PMD
		Complexity	NSIQ Collector
	C/C++ 환경	Rule Check	CPPCheck
		Complexity	NSIQ Collector
동적분석	Java 환경	Unit Test	JUnit
		Coverage	Emma
	C/C++ 환경	Unit Test	CPPUnit
		Coverage	gcov



❖ SonarQube

Jenkins와 동일하게 SonarQube에 별도 plugin을 설치하여 Jenkins와 연계함으로써 Jenkins에서 SW 빌드를 수행한 후, SonarQube에서 자동화 테스트를 수행하는 방식으로 동작한다. Jenkins에 비해 다양한 개발언어를 지원하나 개발 언어에 따라 유료 plugin이 존재한다.



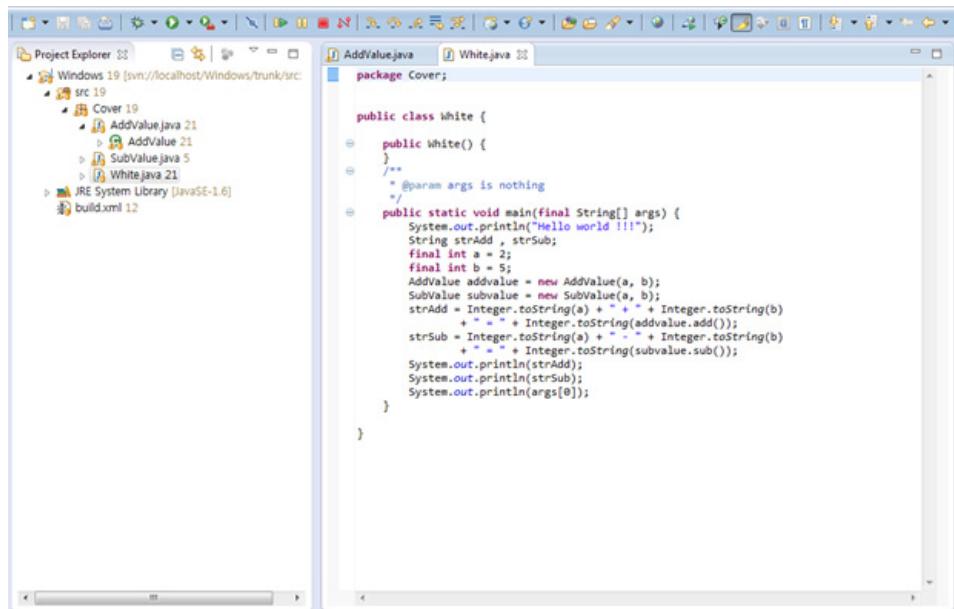
➤ 통합 개발 환경 (Integrated Development Environment)

통합 개발 환경은 요구사항 관리에서부터 실제 SW의 개발과 개발한 소스코드의 형상관리까지 개발자 입장에서 필요한 전체 과정을 통합된 시스템에서 작업할 수 있도록 구성한 시스템이다.

❖ Eclipse

Eclipse는 오픈 소스이며 자바 기반의 확장 가능한 개발 플랫폼이다. Eclipse가 자바로 만들어졌지만 자바 언어에만 국한된 것은 아니다. 예를 들어 플러그인도 사용할 수 있고 C/C++, COBOL, Eiffel과 같은 프로그래밍 언어에 대한 지원도 포함하고 있다.

SW Visualization에서는 Eclipse를 중심으로 Subclipse를 이용하여 형상관리 시스템과 연동하고, Mylyn을 이용하여 요구사항 관리 시스템과 연동함으로써 개발자는 모든 작업을 Eclipse 내에서 수행할 수 있도록 지원한다.



➤ 기타 플러그인

	System
Eclipse, Redmine 연계	Mylyn
Eclipse, Subversion 연계	Subclipse

본 문서의 시스템 구성에서는 개발의 편의를 위하여 통합개발환경인 Eclipse에 Mylyn과 Subclipse를 추가 구성하여 요구사항 관리 및 형상관리 시스템과의 연동이 가능하도록 구성한다.

❖ Mylyn

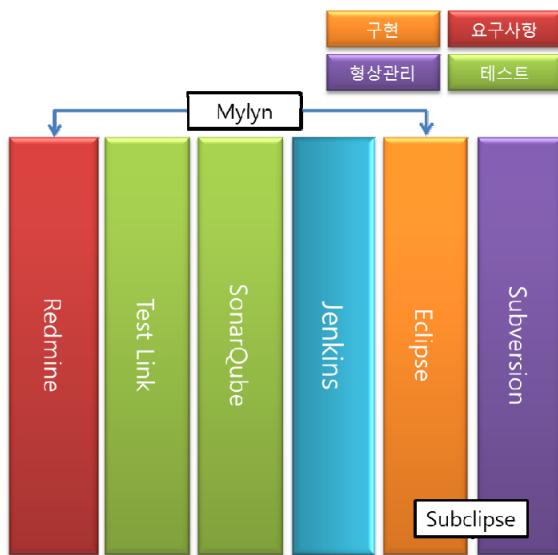
Mylyn은 Task 관리를 위한 Eclipse plugin 형태의 프로젝트로서, Jira, Redmine 등의 이슈트래커와 연동하여 Eclipse에서 Task의 관리를 가능하게 하는 plugin이다. Mylyn의 사용을 위해서는 Eclipse에 plugin을 설치하고 Redmine이 동작하는 서버 측에도 Mylyn 사용을 위한 plugin의 설치가 필요하다.

❖ Subclipse

Subclipse는 버전관리 시스템인 Subversion과 연동을 위한 Eclipse plugin이다. Subclipse를 사용하여 Eclipse에서 Subversion에 등록된 Repository에 접근이 가능하며, 따라서 Eclipse에서 개발한 코드를 바로 Subversion에 적용할 수 있도록 지원한다.

다. 시스템 구성 절차

SW Visualization 시스템의 구성은 기업 및 프로젝트의 현황에 따라 달라질 수 있으나 본 문서에서는 아래 그림과 같은 구조의 구성을 위한 절차를 설명한다. 시스템 구성은 아래 과정에 따라 설치를 진행하여야 하며 개별 시스템의 상세한 사용방법은 정부통신산업진흥원 SW공학센터에서 운영하는 「SW뱅크」 시스템(<http://swbank.kr>)에서 개발 도우미 메뉴의 SW개발도구 항목을 참고할 수 있다.



➤ Apache 설치

본 시스템에서 요구사항 관리를 위하여 사용되는 Redmine은 Apache 위에서 동작하는 Web Application이다. 따라서 Redmine의 활용을 위해서는 우선 Apache의 설치가 선행되어야 한다.

그러나 설치의 편의를 위하여 Bitnami Redmine에서는 Apache 및 Subversion, MySQL과 함께 설치를 할 수 있는 패키지를 제공하므로, Apache를 별도 설치할 필요 없이 Bitnami Redmine의 설치를 통하여 Apache의 설치가 가능하다.

Redmine의 설치 방법은 아래 Redmine 설치 절차에 따른다.

➤ MySQL 설치

SW Visualization에서 Redmine은 등록된 요구사항의 관리를 위하여 MySQL을 사용한다.

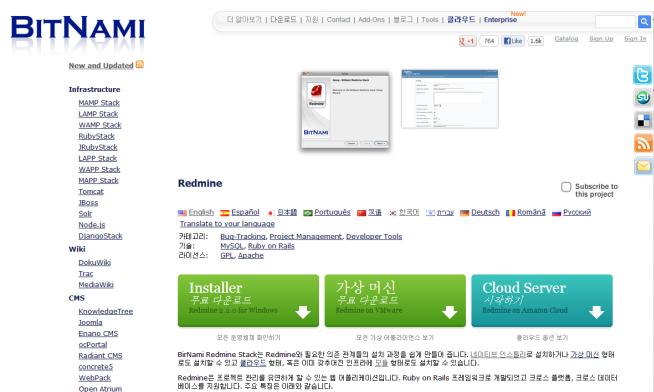
그러나 설치의 편의를 위하여 Bitnami Redmine에서는 Apache 및 Subversion, MySQL과 함께 설치를 할 수 있는 패키지를 제공하므로, MySQL을 별도 설치할 필요 없이 Bitnami Redmine의 설치를 통하여 MySQL의 설치가 가능하다.

Redmine의 설치 방법은 아래 Redmine 설치 절에 따른다.

➤ Redmine 설치

Redmine은 Ruby를 이용하여 개발된 Web Application이다. 따라서 Redmine의 설치를 위해서는 Ruby 및 Rail 등의 설치가 선행되어야 한다. 그러나 Bitnami Redmine에서는 이러한 전체 설치를 통합하여 진행할 수 있도록 패키지 형태의 설치 파일을 제공한다. 따라서 Bitnami Redmine의 설치를 통하여 Redmine 설치를 위한 모든 툴의 설치를 통합하여 진행 가능하다.

※ URL : <http://bitnami.org/ko/stack/redmine>



<Bitnami Redmine Stack 다운로드>

➤ Subversion 설치

본 시스템에서는 Redmine을 요구사항 관리를 위하여 사용한다. 또한 요구사항의 실제 구현 소스코드와 Redmine에 등록된 요구사항의 연결을 위하여 형상관리 시스템인 Subversion과 연동하여 동작한다.

따라서 Redmine의 활용을 위해서는 Subversion이 설치되어야 한다. 그러나 앞서 설치한 Bitnami Redmine 설치 시 Subversion의 설치가 함께 진행되므로 별도의 Subversion의 설치는 필요치 않다. 다만, Subversion의 활용을 위하여 Repository의 정의가 필요하며 그 과정은 아래와 같다.

1. *D:\Webserver\SVNRepo 폴더 생성*
2. *%BitNami Redmine Stack%\Subversion\scripts\service\install.bat REMOVE*
3. *%BitNami Redmine Stack%\Subversion\scripts\service\install.bat 파일 수정*
"D:\BitNami Redmine Stack\subversion\scripts\winserv.exe"
install redmineSubversion -displayname "redmineSubversion"
-start auto "D:\Webserver\BitNami Redmine
Stack\subversion\bint\svnservice.exe" -d --root
"D:\Webserver\SVNRepo" --listen-port=3690
4. *%BitNami Redmine Stack%\Subversion\scripts\service\install.bat INSTALL*

➤ TestLink 설치

테스트 케이스 관리를 위하여 사용되는 TestLink는 Apache 환경에서 동작하는 Web Application이다. 따라서 TestLink의 설치를 위해서는 Apache의 설치가 선행되어야 한다. 또한 TestLink의 실행을 위하여 PHP와 MySQL의 설치가 선행되어야 한다.

그러나 Apache, MySQL, PHP 는 모두 Bitnami Redmine의 설치에 포함되어 있는 부분이므로 본 시스템에서는 별도의 설치 없이 바로 TestLink의 설치를 진행한다.

※ URL : <http://testlink.org>

TestLink

TestLink Open Source Test Management

Get TestLink Stable

[Get it from sourceforge](#)

[Get a Bitnami Virtual Appliance](#)

Access Latest Development Version

[Access Git Repository](#)

Get Connected

[Forum](#)

[Issues/Feature Request](#)

[LinkedIn](#)

[Twitter](#)

© TestLink Development Team 2013

➤ JDK 설치

Jenkins의 동작을 위해서는 Tomcat 과 JDK의 설치가 필요하다. 따라서 JDK가 우선 설치되어야 한다.

JDK는 1.6 버전을 기준으로 설치를 진행한다.

* URL : <http://www.oracle.com/technetwork/java/javase/downloads/index.html>

* Version : Java SE 6

The screenshot shows the Java Platform, Standard Edition download page. It features three main sections:

- Java SE 7u10**: Includes a summary of changes, a "Learn more" link, and links for **JDK DOWNLOAD** and **JRE DOWNLOAD**. Below these are lists of **JDK 7 Docs** and **JRE 7 Docs**.
- JDK 7 and JavaFX Demos and Samples**: Includes a summary of common tasks and source code samples, and a **Demos and Samples DOWNLOAD** link.
- Java SE 6 Update 38**: Includes a summary of security features and bug fixes, a "Learn more" link, and links for **JDK DOWNLOAD** and **JRE DOWNLOAD**. Below these are lists of **JDK 6 Docs** and **JRE 6 Docs**.

To the right of the main content area, there are two sidebar banners:

- Java magazine**: Promotes the magazine with a "Get it now for FREE!" button and a "Subscribe Today" button.
- Java Tutorials**: Promotes Java tutorials with a "We're Hiring! Join the Java Development Team!" message.

〈JDK 다운로드〉

➤ Tomcat 설치

본 문서에서 구성하는 시스템에서는 지속적 통합을 위하여 Jenkins를 사용하며, Jenkins의 동작을 위해서는 Tomcat과 JDK의 설치가 필요하다.

Tomcat의 버전은 6.0 버전을 기준으로 설치를 진행한다.

※ URL : <http://tomcat.apache.org/download-60.cgi>

※ Version : Tomcat 6.0

The screenshot shows the Apache Tomcat 6.0 download page. At the top right is the Apache Software Foundation logo. On the left, there's a cartoon cat icon and a sidebar with links for Home, Taglibs, and Maven Plugin under 'Apache Tomcat', and links for Which version?, Tomcat 7.0, Tomcat 6.0, Tomcat 5.5, Tomcat Connectors, Tomcat Native, Archives under 'Download', and Tomcat 7.0, Tomcat 6.0, Tomcat 5.5, Tomcat Connectors, Tomcat Native, Wiki, Migration Guide under 'Documentation'. The main content area has sections for 'Tomcat 6 Downloads' (with a welcome message), 'Quick Navigation' (with links to KEYS, 6.0.36, Browse, Archives), 'Release Integrity' (with instructions about verifying file integrity using OpenPGP signatures), and 'Mirrors' (with a note about currently using http://apache.tt.co.kr/ and a dropdown menu for other mirrors). A footer bar at the bottom shows '6.0.36'.

〈Tomcat 다운로드〉

➤ Jenkins 설치

본 시스템에서는 지속적 통합을 위하여 Jenkins를 사용한다. Jenkins를 통하여 소스 코드는 정해진 시점에 자동 빌드된다.

Jenkins에서 생성된 프로젝트는 그 프로젝트의 성격 혹은 해당 기업의 상황에 따라 사용되는 빌드 스크립트와 연결 되어야 하며 이를 통하여 Subversion에 등록된 SW는 지속적 통합이 가능하다.

※ URL : <http://jenkins-ci.org>



〈Jenkins 다운로드〉

➤ SonarQube 설치

본 시스템에서는 테스트 자동화를 위하여 SonarQube를 사용한다. Jenkins를 통하여 빌드된 바이너리는 SonarQube에서 자동으로 테스트된다.

SonarQube의 실행을 위해서는 DB가 필요하며 SonarQube가 지원하는 DB는 여러 종류가 있으나, 본 문서에서는 앞서 Bitnami Redmine의 설치 시에 MySQL을 설치하였으므로 별도의 DB를 설치하지 않고 해당 DB를 그대로 활용하고 SonarQube만 설치하는 것으로 한다.

* URL : <http://www.sonarqube.org/downloads/>

The screenshot shows the SonarQube website's download section. At the top, there's a navigation bar with links for Download, Features, Get Support, Get Involved, Development, Roadmap, Blog, and Company. Below the navigation, there's a heading for 'License' which includes a detailed text about the software's distribution under the GNU Lesser GPL License, version 3, and a link to the original license. Under 'Download SonarQube', there's a table listing releases from version 3.6.2 down to 3.5, with columns for Version, Date, Description, Release notes, Screenshots, Download, and MD5. A 'Show all releases' link is at the bottom. Below that is another section for 'Download SonarQube Runner' with a similar release table for versions 2.3 down to 2.2, also with a 'Show all releases' link.

Version	Date	Description	Release notes	Screenshots	Download	MD5
3.6.2	Jul. 18, 2013	Fix issue on Oracle DB migration.	Release notes	Screenshots	Download	MD5
3.6.1	Jul. 12, 2013	Faster database migration, security fix and other bug fixes	Release notes			
3.6	Jun. 26, 2013	Violations become issues and come with new features: search engine, change log, etc., tracking of new coding rules, quality profile administrator, highlighting of usage and declaration of variables/functions in source code viewer	Release notes			
3.5.1	Apr. 3, 2013	Fix two regressions when using SonarQube Runner to perform analyses in specific cases	Release notes	Screenshots	Download	MD5
3.5	Mar. 13, 2013	Tracking of unit tests, new rules on unit tests, new exclusion settings, enhanced update center, enhanced email notifications	Release notes			

[Show all releases](#)

Version	Date	Description	Release notes	Download	MD5
2.3	Jul. 23, 2013	Technical release: API improvements	Release notes	Download	MD5
2.2.2	Jun. 7, 2013	Fixes an issue while launching a task on Windows	Release notes	Download	MD5
2.2.1	May 21, 2013	Fixes issue with sym links	Release notes		
2.2	Apr. 18, 2013	Provides a new API that allows to fork the process when launching a SonarQube analysis	Release notes		

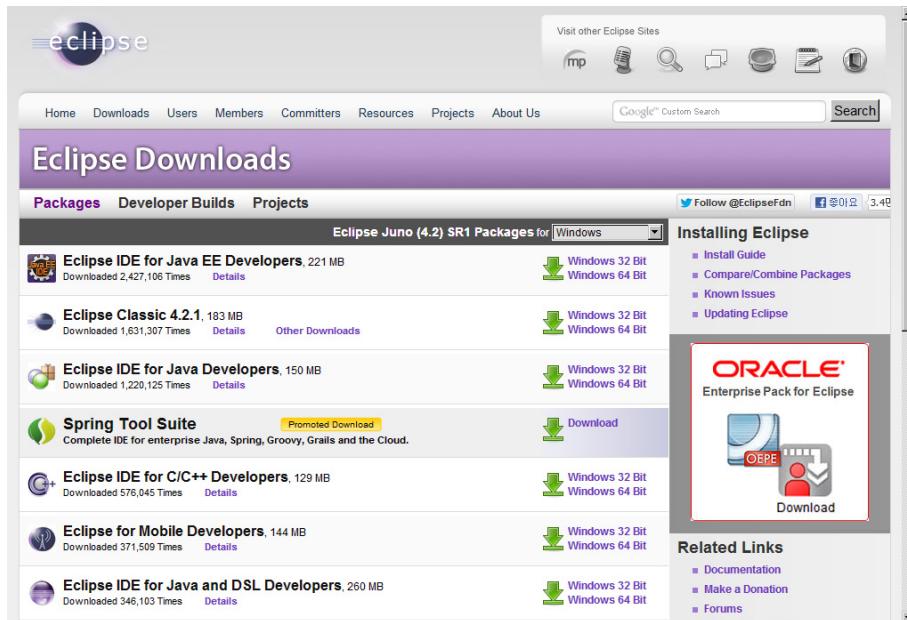
[Show all releases](#)

〈SonarQube 다운로드〉

➤ Eclipse 설치

본 문서에서 구성하는 시스템에서는 SW 개발을 위하여 Eclipse를 사용한다. Eclipse는 일반적으로 Java 개발에 사용되나 Eclipse C/C++ 혹은 CDT plugin 설치를 통하여 C/C++ 개발에도 활용이 가능하다.

* URL : <http://www.eclipse.org/downloads/>



〈Eclipse 다운로드〉

➤ Plugin 설치

Eclipse는 Subversion, 혹은 Redmine과의 연계를 위하여 여러 plugin을 활용한다. 본 절에서는 이러한 plugin의 설치에 대하여 설명한다.

❖ Mylyn 설치

최신 버전의 Eclipse에는 Mylyn 플러그인이 기본으로 설치되어 있다. 따라서 일반적으로 Eclipse에는 plugin을 별도 설치할 필요가 없으며, Redmine이 설치된 서버에 Mylyn Connector plugin의 설치가 필요하다.

Windows의 시작 버튼, Bitnami Redmine Stack, Bitnami Redmine Stack을 선택하여 입력 콘솔을 활성화 시킨다.



〈Redmine 입력콘솔 활성화〉

활성화 된 입력창에서 다음과 같이 입력하여 Mylyn Connector plugin을 설치한다.

```
ruby script/plugin install git://redmine-mylyncon.git.sourceforge.net/gitroot/
redmine-mylyncon/redmine-mylyn-connector
```

A screenshot of a terminal window titled "BitNami Redmine Stack Environment". The command "ruby script/plugin install git://redmine-mylyncon.git.sourceforge.net/gitroot/redmine-mylyn-connector" is entered and its output is displayed. The output shows the process of cloning a Git repository from the specified URL into the Redmine stack's plugin directory. It includes details like object counting, compression, receiving objects, and resolving deltas.

〈Mylyn Connector plugin 설치〉

Redmine에서 plugin의 설치 여부를 확인할 수 있다.

The screenshot shows the Redmine application interface. On the left, there's a sidebar with various management links like '프로젝트', '사용자', '그룹', etc. The main content area is titled '플러그인' (Plugins) and lists several installed plugins:

- Redmine Graphs plugin** by Brad Beattie, version 0.1.0
- Redmine Impasse plugin** by kawasima, version 1.2.2 (with a '설정' link)
- Mylyn Connector plugin** by Sven Krzyzak, version 2.7.6.stable

A note below the list states: "This plugin provides a webservice API for Eclipse Mylyn (RESTful)".

〈Redmine에서 Mylyn plugin 설치 확인〉

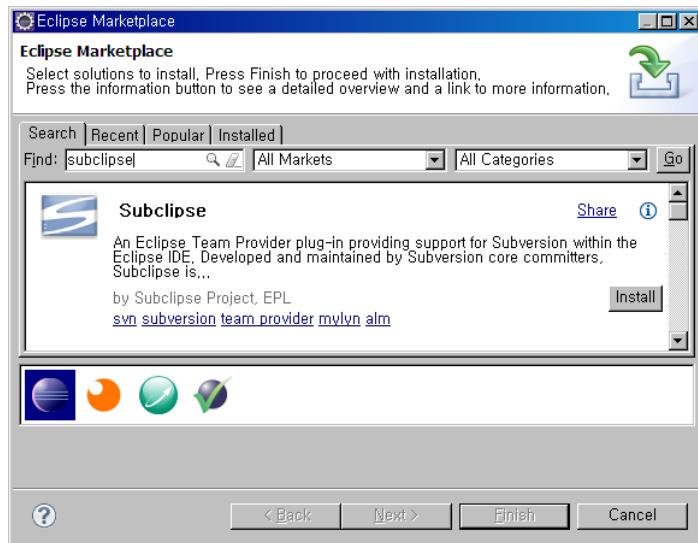
Redmine의 설정에서 REST 웹서비스 활성화 항목을 체크한다.

The screenshot shows the Redmine '설정' (Settings) page. On the left, there's a navigation bar with tabs like '일반', '표시방식', '인증', '프로젝트', etc. The main content area is titled '설정' and contains several configuration options. One of the checkboxes is specifically for 'REST 웹서비스 활성화' (REST Web Services Enabled), which is checked. Other visible settings include password requirements, account deletion, and OpenID integration.

〈Mylyn 사용을 위한 Redmine 설정〉

❖ Subclipse 설치

Eclipse의 Help, Eclipse Marketplace에서 subclipse를 설치한다.



〈Subclipse plugin 설치〉

3. SW Visualization 구성원 역할

가. Role & Responsibility 정의

효율적이고 체계적인 SW개발을 위해서는 각 구성원의 역할과 책임이 명확히 정의되고 이행되어야 한다. 따라서 현재 조직의 상황을 고려하여 아래와 같이 구성원의 역할을 수행할 담당자를 선정하여 각자가 맡은 책임을 이행할 수 있도록 교육을 제공하고 기반환경 조성을 하도록 해야 한다.

각 구성원의 담당자는 반드시 독립적인 역할을 해야 하는 것은 아니지만, 엄격한 품질관리 및 통제를 위해서 품질관리 담당자(QA)는 가급적 독립적인 역할을 수행할 수 있도록 개발팀(설계/테스트/개발담당자)과는 별도로 담당자를 선정하도록 하는 것이 바람직하다.

구성원(Role)	Responsibility
비즈니스 분석가 (BA, Business Analyst)	<ul style="list-style-type: none"> 비즈니스 환경, 기술트렌드, 잠재적 수요자 요구사항 등 분석 비즈니스 목표 및 상위 수준의 비즈니스 요구사항 도출 비즈니스 목표/요구사항에 기반하여 세부 SW, 시스템 요구사항 도출 프로젝트 전반에 걸쳐 비즈니스 목표·요구사항에 적절하게 개발이 되고 있는지 확인
품질관리 담당자 (QA, Quality Assurance)	<ul style="list-style-type: none"> 비즈니스 목표/요구사항에 따라 품질목표 수립 관련 Constraints, Regulations, Standards 등이 도출되고 개발에 반영되고 있는지 확인 품질보증 계획을 수립하고, 이행을 위한 준비·교육 제공 개발 단계 말 품질보증을 위한 프로세스 이행여부 확인 및 산출물 품질 검토 인수 및 릴리즈 충족성 기준 수립 및 충족여부 평가보고
설계 담당자 (Architect)	<ul style="list-style-type: none"> 비즈니스 목표/요구사항, 기술트렌드 등을 고려한 적절한 아키텍처 전략 수립 세부 요구사항(기능·비기능)을 충족하는 SW 아키텍처 설계 및 검증 아키텍처에 따른 상세 설계 및 구현의 적절성 확인

구성원(Role)	Responsibility
테스트 담당자 (Tester)	<ul style="list-style-type: none"> · 테스트 전략 및 총괄 테스트 계획 수립 · 요구사항 분석 지원(Testability 분석) · 테스트 환경구축(테스트 도구, 테스트데이터 등) · 테스트 설계(테스트 시나리오, 테스트케이스) 및 이행 · 테스트 결과 분석·보고
SW 개발 담당자 (개발자)	<ul style="list-style-type: none"> · 정의된 요구사항, 설계에 따라 코드 구현 · 구현에 따른 검증(정적분석, 화이트박스테스트, 단위 테스트 등)

나. SW 개발단계별 주요 Activity

개발단계에 따라 이행할 주요 Activity 및 각 구성원의 역할은 다음과 같다.

단계	Activity	구성원					관련지표	도구
		QA	BA	아 카 텍 트	테 스 터	개 발 자		
착수 / 계획	<ul style="list-style-type: none"> · High Level Requirements 분석·정의 <ul style="list-style-type: none"> - Business Requirements - 관련 Standards, Constraints - 품질목표, 관련지표 Define 	○	◎				-	-
	<ul style="list-style-type: none"> · Business Requirements, Goal Review 	○	◎	○	○	○	-	Redmine
요구 분석	<ul style="list-style-type: none"> · Low Level Requirements 분석·정의 <ul style="list-style-type: none"> - System, SW Requirements - Testability, Priority 분석 및 정의 	○	◎		○	○	-	-
	<ul style="list-style-type: none"> · Requirements Review 	◎	○	○	○	○	✓ 추적성 수준 ✓ 요구사항 커버리지	Redmine

단계	Activity	구성원					관련지표	도구
		QA	BA	아 카 텍 트	테 스 터	개 발 자		
설계	· SW Design (Architecture, Interface, Class 등)			◎		○	-	-
	· 테스트 케이스 Design (System, Integration 테스트 케이스)				◎	○	-	Redmine
	· Design Review (Architecture, 테스트 케이스 등 상세 설계) ※ 고신뢰SW는 model checking 포함	◎	○	○	○	○	✓ 추적성 수준 ✓ 요구사항커버리지 ✓ 정형검증(모델체킹)	Redmine
구현	· SW Code 구현					◎	-	IDE (Eclipse)
	· 구현된 Code에 대한 Static Analysis (Coding Rule Check, Complexity 산출 등)				○	◎	✓ 정적분석 이행율 ✓ 코딩표준 준수율 ✓ 매트릭 만족도	SonarQube
	· Code Review	○				◎	-	-
테 스 트	· White Box Test 이행 (구조 검증 Focus)	○			○	◎	✓ 구조적 커버리지 (Statement, MC/DC)	Emma, gcov
	· SW Unit/Integration Test 이행 (기능 검증 Focus)	○	○		◎	○	✓ 요구사항커버리지 ✓ 요구사항 달성을 ✓ 기능 검증	SonarQube, TestLink
	· System Test 이행 (비기능 검증 Focus)	○	○	○	◎	○	✓ 비기능 검증 (성능, 볼륨, 스트레스, 보안 등)	-
	· Test Result 분석(각 테스트 이행 후)	○			◎		✓ 요구사항 달성을 ✓ 테스트 이행률 ✓ 테스트 성공률 ✓ 결합밀도	Jenkins
	· 인수/릴리즈 충족성 평가 ※ 매 릴리즈 단계 or 개발완료 후	◎	○				✓ 요구사항 달성을 ✓ 결합밀도 ✓ 결합조치율	Jenkins

➤ 착수/계획단계 : High Level Requirements 분석·정의 and Review

상위수준의 요구사항 분석 및 정의는 비즈니스 분석가를 중심으로 수행한다.

관련 산업 및 기술트렌드 등을 고려한 로드맵을 기반으로 잠재적 시장 및 수요자 요구사항 등을 식별하고, 조직의 비즈니스 목표를 수립한다.

비즈니스 요구사항은 비즈니스 목표를 충족하기 위한 기능/비기능 요구사항을 모두 포함하여야 하며, 관련 국내외 표준 및 법제도 등에 의한 제약조건 등을 모두 고려하여야 한다.

이러한 요구사항은 품질관리 담당자에 의해 상위수준의 품질목표와 세부 지표로 연계되어 정의되어야 한다.

정의된 비즈니스 요구사항, 비즈니스 목표 및 이에 기반을 둔 세부 품질목표·지표 등은 관련 이해관계자 간의 리뷰를 통해 일관성, 완전성, 추적성 등을 검증한다.

➤ 요구분석단계 : Low Level Requirements 분석·정의 and Review

비즈니스분석가는 상위수준의 요구사항, 비즈니스 룰 등을 기반으로 세부적인 시스템·SW 요구사항을 도출한다.

도출된 요구사항은 구현가능한지, 테스트 가능한 수준으로 명확히 정의되었는지 분석한 후 중요도, 리스크수준 등을 고려하여 우선순위를 부여한다.

도출된 시스템·SW 요구사항이 비즈니스 목표 및 요구사항과 연계하여 잘 정의되었는지 중점적으로 검증한 후 최종적으로 정의된 요구사항은 Visualization Tool(예, Redmine)에 등록하여 관리하도록 한다.

품질관리담당자는 정의된 프로세스를 준수하여 요구사항 분석·정의활동이 이행되었는지, 요구분석단계에 작성될 산출물이 표준에 따라 잘 작성되었는지를 검토하고, 도출된 요구사항이 비즈니스 요구사항으로부터 양방향으로 추적성을 확보하는지 검증한다.

➤ 설계단계 : SW Design and Review

설계담당자는 요구분석단계에서 정의된 요구사항을 기반으로 아키텍처설계를 수행한다. 이때, 요구사항단계에서 정의된 비기능 요구사항을 고려해야 하며, 상충되는 요구 사항에 대해서는 가중치, 우선순위 등을 고려하여 대안을 제시하여 의사결정을 할 수 있도록 해야 한다. 상위수준의 아키텍처설계를 기반으로 세부 인터페이스, 클래스 등 의 세부설계를 수행한다.

테스터는 시스템설계를 기반으로 시스템·통합테스트를 위한 테스트 설계를 수행한다.

설계가 완료되면, 품질관리담당자 주도하에 관련 이해관계자이 설계단계 산출물에 대한 통합리뷰를 진행한다. 또한, 요구사항으로부터 양방향으로 추적성을 확보하는지 검증한다.

신뢰성이 요구되는 SW의 경우는 모델 체킹 등 정형검증을 추가적으로 수행하여 신뢰성을 확보할 수 있도록 하여야 한다.

➤ 구현단계 : SW Design and Review

SW개발담당자는 설계에 따라 코드를 구현한 후 구현된 코드의 신뢰성 검증을 위해 정적분석을 이행한다.

테스터는 적절한 도구(예, PMD 등)를 이용하여 정적분석을 수행할 수 있도록 지원하며, 분석결과는 Visualization Tool(예, Jenkins)과 연계하여 Report 할 수 있도록 한다.

품질관리담당자는 Visualization Tool(예, Jenkins)을 통해 정적분석결과(코딩표준준수율, 정적분석이행율, 매트릭만족도 등)가 목표한 수준을 충족하는지 확인한다.

SW개발담당자는 구현된 코드의 적정성 및 최적화를 위해 코드리뷰를 실시한다.

➤ 테스트단계 : SW Test and Result 분석

SW개발담당자는 구현이 완료된 코드를 대상으로 구조적 검증을 위해 White Box Test를 이행한다. 이때, 테스터는 적절한 도구(예, JUnit, Emma 등)를 이용하여 테스트를 수행할 수 있도록 지원하며, 분석결과는 Visualization Tool(예, Jenkins)과 연계하여 Report 할 수 있도록 한다.

SW개발담당자는 White Box Test 결과(구조적 커버리지)가 목표한 수준을 충족하는지 확인한다.

White Box Test가 완료되면, SW개발담당자 및 테스터는 SW 기능검증 중심의 단위 및 통합테스트를 이행한다.

시스템테스트는 테스터의 주도하에 비기능 요구사항(성능, 볼륨, 스트레스 등)을 충족하는지 검증하며, 테스트결과가 요구사항에 충족되는지 확인한다.

품질관리담당자는 테스트 진행에 따라 Visualization Tool(예, Jenkins)을 통해 테스트결과에 따른 테스트이행률, 테스트성공률, 결함밀도, 요구사항달성을 등이 목표에 충족되는지를 확인하고, 인수/릴리즈 요건이 충족되면 이해관계자(CEO, 고객 등)에게 보고한다.

V

소스코드의 시각화

1. SW Visualization 품질지표
 - 가. 개발 SW에 따른 지표 설정
2. SW Visualization 시스템
 - 가. SW Visualization 프로세스
 - 나. SW Visualization 시스템 구성요소
 - 다. 시스템 구성 절차
3. SW Visualization 구성원 역할
 - 가. Role & Responsibility 정의
 - 나. SW 개발단계별 주요 Activity

V

소스코드의 시각화

소프트웨어 개발 프로젝트에서는 요구사항의 잦은 변경, 무리한 계획 수립과 같은 관리적인 차원의 문제들도 많지만, 마찬가지로 기술적인 차원에서 해결해야하는 문제들도 많이 존재하며, 이는 대부분 소스 코드의 복잡성의 관리에서 기인한다. 소프트웨어의 개발 과정이 진행되면 진행될수록, 특정 코드를 변경했을 경우 무슨 일이 일어나며 어떤 영향을 미치는지 아무도 모르게 될 정도로 복잡성이 증가하며, 이를 방지하기 위해서 소스 코드의 내부 구조 품질 관리가 필요하게 된다.

1. 역공학

가. 역공학 기법의 개요

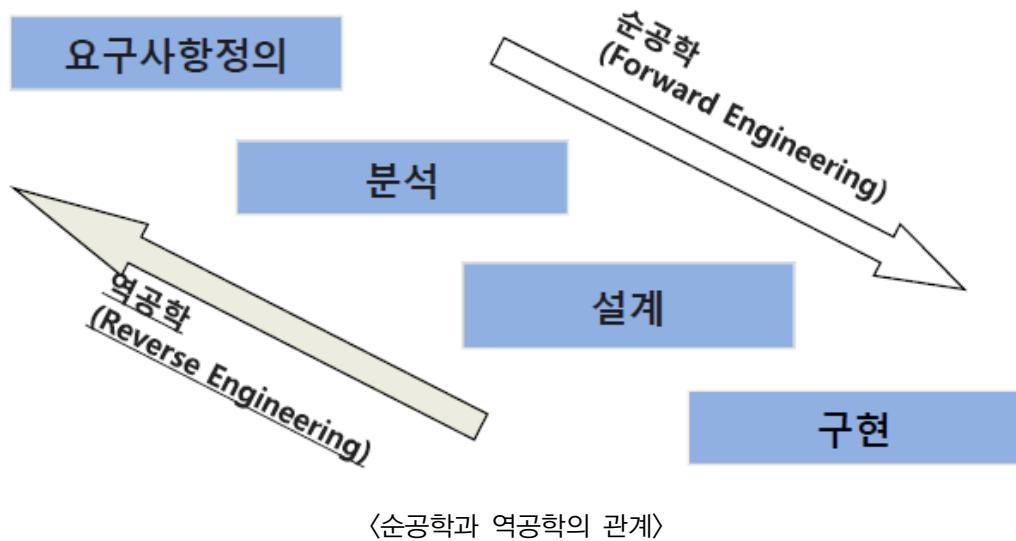
순공학 (Forward Engineering)은 요구사항 명세와 같은 높은 수준의 추상화 산출물에서 시작하여 점진적으로 분석, 설계 등 상세화를 통해서 소프트웨어를 구현하는 전통적인 절차라고 할 수 있다.

SW소프트웨어의 성공적인 개발을 위해서는 우수한 설계 과정이 동반되어야 하며, 이는 소프트웨어 아키텍처 전문가가 직접 담당하거나 관련 담당자가 아키텍처 설계 기법을 따라 착실하게 진행하여야 한다. 이와같은 아키텍처 설계를 위해서는 소프트웨어의 내부 구성 요소들간의 특성과 관계를 이해하고 상호작용을 분석하기 위한 아키텍처 전문 지식이 필요하며, 이와 관련된 내용은 SW공학센터의 아키텍처 설계지침(http://www.software.kr/mbs/swkr/jsp/board/view.jsp?spage=2&boardId=143&boardSeq=12450&mcategoryId=&id=swkr_040100000000)에서 제공하고 있다.

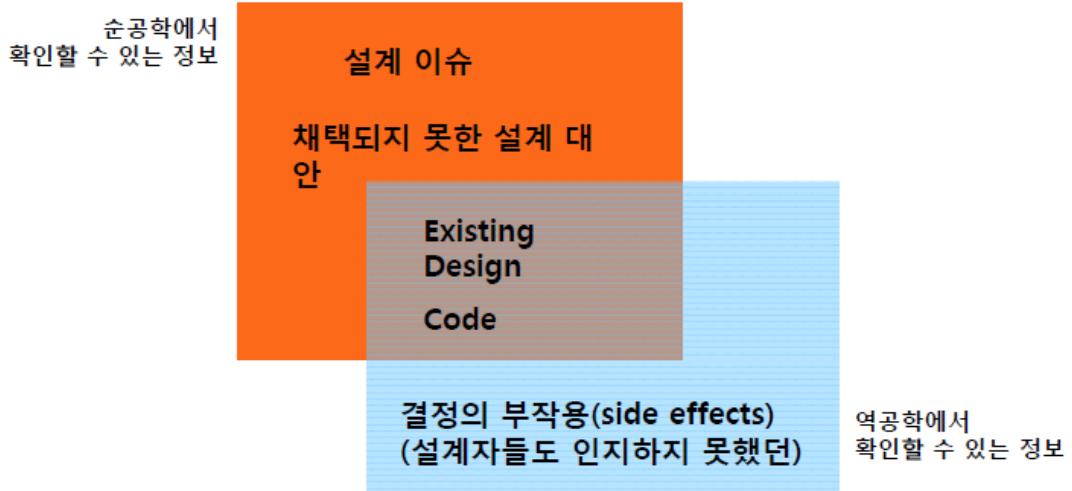
역공학 (Reverse Engineering) 기법은 소프트웨어의 컴포넌트와 이들 간 상호관계 식별 및 또 다른 형태 또는 더 높은 추상 수준의 표현물을 복원하기 위한 시스템 분석 기법으로 활용된다. 역공학 기법으로 시각화를 수행하는 이유는 첫 번째는 소프트웨

어를 개발한 프로그래머의 도움이 없이 소프트웨어에 대한 이해를 얻기 위해서이고, 두 번째는 대상 시스템을 깊이 분석하지 않고도 기존 시스템을 점검할 수 있는 방법이기 때문이다.

따라서 역공학 (Reverse Engineering)은 소프트웨어의 컴포넌트와 이들 간의 상호관계 식별 및 또 다른 형태 또는 더 높은 추상 수준의 표현물을 복원하기 위한 시스템 분석 기법이라 할 수 있다.



그림처럼 순공학은 복잡한 기능들을 구현하기 위하여 점진적으로 상세화한다. 그러나 역공학은 반대의 경우로 주로 코드 상의 정보들을 추출하여 해석함으로써 보다 큰 그림을 복원하는 과정이다. 종종, 원저자의 기억과는 다른 그림이 얻어지지만 더 정확한 경우도 많으며, 소프트웨어에 대한 다양한 통찰력을 제공할 수 있다.



이처럼 역공학은 순공학에서 얻기 어려운 새로운 관찰을 제공하지만, 역공학을 통해서 확보할 수 없는 정보들도 존재한다. 위의 그림은 그러한 부분을 잘 나타내고 있다.

2. 소프트웨어의 내부 설계

가. 개요

소프트웨어의 내부 구조를 설계하는 기법으로 추상화/모듈화/캡슐화가 있다.

- 추상화 (abstraction) : 사용자와 구현자의 분리를 통하여 그 세부적인 내용을 모르는 여러 사용자가 이용할 수 있도록 한다
- 모듈화 (modularity) : 소프트웨어를 각 기능별로 분할하여 각각의 모듈에 대한 복잡도를 낮추고, 독립성을 증가시켜 변경 및 구현을 용이하게 한다
- 캡슐화 (encapsulation) : 내부는 숨기고, 외부에 관계된 요소만 개방함을 통해서 블랙박스화된 모듈의 성격을 구현한다. 프로세스와 정보를 내부로 감추어 다른 모듈에 영향을 주지 않게 함으로써 완전히 독립시킨다

나. 품질 지표

소스 코드의 내부 구조의 품질을 측정하는 지표로는 결합도와 응집도가 있다

- 결합도 (cohesion) : 결합도는 모듈 간의 상호의존하는 정도 또는 두 모듈 사이의 연관관계를 의미한다. 독립적인 모듈이 되기 위해서는 각 모듈간의 결합도가 약해야 하며 의존하는 모듈이 적어야 한다.
- 1) 자료 결합도 : 모듈간의 인터페이스가 자료 요소만으로 구성된 결합도이다. 한 모듈이 다른 모듈을 호출하면서 매개 변수나 인수로 데이터를 넘겨주고, 호출 받은 모듈은 받은 데이터에 대한 처리 결과를 돌려주는 방식으로 모듈 간의 내용을 전혀 알 필요가 없고 다른 모듈에 전혀 영향을 끼지치 않는 가장 바람직한 결합도이다.
- 2) 스템프 결합도 : 모듈간의 인터페이스로 배열이나 레코드 등의 자료 구조가 전달되는 결합도이다. 두 모듈이 동일한 자료 구조를 조회하는 방식이며 자료 구조의 변화는 실제로 조회하지 않는 모듈에까지도 영향을 미치게 된다.
- 3) 제어 결합도 : 논리적인 흐름을 제어하는 데 사용하는 제어 요소 (Function Code, Switch , Tag, Flag)가 전달되는 결합도이다. 상위 모듈이 하위 모듈의 상세한 처리 절차를 알고 통제하는 경우나 처리 기능이 두 모듈에 분리되어 설계된 경우이다.
- 4) 외부 결합도 : 한 모듈에서 외부로 선언한 데이터(변수)를 다른 모듈에서 참조하는 결합도이다.
- 5) 공유 결합도 : 공유되는 공통 데이터 영역을 여러 모듈이 사용할 때의 결합도이다. 공통 데이터 영역의 내용을 조금만 변경하더라도 이를 사용하는 모든 모듈에 영향을 미치게 되므로 독립성이 약하다
- 6) 내용 결합도 : 한 모듈의 다른 모듈의 내부 기능 및 그 내부 자료를 직접 참조하거나 수정할 때의 결합도이다.



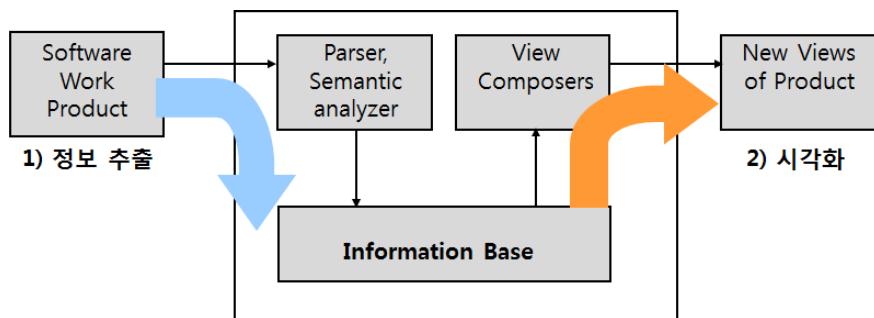
- 응집도 (coupling) : 응집도는 정보 은닉 개념을 확장한 것으로 모듈 안의 요소들이 서로 관련되어 있는 정도, 즉 모듈이 독립적인 기능으로 정의되어 있는 정도를 의미한다.
 - 1) 기능적 응집도 : 모듈 내의 모든 요소들이 단일 기능을 수행한다
 - 2) 순차적 응집도 : 모듈 내의 한 요소의 출력 자료가 다음 요소의 입력 자료로 사용된다
 - 3) 통신적 응집도 : 모듈 내의 요소들이 동일한 입출력 자료를 이용하여 서로 다른 기능을 수행한다
 - 4) 절차적 응집도 : 모듈의 수행 요소들이 반드시 특정 순서대로 수행된다
 - 5) 일시적 응집도 : 모듈의 기능 요소들이 같은 시간에 모두 실행된다
 - 6) 논리적 응집도 : 논리적으로 유사한 기능을 수행하지만 서로 관계하지 않는다
 - 7) 우연적 응집도 : 모듈 내의 요소들이 뚜렷한 관계없이 존재한다.



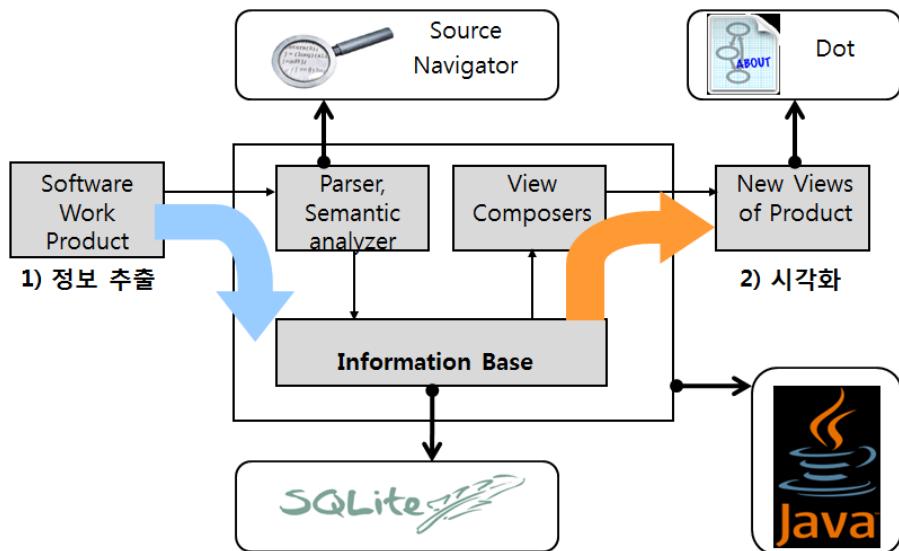
모듈의 독립성은 결합도(Coupling)와 응집도(Cohesion)에 의해 측정되며, 독립성을 높이려면 모듈의 결합도를 약하게 하고 응집도를 강하게 하며 모듈의 크기를 작게 만들어야 한다.

3. 소스 코드의 시각화

도구를 사용한 소프트웨어 아키텍처의 복원 절차는 아래 그림과 같다. 소스코드로부터 정보를 추출하여 의미 있는 단위로 구분하고 이를 데이터베이스화하여 저장한다. 저장된 데이터를 기반으로 시각화 도구를 사용하여 다이어그램 형태로 SW구조를 보여준다.

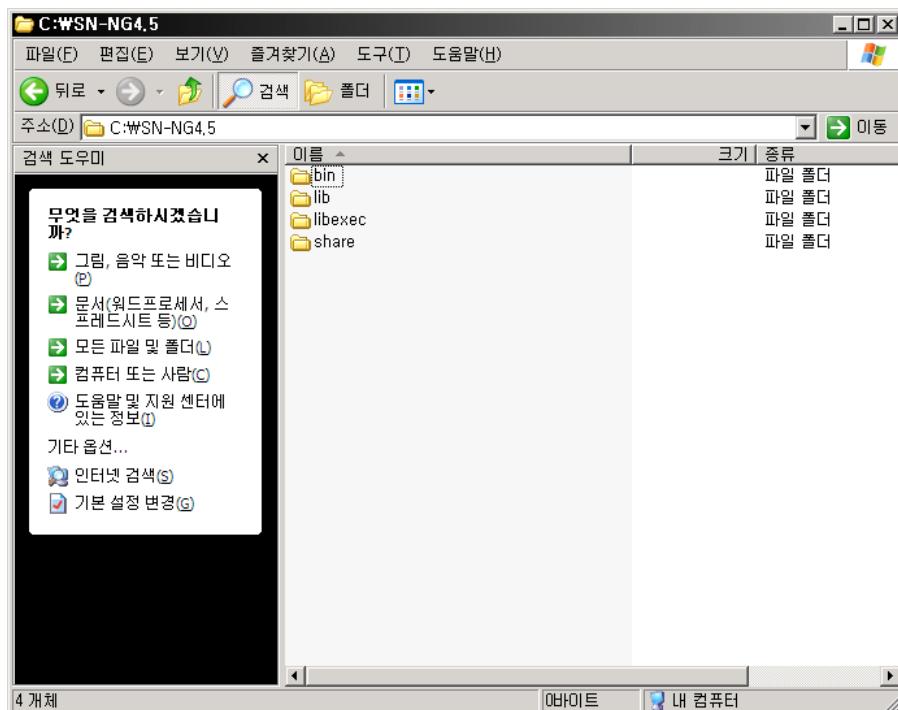


위와 같은 기본적인 흐름을 수행할 수 있는 시스템을 공개 도구기반으로 구성한다. 이클립스를 기반으로 하여 소스분석도구, 데이터베이스, 그래픽도구 등을 아래와 같이 구성한다.



가. Source Navigator 설치

Source Navigator는 4.5 버전을 기준으로 하며, 별도의 설치 없이 압축 파일을 압축 해제하면 된다. Source Navigator의 실행 경로에는 한글 폴더 이름이 있으면 오동작을 할 수 있으며, 압축 해제 후 libexec 폴더의 파일들을 bin 폴더로 옮긴 후 snavigator.exe를 실행하면 동작한다.



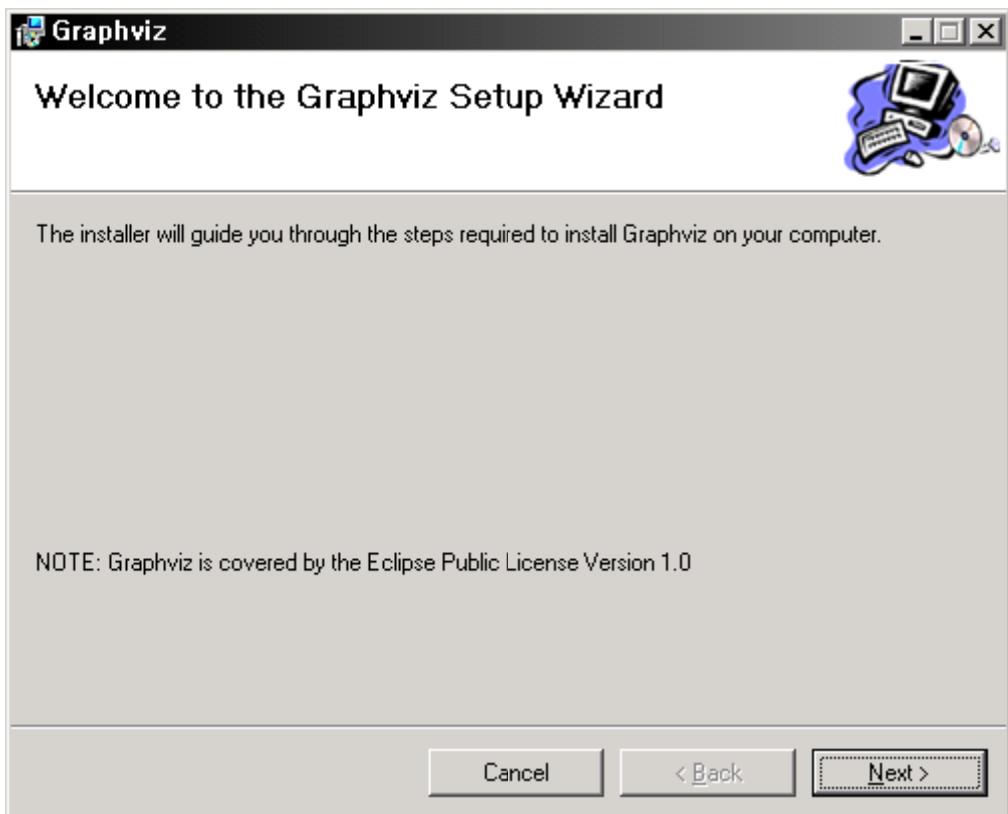
〈Source Navigator 폴더 구조〉

* URL : <http://sourcenav.sourceforge.net/index.html>

* Version : SN-NG4.5

4. Graphviz 설치

Graphviz 는 2.30 버전을 기준으로 설치한다

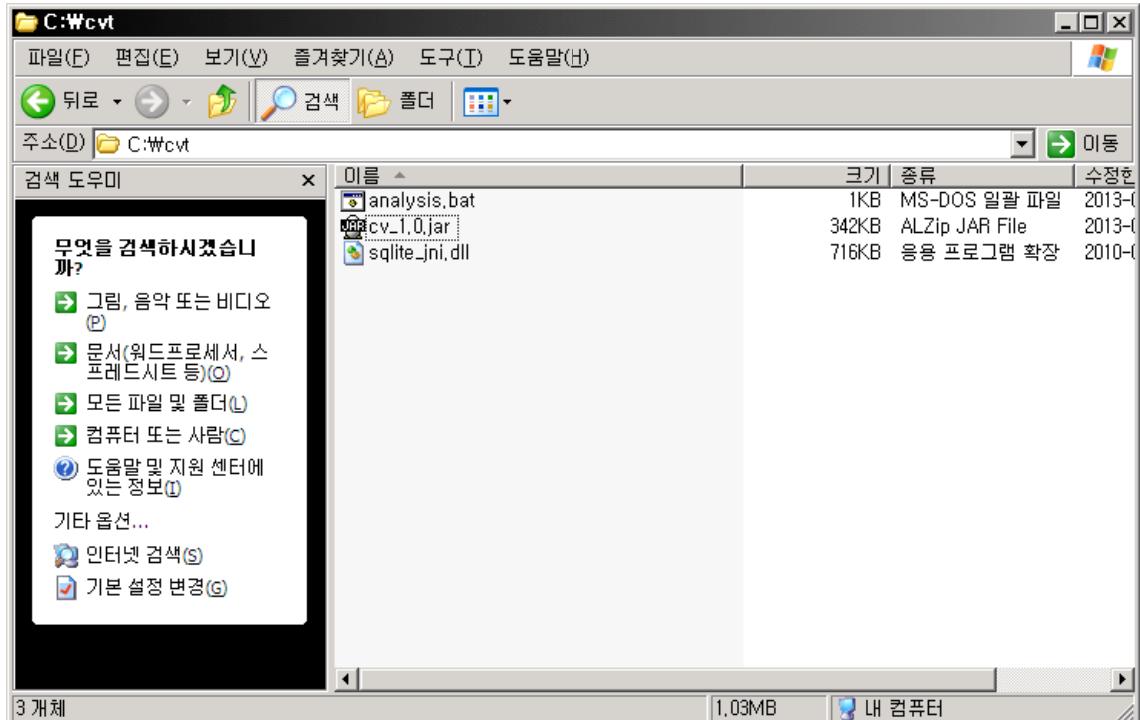


* URL : http://www.graphviz.org/Download_windows.php

* Version : graphviz 2.30.1

다. Code Visualization Tool 동작

Code Visualization Tool 은 위의 두 도구를 이용하여 소스 코드 전체에 대한 call graph 를 생성하는 도구이며, SW공학센터에 신청을 통하여 제공받을 수 있다.



〈Code Visualization Tool〉

C, C++, Java 가 지원되며 해당 소스 코드가 설치된 경로를 analysis.bat 〈경로명〉로 입력하여 실행시키면 해당 소스 코드에 대한 Call Graph 를 얻을 수 있다.

```
C:\jos77\javaworkspace>analysis.bat c:\jos77\javaworkspace\javaproject

C:\jos77\javaworkspace>cd c:\jos77\javaworkspace

C:\jos77\javaworkspace>dir /b/s 1>files.txt

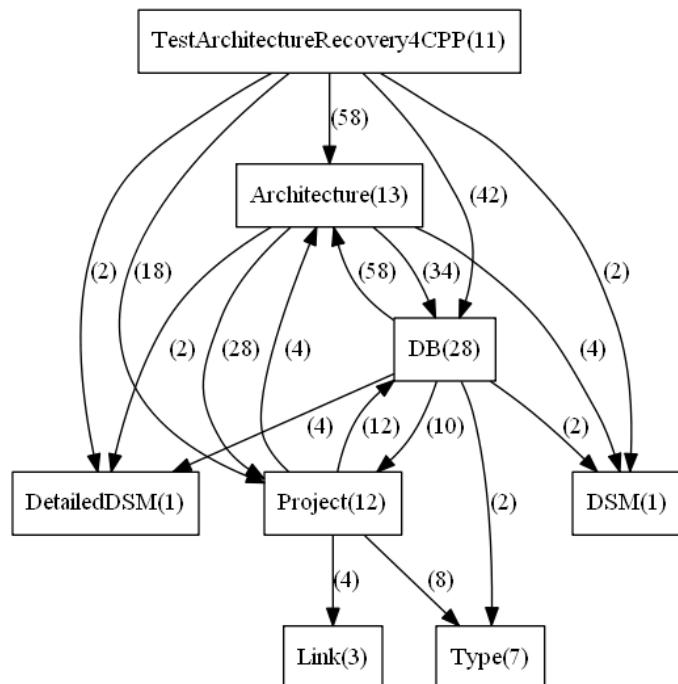
C:\jos77\javaworkspace>cd c:\WSN-6.0\bin

C:\WSN-6.0\bin>snavigator --batchmode --import files.txt --databasedir .\WSNDB --projectname javaproject

C:\WSN-6.0\bin>cd c:\jos77\javaworkspace

C:\jos77\javaworkspace>java -jar cv_1.0.jar c:\jos77\javaworkspace\javaproject
done
```

〈Code Visualization Tool 실행 화면〉



〈예제 : Call Graph 산출물〉

VI

소스코드의 문서화

1. JavaDoc

- 가. JavaDoc 주석 작성방법
- 나. JavaDoc 명령어를 통한 생성방법
- 다. 이클립스에서의 JavaDoc 생성

2. Doxygen

- 가. Doxygen 주석 작성방법
- 나. Doxygen 문서 생성방법
- 다. Doxygen 출력 문서 확인

VI 소스코드의 문서화

SW의 대형화, 복잡화에 따라 대부분의 SW개발은 다수의 개발자에 의한 공동개발이 진행되고 있다. 또한 국내 중소기업은 대부분 인력 및 비용의 부족에 의한 어려움을 겪고 있는 현실이다. 이러한 현실에서 SW 개발 중 개발 인력의 이동은 SW 개발에 심각한 문제를 야기할 수 있으며, 신규 개발자의 경우 타인의 소스코드를 분석해야 하는 어려움이 가중된다.

JavaDoc과 Doxygen은 소스코드의 문서화를 위한 지원도구로써 이러한 문제점의 완화를 위하여 활용될 수 있다. JavaDoc과 Doxygen은 문서화 작업을 위하여 개발자가 별도의 문서를 생성할 필요가 없으며, 소스코드에 정해진 태그를 활용하여 주석 처리를 하는 것만으로 문서의 생성이 가능하도록 지원한다. 또한 Eclipse에서는 이러한 문서화를 위한 많은 편의 기능을 제공하고 있어, 개발자의 문서화 부담을 완화하고 있다.

아래 표는 JavaDoc과 Doxygen을 비교한 것으로 Java 개발에서는 JavaDoc이 주로 사용되며 이외 개발 언어에서는 Doxygen이 주로 활용되고 있다.

	JavaDoc	Doxygen
개요	Document를 코드에 포함하여 배포하는 형태로 별도의 Document 관리 부담을 줄임	
지원언어	Java	C, C++, C#, Fortran, Java, Objective-C, PHP, Python, IDL, VHDL
문서포맷	HTML	HTML, RTF, PDF, LaTex, PostScript, man
태그	자체 태그	JavaDoc, QT toolkit
장점	다양한 도구 연계 JDK에 기본 탑재	다양한 지원언어 및 문서 포맷 Diagram 포함 가능
단점	제한적 지원언어 및 문서 포맷	doxyfile 별도 필요

1. JavaDoc

JavaDoc은 Java 소스 코드로부터 HTML 형식으로 된 API(Application Programming Interface) 문서를 만들어 주는 문서 생성 기능을 제공한다. 출력된 HTML 형식의 문서는 문서의 각 내용들이 hyperlink로 연결되어 사용자의 편의성을 증대시킨다.

JavaDoc은 API 문서를 쉽게 생성할 수 있도록 도와준다. 프로그래밍에 대한 API 문서를 따로 만드는 것이 아니라, Java 소스 코드 안에 JavaDoc에서 요구하는 형식의 주석문을 작성하고, 간단한 명령어로 JavaDoc 생성을 실행하여 API문서를 만들 수 있다. 즉, Java 프로그래밍을 하면서 수없이 봐 왔던 API들을 문서를 통하여 보다 쉽고 효율적으로 관리할 수 있는 것이다. 뿐만 아니라, Netbeans나 Eclipse와 같은 IDE에서는 JavaDoc 문서를 자동적으로 생성할 수 있도록 편리한 기능들을 제공하고 있다.

가. JavaDoc 주석 작성방법

- 일반적인 주석의 형식은 /* 로 시작해서 */로 끝나지만, JavaDoc 주석은 /** 로 시작해서 */로 끝나야 한다.
- 주석 작성에 필요한 대표적인 Tag는 다음과 같다.

```
@author : 소스코드 작성자  
@param : method의 패러미터에 대한 간략한 설명  
@return : 리턴값에 대한 설명  
@exception : method가 발생시킬 수 있는 예외를 설명  
@throws : @exception 과 동일  
@deprecated : 사용 추천 안함 (폐기된 method)  
@see : 참조되거나 관련있는 클래스 표시  
@version : 버전 정보 표시  
@since : 소스코드가 적용된 버전 정보  
(Tag를 가진 객체가 언제 추가되었는지 명시)
```

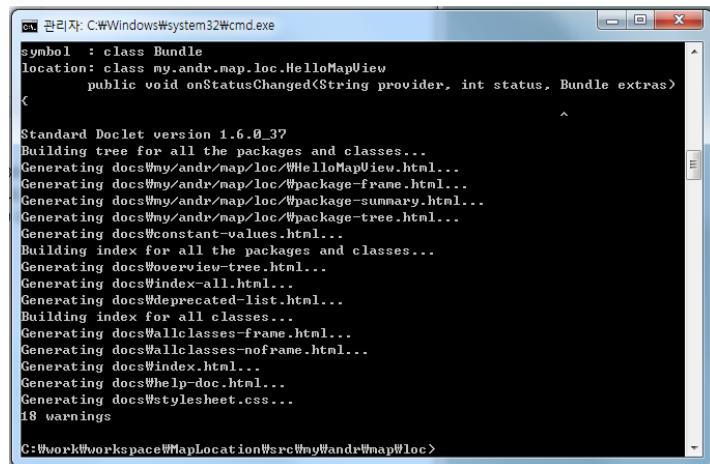
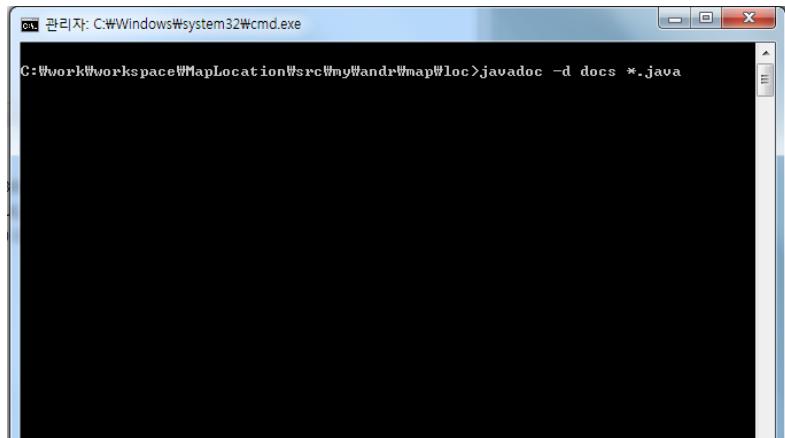
- 주석 작성 시, 다른 클래스를 참조할 경우에는 {@link }를 사용한다. 이것은 JavaDoc에서 하이퍼링크를 생성한다.
- Tag 이름은 대소문자를 구분한다.
- @throws는 사용자가 처리해야 할 예외이다. 따라서 이 예외가 언제 발생하는지에 대해 반드시 설명이 필요하다.
- @return에서 반환타입에 대해서 기술할 필요가 없다.
- @param, @return, @throws는 method에서 필수적인 속성이다.
- 클래스와 인터페이스의 주석에는 @author, @version, @since, @deprecated, @see 등의 Tag가 작성 가능하다.
- method의 주석에는 @throws, @exception, @param, @return 등의 Tag가 작성 가능하다.
- method의 설명은 HTML 형식으로 출력되므로, 줄바꿈을 하려면
태그를 사용해야 한다.
- interface가 존재한다면, 구현체에서는 주석을 생성할 필요가 없다. interface에서 정의된 method의 의도대로 구현을 한 것이므로, interface에서의 주석만으로 충분하다. 단, interface에 정의되어 있지 않은 구현체의 method는 주석을 필요로 한다.
- JavaDoc 문서를 만들 때 속성이 public 또는 protected로 된 class, method, variable만 생성되고, 속성이 private인 것은 제외된다.
- Eclipse에서는 /** 을 입력 후 엔터키를 누르면 자동으로 @param, @return, @throws 등 Tag가 생성된다.

나. JavaDoc 명령어를 통한 생성방법

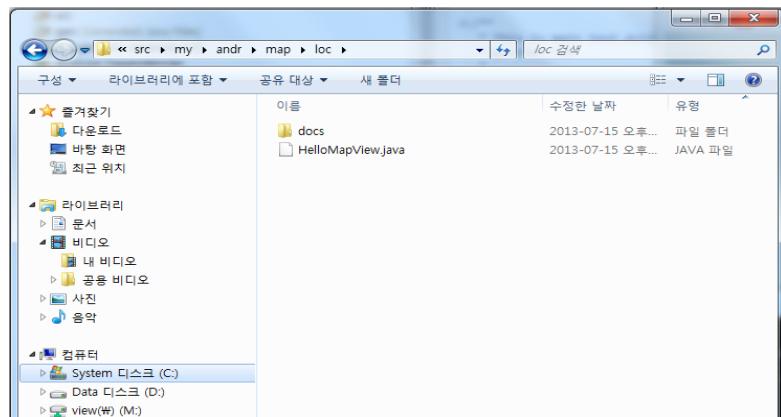
기본적으로, 커맨드 창을 통하여 javadoc 명령어를 실행시킴으로써 JavaDoc 문서를 생성할 수 있다.

커맨드 창을 실행시킨 후 java 파일이 있는 폴더로 이동한다. 그리고, 아래와 같이 명령어를 입력한다. 해당 폴더내의 모든 java 파일에 대하여 JavaDoc 문서를 생성하고 결과물을 docs 폴더에 저장하라는 명령어이다.

```
javadoc -d docs *.java
```



명령어를 실행하면 문서 생성이 진행되고, 그 결과 docs 폴더가 현재 위치에 생성되고 그 속에 패키지별로 JavaDoc 결과물이 만들어진다. 이것은 사용법의 한 예를 든 것이며, 커맨드 창에서 javadoc 을 실행하면 기본적인 사용법 및 여러 가지 패러미터 옵션들을 확인할 수 있다.



다. 이클립스에서의 JavaDoc 생성

JavaDoc은 통합개발환경인 Eclipse 안에서 편리하게 사용할 수 있다.

우선, 다음과 같이 JavaDoc 형식에 맞는 주석문 작성이 필요하다.

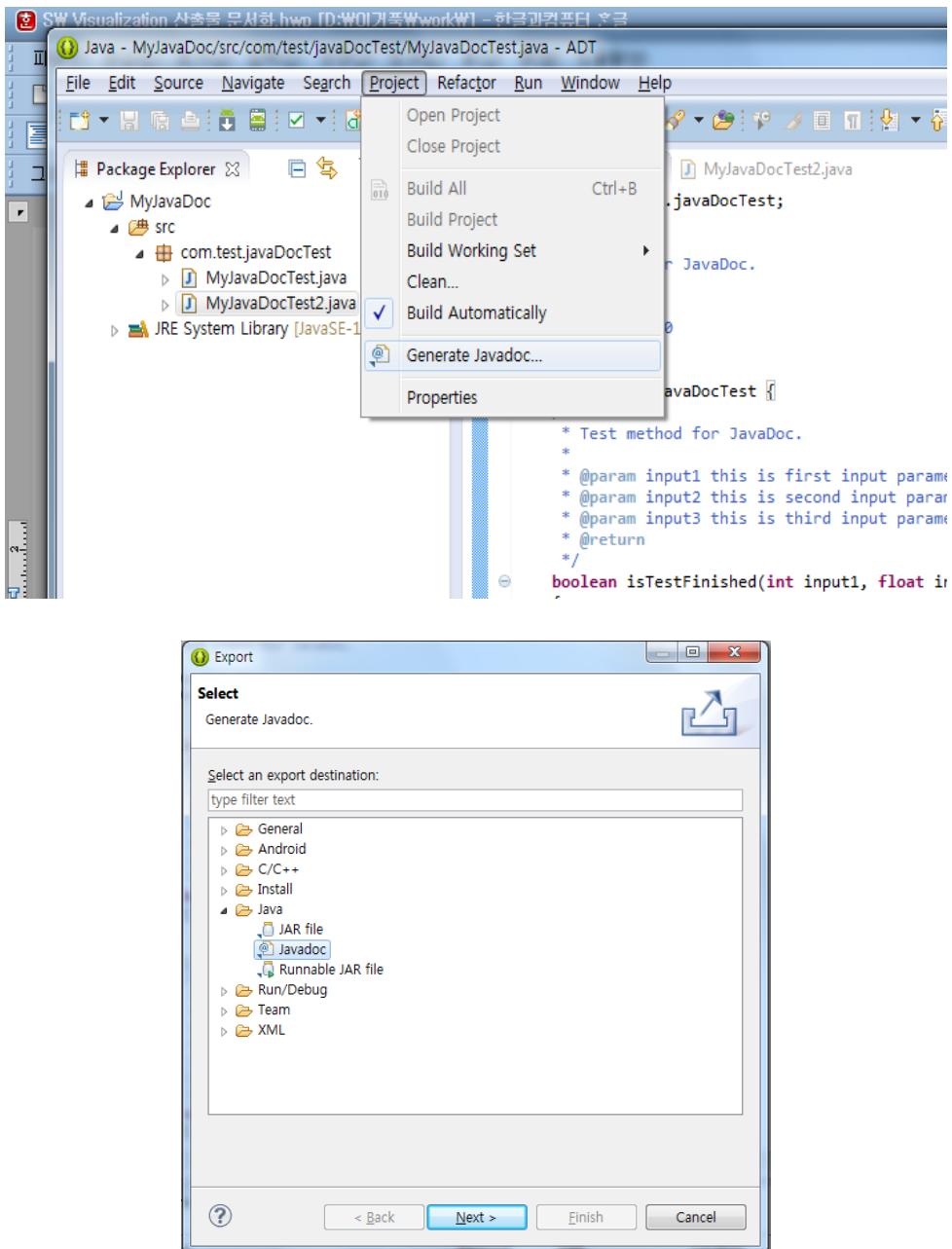
```

MyJavaDocTest.java  MyJavaDocTest2.java
package com.test.javaDocTest;

/**
 * Test class for JavaDoc.
 * @author test
 * @version 1.0.0
 */
public class MyJavaDocTest {
    /**
     * Test method for JavaDoc.
     *
     * @param input1 this is first input parameter
     * @param input2 this is second input parameter
     * @param input3 this is third input parameter
     * @return
     */
    boolean isTestFinished(int input1, float input2, String input3)
    {
        return false;
    }
    /**
     * Test method 2 for JavaDoc.
     *
     * {@link #isTestFinished(int, float, String)} is main method.
     *
     * @param input1 this is first input parameter
     * @param input2 this is second input parameter
     */
    void makeFirstDoc(int input1, boolean input2)
    {
    }
}

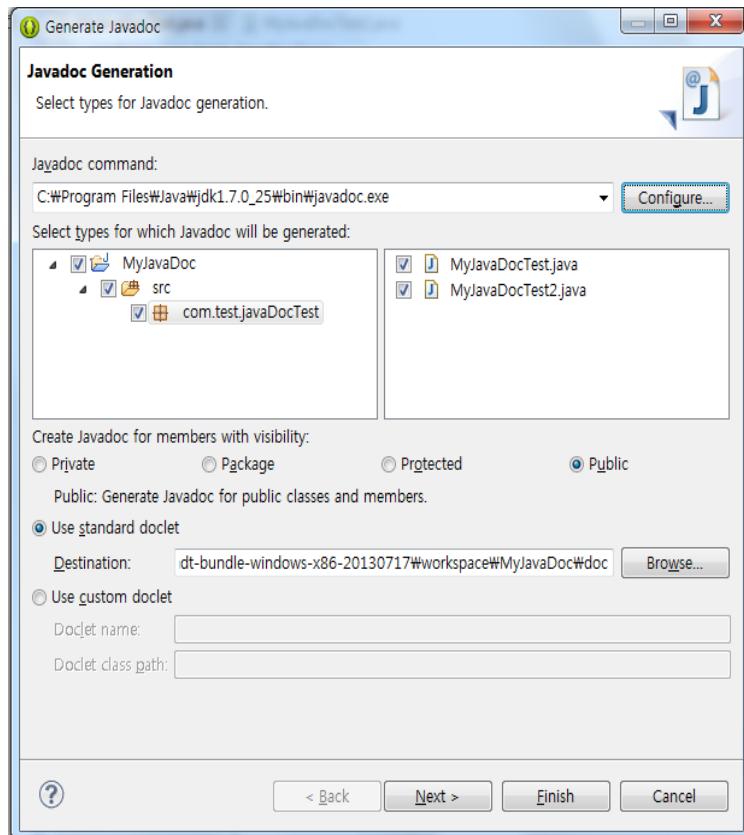
```

주석문 작성이 완료되었다면, 이제 Eclipse 메뉴를 통하여 JavaDoc 문서 생성을 위한 설정으로 넘어간다.



위와 같이 Eclipse 상단 Project 탭 메뉴의 Generate Javadoc... 을 통해서 진입하거나 해당 프로젝트의 Export 메뉴에서 Java 폴더 하위의 Javadoc 기능을 통하여 진입한다.

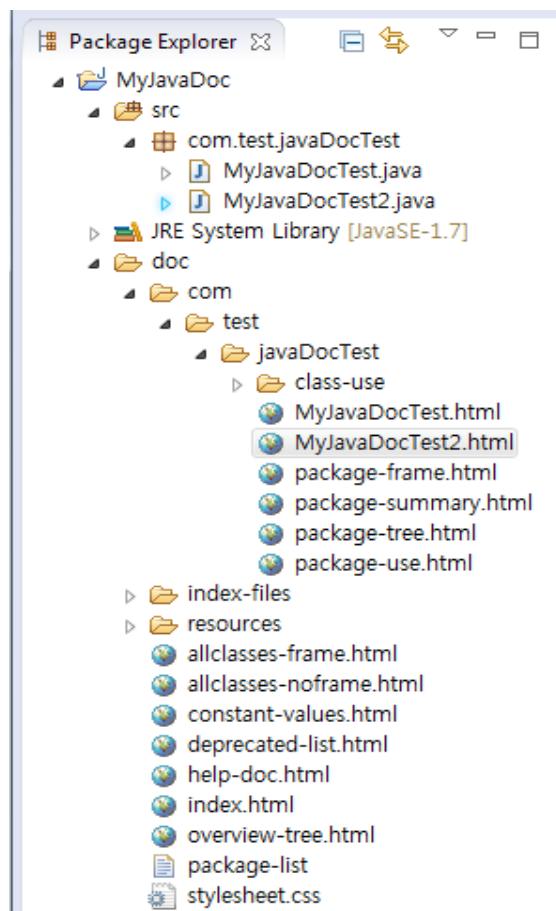
Javadoc 기능을 실행하면 다음 그림과 같이 JavaDoc 문서를 생성하기 위한 창이 뜬다. 여기서 설정해야 하는 옵션들은 다음과 같다.



- Javadoc command
 - : JDK 가 설치된 폴더에서 javadoc.exe를 찾아 선택한다.
- Select types for which Javadoc will be generated
 - : JavaDoc 문서를 생성할 패키지 또는 파일들을 선택한다.

- Create Javadoc for members with visibility
 - : JavaDoc 문서를 생성할 범위를 선택한다.
- Destination
 - : JavaDoc 문서가 만들어질 위치를 선택한다.

JavaDoc 생성이 완료되면, 다음과 같이 프로젝트 폴더에서 doc 폴더가 생성되고 HTML 문서가 들어가 있음을 확인할 수 있다.



생성된 html 문서는 아래와 같이 보여진다.

The screenshot shows a Java class documentation page. At the top, there's a navigation bar with links: Package, Class (which is highlighted in orange), Use, Tree, Deprecated, Index, and Help. Below the navigation bar, there are links for Prev Class, Next Class, Frames, No Frames, All Classes, Summary, Nested, Field, Constr, Method, and Detail, Field, Constr, Method.

The main content area starts with the package name: com.test.javaDocTest. Then it shows the class definition: **Class MyJavaDocTest**. It indicates that it extends java.lang.Object. Below this, there's a code snippet:

```
public class MyJavaDocTest
extends java.lang.Object
```

Following the code snippet, there are metadata sections: Test class for JavaDoc, Version: 1.0.0, and Author: test.

At the bottom of the main content area, there's a link labeled "Constructor Summary".

This section is titled "Constructor Summary". It has a navigation bar with links: Constructors (which is highlighted in orange), Constructor and Description, and MyJavaDocTest(). The "MyJavaDocTest()" link is underlined, indicating it is currently selected.

This section is titled "Method Summary". It has a navigation bar with links: Methods (which is highlighted in orange), Modifier and Type, and Method and Description. Below this, there are two method entries:

Modifier and Type	Method and Description
boolean	isTestFinished(int input1, float input2, java.lang.String input3) Test method for JavaDoc.
protected void	makeFirstDoc(int input1, boolean input2) Test method 2 for JavaDoc.

Below the methods, there's a section titled "Methods inherited from class java.lang.Object" containing a list of methods: clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait.

Method Detail

isTestFinished

```
public boolean isTestFinished(int input1,  
                             float input2,  
                             java.lang.String input3)
```

Test method for JavaDoc.

Parameters:

- input 1 - this is first input parameter
- input 2 - this is second input parameter
- input 3 - this is third input parameter

Returns:

makeFirstDoc

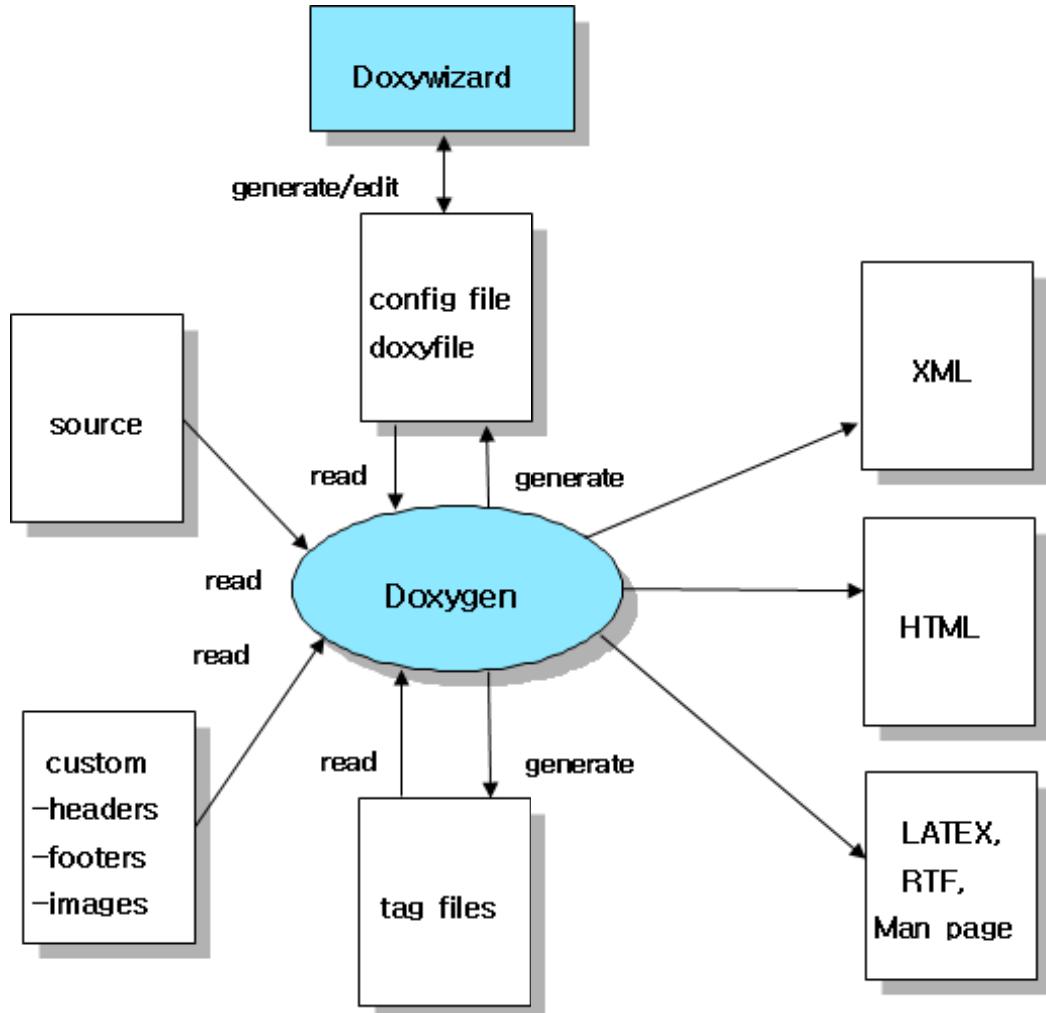
```
protected void makeFirstDoc(int input1,  
                           boolean input2)
```

Test method 2 for JavaDoc. `isTestFinished(int, float, String)` is main method.

Parameters:

- input 1 - this is first input parameter
- input 2 - this is second input parameter

2. Doxygen



위의 그림은 doxygen tool 과 그와 연관된 정보들의 흐름을 보여준다. 실행 가능한 doxygen 은 소스 프로그램을 파싱하여 결과 문서를 생산해 내는 main 프로그램이며, doxywizard 는 doxygen 실행에 필요한 configuration 파일을 생성하고 doxygen 을 running 하는 그래픽 환경을 제공한다.

가. Doxygen 주석 작성방법

doxygen 문서를 만들기 위해서는 doxygen이 지원하는 형태의 주석을 작성해야 한다. 주요 작성법은 다음과 같다.

- doxygen 은 여러 언어의 주석 블록을 지원한다. Python, VHDL, Fortran, Tcl 등 을 위한 주석 블록도 각각 지원하지만 C, C++, C#, Objective-C, PHP, Java 등에서 공통적으로 사용되는 주석 블록이 잘 알려져 있다. 이것은 기존 언어의 주석 법에 문자를 하나 더 추가하여 사용하는 방법이며, 반드시 설명하고자 하는 코드(함수선언/클래스 선언/매크로, 상수 선언)의 상단에 위치해야 하며 doxygen 최신 버전은 source파일에 대한 주석도 지원한다.

명칭	내용
JavaDoc 스타일	전형적인 /* */ 의 주석에 JavaDoc 스타일처럼 *를 두 개 붙여서 사용
C++ 스타일	전형적인 주석 방법인 // 문자에 / 문자나 ! 문자를 하나 더 추가하여 사용
Qt 스타일	* 문자 뒤에 ! 문자를 추가하여 사용

```
/*
 *
 * this is JavaDoc - style.
 *
 */
void firstCommentExample()
{
}

///
/// this is C++ - style.
///
///
///!
///! this is C++ - style.
///!
void secondCommentExample()
{
}

/*
 * this is Qt - style.
 */
void thirdCommentExample()
{}
```

- 주석 설명은 해당 함수, 클래스, 매크로, 상수값 등이 무엇인지 설명하는 것으로, 설명은 반드시 마침표로 끝나야 하며 주석블럭에서 처음 항목에 기술되어야 한다.
- 효과적이고 다양한 문서작성을 위하여 doxygen에서 제공하는 명령어는 '@' 문자로 시작한다. 여러 가지 명령어들이 다양한 기능들을 제공하나, 일부 주요 명령어들은 다음과 같다.

```

@author : 작성자를 표시한다.
@brief : 간략한 설명을 기재한다.
@data : 작성한 날짜를 표시한다.
@see : 참고하는 함수/클래스 등의 표시로 해당 항목(doxygen에서 존재하는)의 링크를 제공한다.
@return : 함수의 리턴값을 설명
@exception : 예외 처리에 대한 설명
@throw : throw 하는 객체나 변수 등에 대한 설명
@version : 버전 설명
@param : parameter에 대한 설명
@warning : 경고
@bug : 버그에 대한 기술, 전체 항목의 @bug에 대해서 따로 리스트가 생성된다.
@todo : 해야 할 일들에 대한 기술, 전체 항목의 @todo에 대한 리스트가 생성된다.
@code, @endcode : 예제 코드 등의 코드 삽입을 위한 명령어로, 시작과 끝을 표시해 준다. 이 안에 있는 내용들은 코드 형태로 문서화된다.
@remark : 자세한 설명 및 주의사항에 관하여 기재한다.

```

- 일반적으로 함수, 클래스, 구조체, 공용체, 매크로, 상수값에는 위와 같은 방법으로 기술하면 되지만 변수에 대해서는 다른 방법으로도 기술 할 수 있다. 기존에 사용하는 주석 블록 표시에 '\'' 문자를 붙여서 사용하면 변수에 대한 문서화 작업을 하게 된다.
- 함수, 클래스, 구조체, 변수 등의 항목에는 반드시 해당 코드 상위에 코드를 작성하지 않아도 주석 블록 처음에 해당 항목을 정확히 지시(indicate)해 주면 주석을 한곳에 모아서 작성할 수 있다. 하지만 이 때 반드시 정확한 정보를 작성해야 하

는데 이것은 코드부와 주석이 다른 파일에 존재할 수도 있기 때문이다. 예를 들어 클래스의 경우 @class 이후에 선언되어 있는 헤더 파일, 문서에 기록될 헤더 파일 경로를 수동으로 적어 주어야 한다.

- 소스가 문서화 되면 각각의 항목들은 링크를 가지게 된다. 이 링크를 문서 설명에 강제로 링크를 걸 수가 있는데 이 때 사용되는 명령이 @ref 이다. @ref 이후에 존재하는 링크의 이름을 적어주게 되면 그 항목이 가리키는 함수, 클래스 등에 대해서 링크가 걸어지게 된다.
- bold, italic, argument 문자 표현하기 등이 각각의 명령인 @b, @e, @a에 따라 단어가 꾸며진다.
- 주석 블록에 @mainpage 를 사용하면 그 페이지가 헬프문서의 메인페이지가 된다.

나. Doxygen 문서 생성방법

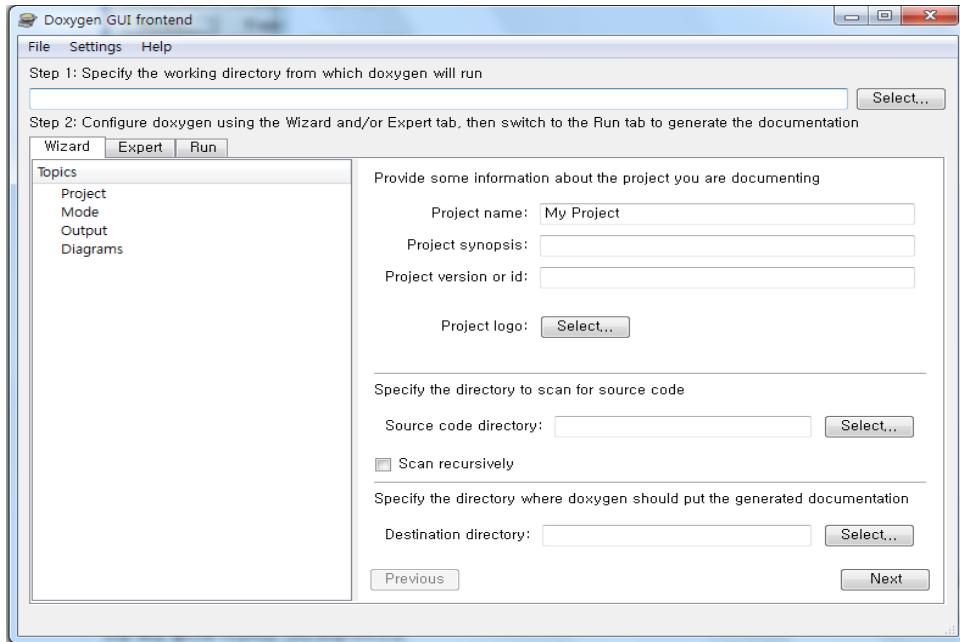
Doxygen 사용을 위해서는 먼저 다운로드 사이트에서 설치 프로그램을 다운로드 받아야 한다.

<http://www.stack.nl/~dimitri/doxygen/download.html>

위의 사이트에 접속하면 설치 프로그램을 다운로드 받을 수 있다. 설치 프로그램을 실행하고 안내에 따라 계속 진행하면 Doxywizard, Documentation, Example 등으로 이루어진 패키지가 모두 설치됨을 확인할 수 있다.

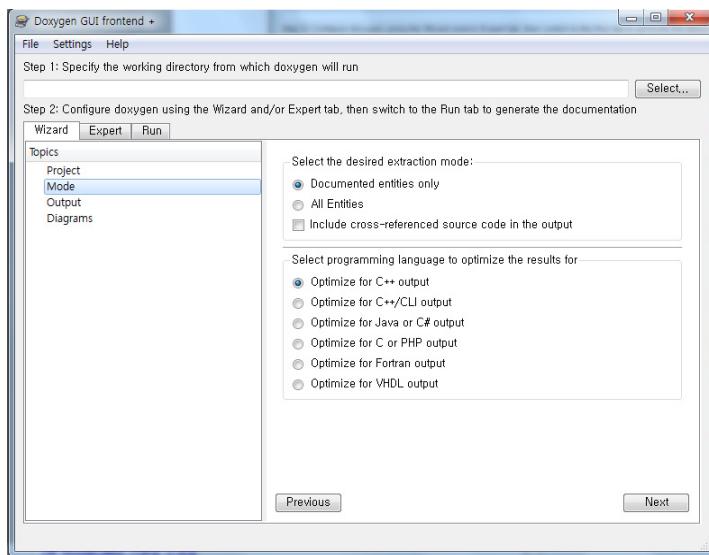
설치 완료 후 Doxywizard 실행파일을 실행하면 다음 페이지의 그림과 같은 GUI 가 보여진다. Wizard는 간단하게 마법사 형식으로 Doxygen의 기능을 사용할 수 있다. 처음 작업하는 경우, Wizard 기능을 이용하는 것을 권장한다.

각 항목들의 설정은 다음과 같다.

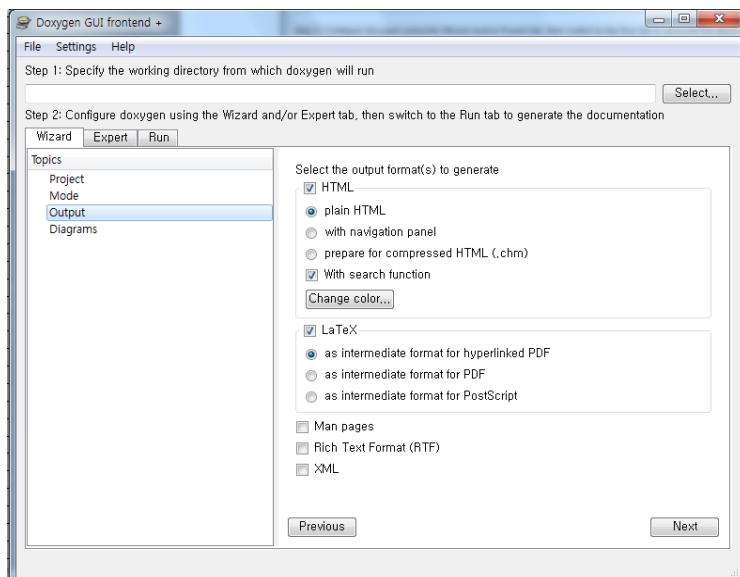


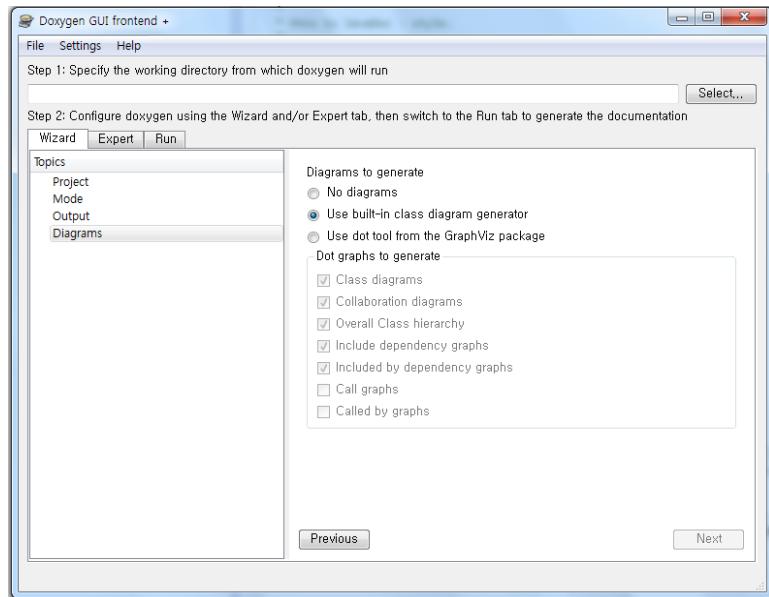
- Specify the working directory from which doxygen will run
: 프로젝트 폴더를 지정하며, 소스 코드와 Doxygen 결과 문서가 저장될 폴더를 포함한다.
- Project name : 프로젝트 이름 (한글 출력도 완벽하게 지원되므로, 한글을 사용해도 무방하다.)
- Source code directory : 소스 코드가 있는 디렉토리를 선택해 주면 된다. 서브 폴더에도 소스 파일들이 있다면, Scan recursively를 체크해야 한다.
- Destination directory : 출력결과를 저장할 디렉토리를 지정해 주면 된다. 실제 결과물은 html, Latex 등 서브디렉토리를 만들고 그 밑에 저장하므로 꼭 비어있는 디렉토리가 아니어도 된다.

SW개발 품질관리 매뉴얼



- Select the desired extraction mode
 - : Doxygen이 소스 코드에서 찾는 범위를 설정한다.
기본값은 Documented entities only 이다.
- Select programming language to optimize the results for
 - : 결과물을 위하여 해당 프로그래밍 언어를 선택하면 된다.





- Select the output format(s) to generate : 출력될 문서의 형식을 지정해 준다.
- Diagrams to generate : Doxygen에서는 소스 코드들의 관계를 그래프로 출력해 주는 기능이 있다. 이에 대한 설정을 지원한다.
- 이제 문서 출력을 위한 기본적인 설정은 완료하였다. Run 탭 메뉴로 진입하여 Run doxygen 버튼을 클릭하면 진행창을 통하여 output이 만들어지는 과정을 확인할 수 있다. 또한, Expert 탭 메뉴를 통하여 다양한 상세 설정에 접근할 수 있다.

다. Doxygen 출력 문서 확인

간단한 Doxygen 출력 문서의 예제를 보면 다음과 같다. HTML 형식으로 출력한 내용이다.

My Doxygen Test 1.0.0

Main Page Files File List Globals Functions

Search

docTest.c File Reference

Go to the source code of this file.

Functions

void **myDoxygenTestFunc** (char **input1**,
 int **input2**
)

this is test function

Returns
no return value

Parameters
 input1 first test parameter
 input2 second test parameter

Definition at line 15 of file **docTest.c**.

VII

프로세스의 문서화

1. Redmine 및 Impasse 설정
2. 프로젝트 산출물
3. Redmine Table 분석
4. Impasse Table 분석
5. 프로젝트 산출물 생성용 SQL

VII 프로세스의 문서화

SW개발은 그 과정에서 많은 산출물을 발생시키며, SW공학의 관점에서 이러한 산출물은 체계적으로 관리되어야 한다. 지원도구의 활용은 도구의 활용 그 자체만으로 산출물이 관리되고 있음을 의미하나, 기업의 산출물 관리, 발주자의 요구, 기타 인증 및 심사 과정에서의 필요에 따라 개발 산출물은 시스템 그 자체가 아닌 문서화된 형태로 작성되어야 할 수 있다. 따라서 본 절에서는 지원도구의 활용에 따라 내부적으로 관리되고 있는 데이터를 산출물 형태로 발생시키기 위한 방안을 소개한다.

1. Redmine 및 Impasse 설정

SW Visualization 지원도구를 활용하여 산출물을 생성하기 위해서는 SW Visualization 지원 도구의 기본 입력 필드 이외에 별도의 항목들이 관리되어야 한다. Redmine 및 Impasse에서는 사용자 정의 기능을 활용하여 추가적인 항목들을 입력하고 관리할 수 있다. 본 장에서는 이와 관련하여 산출물 생성을 위하여 필요한 Redmine 및 Impasse의 설정에 대하여 정리한다.

본 문서에서의 사용자 정의 항목 추가 및 설정은 산출물 문서화를 위한 최소한의 설정이며 사용자의 필요에 따라 별도의 항목을 추가할 수 있다.

가. Redmine 설정

Redmine의 설정은 요구사항 명세서 및 요구사항 추적표의 생성을 위하여 일감 유형의 정의와 사용자 정의항목 추가가 필요하다.

➤ 일감 유형

Impasse의 활용을 위하여 요구사항에 해당하는 일감 유형과 등록된 요구사항의 검증 결과 문제가 발생한 경우 결함 사항의 등록을 위하여 사용되는 일감 유형이 존재하여야 한다.

이름	형식	필수	모든 프로젝트	사용됨	정렬	삭제
REQ_ID	문자열	✓	✓	△ ▲ ▽ ▾	▲	삭제
기능/비기능	목록	✓	✓	△ ▲ ▽ ▾	▲	삭제
출처	문자열	✓		△ ▲ ▽ ▾	▲	삭제
제약사항	텍스트	✓		△ ▲ ▽ ▾	▲	삭제

➤ 사용자 정의 항목

요구사항의 일감 유형 등록 시 추가적으로 등록되어야 하는 항목은 REQ_ID, 기능/비기능, 출처, 제약사항이 있으며 자세한 내용은 아래 그림과 같다.

이름	형식	필수	모든 프로젝트	사용됨	정렬	삭제
REQ_ID	문자열	✓	✓	△ ▲ ▽ ▾	▲	삭제
기능/비기능	목록	✓	✓	△ ▲ ▽ ▾	▲	삭제
출처	문자열	✓		△ ▲ ▽ ▾	▲	삭제
제약사항	텍스트	✓		△ ▲ ▽ ▾	▲	삭제

VII. 프로세스의 문서화

사용자 정의 항목 » 일감 » 기능/비기능

이름 *

형식

복수선택가능

가능한 값들

각 값 당 한 풀

기본값

일감 유형
 결합 요구사항

필수
모든 프로젝트
검색조건으로 사용됨
검색가능

관리

- 프로젝트
- 사용자
- 그룹
- 역할 및 권한
- 일감 유형
- 일감 상태
- 업무흐름
- 사용자 정의 항목
- 코드값
- 설정
- LDAP 인증
- 플러그인
- 정보

사용자 정의 항목 » 일감 » REQ_ID

이름 *

형식

최소 - 최대 길이 -
(0는 제한이 없음을 의미함)

정규식
(예) ^[A-Z0-9]+\$

기본값

일감 유형
 결합 요구사항

필수
모든 프로젝트
검색조건으로 사용됨
검색가능

관리

- 프로젝트
- 사용자
- 그룹
- 역할 및 권한
- 일감 유형
- 일감 상태
- 업무흐름
- 사용자 정의 항목
- 코드값
- 설정
- LDAP 인증
- 플러그인
- 정보

사용자 정의 항목 » 일감 » 출처

이름 *

형식

최소 - 최대 길이 -
(0는 제한이 없음을 의미함)

정규식
(예) ^[A-Z0-9]+\$

기본값

일감 유형
 결합 요구사항

필수
모든 프로젝트
검색조건으로 사용됨
검색가능

관리

- 프로젝트
- 사용자
- 그룹
- 역할 및 권한
- 일감 유형
- 일감 상태
- 업무흐름
- 사용자 정의 항목
- 코드값
- 설정
- LDAP 인증
- 플러그인
- 정보

SW개발 품질관리 매뉴얼

사용자 정의 항목 » 일감 » 제약사항

이름 * 제약사항
형식 텍스트
최소 - 최대 길이 [0] - [0]
(0는 제한이 없음을 의미함)
정규식
(예) ^[A-Z0-9]+\$
기본값

일감 유형
 결합 요구사항

필수
모든 프로젝트
검색 조건으로 사용됨
검색 가능

관리
프로젝트
사용자
그룹
역할 및 권한
일감 유형
일감 상태
업무흐름
사용자 정의 항목
코드값
설정
LDAP 인증
플러그인
정보

나. Impasse 설정

테스트 시나리오 및 결과서와 요구사항 추적표의 생성을 위하여 아래와 같은 사용자 정의 항목 추가가 필요하다. Impasse의 사용자 정의 항목은 Redmine 설정의 플러그인 메뉴에서 Impasse 설정을 통해 추가 가능하다.

사용자 정의 항목

	이름	형식	필수	정렬	
ID		문자열	✓	△ ▲ ▽ ▾	삭제
시험종류		목록	✓	△ ▲ ▽ ▾	삭제
시험유형		문자열		△ ▲ ▽ ▾	삭제

새 사용자 정의 항목

사용자 정의 항목 » Test cases » ID

이름 *

형식

최소 - 최대 길이 -
(0는 제한이 없음을 의미함)

정규식

(예) ^[A-Z0-9]+\$

기본값

필수

사용자 정의 항목 » Test cases » 시험종류

이름 *

형식

복수선택가능

가능한 값들

각 값 당 한 줄

기본값

필수

사용자 정의 항목 » Test cases » 시험유형

이름 *

형식

최소 - 최대 길이 -
(0는 제한이 없음을 의미함)

정규식

(예) ^[A-Z0-9]+\$

기본값

필수

2. 프로젝트 산출물

프로젝트 산출물은 프로젝트의 특성 혹은 상황에 따라 다양한 형태의 산출물이 요구될 수 있다. 그러나 본 절에서는 일반적인 SW 프로젝트에서 요구되는 대표적인 산출물로 요구사항 명세서, 요구사항 추적표, 테스트 시나리오, 테스트 결과서의 문서화를 위하여 지원도구와 명세서 항목 간의 관계에 대해 확인한다.

가. 요구사항 명세서

SW Visualization에서 요구사항은 Redmine을 통하여 관리된다. 따라서 요구사항 명세서의 문서화를 위해서는 Redmine과 요구사항 명세서 항목 간의 연결이 필요하다.

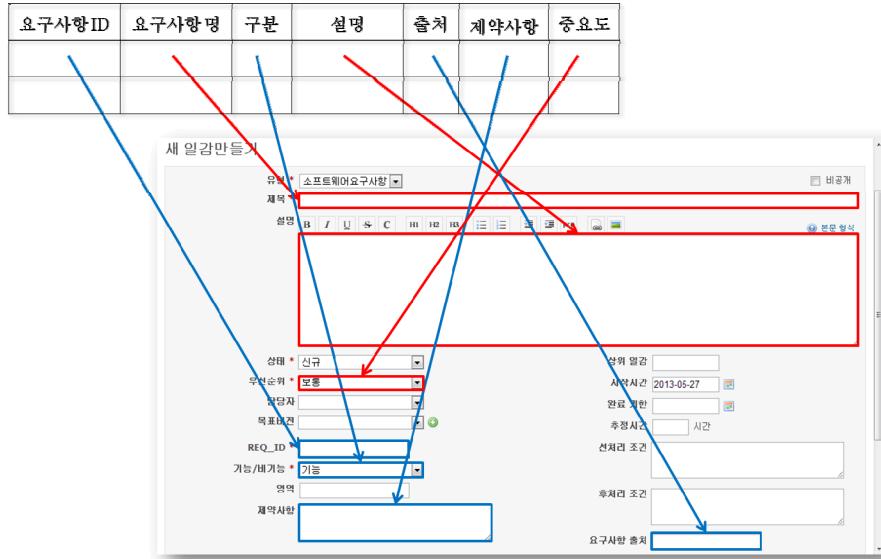
아래 그림은 일반적으로 요구사항 명세서에 포함되는 항목과 Redmine의 항목 간의 연결 관계를 표시한 그림이다. Redmine에서는 사용자의 필요에 따라 사용자 정의 항목을 추가할 수 있으며, 그림의 파란 부분은 사용자 정의 항목으로 별도 정의한 내용이다.

이와 유사한 방법으로 요구사항 명세서에 추가로 포함되어야 하는 항목이 있다면 Redmine에 사용자 정의 항목을 활용함으로써 이후 요구사항 명세서의 문서화에 대비 할 수 있다.

요구사항 명세서

프로젝트명						
버전				작성일자		
요구사항 ID	요구사항 명	구분	설명	출처	제약사항	중요도

〈요구사항 명세서 양식〉



<요구사항 명세서와 지원도구>

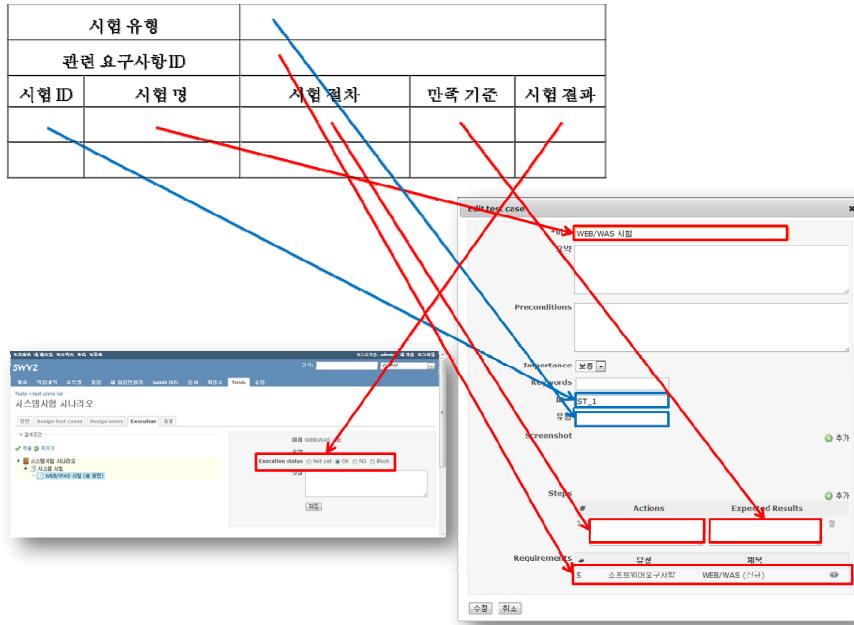
나. 테스트 시나리오 및 결과서

SW Visualization 지원도구를 활용한 테스트는 크게 자동화 테스트와 매뉴얼 테스트로 구분할 수 있다. 자동화 테스트는 Jenkins를 중심으로 하여 PMD, JUnit, NSIQ Collector 등의 플러그인을 활용하여 수행하는 부분이며, 매뉴얼 테스트는 Redmine의 Impasse를 활용하는 부분이다. 일반적인 SW공학에서 테스트는 크게 단위 테스트, 통합 테스트, 시스템 테스트, 인수 테스트의 4가지 종류로 구분한다. 테스트의 방법 및 지원도구는 프로젝트의 성격이나 기업체의 특성에 따라 상이할 수 있으나 본 절에서는 단위 테스트는 Jenkins의 JUnit을 활용하는 것으로 가정하며, 그 외의 통합 테스트, 시스템 테스트, 인수 테스트는 Impasse를 활용하는 것으로 가정하고 문서화 관련 작업을 진행한다.

JUnit을 활용한 단위 테스트의 테스트를 위해서는 개발자의 테스트 코드 개발이 필요하다. 따라서 테스트 코드와 시나리오가 함께 관리되는 것이 가장 이상적이며 따라서 실제로 JavaDoc 혹은 Doxygen을 활용하여 JUnit의 테스트 시나리오를 관리하는 방

SW개발 품질관리 매뉴얼

법이 일반적이다. JavaDoc 혹은 Doxygen을 활용하는 방법은 개발자를 위한 문서화 부분에서 언급하였으므로 여기에서는 별도의 언급을 하지 않는다.



〈테스트 시나리오/결과서와 지원도구〉

시험 시나리오/결과서

프로젝트명				
버전		작성일자		
시험유형				
관련 요구사항 ID				
시나리오 ID	시나리오 명	절차	만족기준	시험결과

〈시험 시나리오/결과서 양식〉

다. 요구사항 추적표

요구사항 추적표의 경우 일반적으로 요구사항, 설계서, 소스 코드, 테스트 결과를 상호 추적할 수 있는 형태로 구성한다. 그러나 SW Visualization에서 현재 설계를 관리하는 도구가 포함되어 있지 않으므로 설계를 제외한 요구사항과 소스코드, 테스트 결과를 상호 추적할 수 있는 형태로 구성한다. 이 항목은 앞서 언급한 Redmine 및 Impasse를 활용하여 구성이 가능하다.

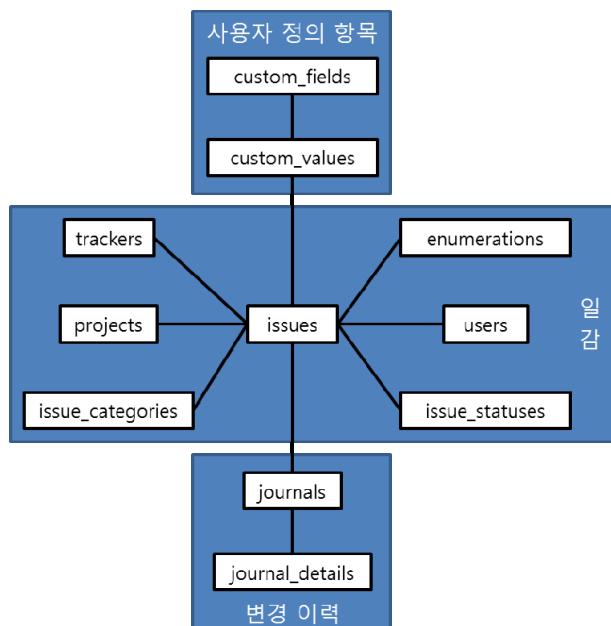
요구사항 추적표						
프로젝트명						
버전				작성일자		
요구사항 ID	유스케이스 ID	아키텍처 설계서	프로그램 코드	단위시험 결과서	통합시험 결과서	시스템시험 결과서

〈요구사항 추적표 양식〉

3. Redmine table 분석

SW Visualization에서 활용하는 Redmine은 요구사항 및 테스트 관리를 위하여 매우 많은 데이터를 내부에 저장한다. 실제로 Redmine은 MySQL을 활용하여 데이터를 저장하고 있다. SW Visualization에서는 개발 산출물의 문서화를 위하여 Redmine의 MySQL의 구조를 분석하고 이를 기반으로 앞서 확인한 산출물 문서화에 필요한 항목을 Database에서 추출함으로써 산출물의 문서화를 진행하고자 한다. 따라서 본 절에서는 Redmine에서 활용하고 있는 MySQL에서 산출물 문서화를 위하여 필요한 내용을 확인한다.

Redmine의 Database는 크게 Redmine 자체에 포함되어 있는 테이블과 Impasse를 위하여 포함된 테이블로 구분된다. Redmine 자체에 포함되어 있는 테이블은 다시 일감 관련 부분, 일감의 수정 이력 관리를 위한 부분, 사용자 정의 항목 관리를 위한 부분으로 구분된다.



〈Redmine DB Table 구조〉

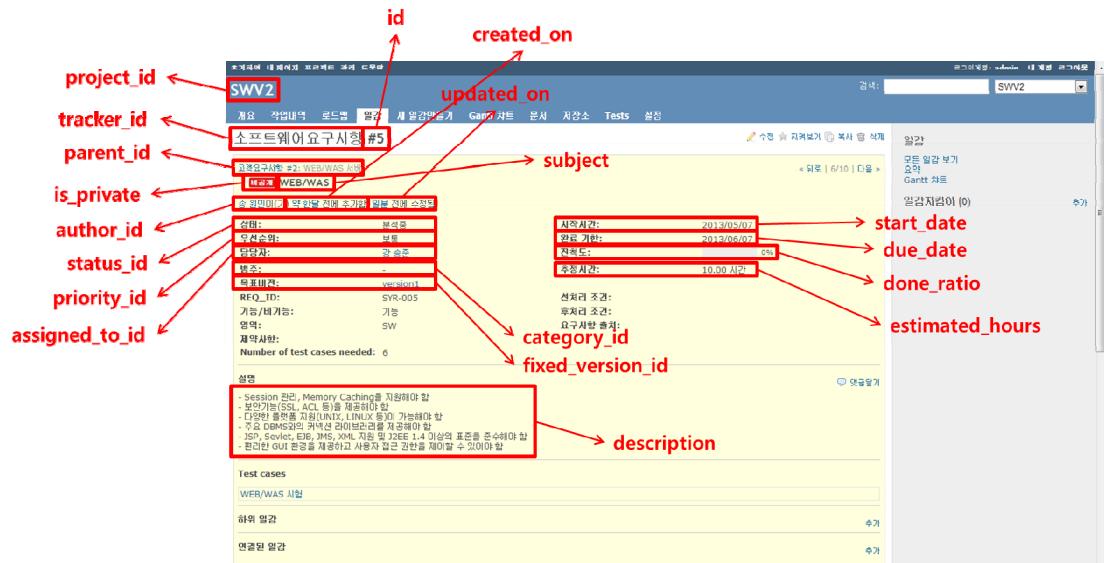
가. issues

Redmine의 일감을 관리하는 Table로써, Column의 구성은 아래와 같다.

column	type	default	참조	일감 표시
id	int(11)	AI		
tracker_id	int(11)	0	trackers id	trackers name
project_id	int(11)	0	projects id	projects name
subject	varchar(255)	''		
description	text			
due_date	date	null		
category_id	int(11)	null	issue_categories id	issue_categories name
status_id	int(11)	0	issue_statuses id	issue_statuses name
assigned_to_id	int(11)	null	users id	users firstname, lastname
priority_id	int(11)	0	enumerations id	enumerations name
fixed_version_id	int(11)	null	versions id	versions name
author_id	int(11)	0	users id	users firstname, lastname
lock_version	int(11)	0		
created_on	datetime	null		
updated_on	datetime	null		
start_date	date	null		

SW개발 품질관리 매뉴얼

column	type	default	참조	일감 표시
done_ratio	int(11)	0		
estimated_hours	float	null		
parent_id	int(11)	null	issues id	
root_id	int(11)	null	issues id	
lft	int(11)	null		
rgt	int(11)	null		
is_private	tinyint(1)	0		



나. journals

column	type	default	참조	비고
id	int(11)	AI		
journalized_id	int(11)	0	issues_id	관련 이슈 ID
journalized_type	varchar(30)	" "		이슈에 대한 이력인 경우 'Issue'
user_id	int(11)	0	users_id	변경한 사용자 ID
notes	text			덧글
created_on	datetime			변경한 시간
private_notes	tinyint(1)	0		덧글 비공개 여부

다. journal_details

column	type	default	참조	비고
id	int(11)	AI		
journal_id	int(11)	0	journals_id	관련 journals ID
property	varchar(30)	" "		일감 변경의 경우 'attr'
prop_key	varchar(30)	" "		값이 변경된 column 이름
old_value	text			이전 값
value	text			변경된 값

라. custom_values

column	type	default	참조	일감 표시	비고
id	int(11)	AI			
customized_type	varchar(30)	''			일감 관련 항목인 경우 'Issue'
customized_id	int(11)	0	issues id		관련 일감 ID
custom_field_id	int(11)	0	custom_ fields id	custom_ fields name	추가된 사용자 정의 항목 ID
value	text				해당 사용자 정의 항목 값

마. projects

column	type	default	참조	일감 표시
id	int(11)	AI		
name	varchar(255)	''		
description	text	null		
homepage	varchar(255)	null		
is_public	tinyint(1)	1		
parent_id	int(11)	null	projects id	projects name
created_on	datetime	null		
updated_on	datetime	null		
identifier	varchar(255)	null		
status	int(11)	1		

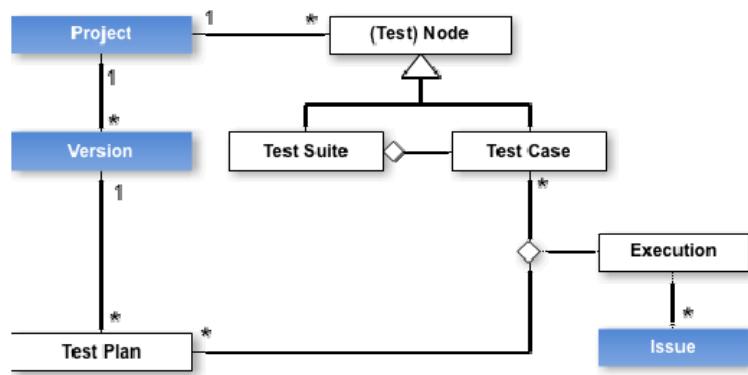
column	type	default	참조	일감 표시
lft	int(11)	null		
rgt	int(11)	null		
inherit_members	tinyint(1)	0		

8. enumerations

column	type	default	참조	일감 표시
id	int(11)	AI		
name	varchar(255)	"		
description	text	null		
position	int(11)	1		
is_default	tinyint(1)	0		
type	varchar(255)	null		
active	tinyint(1)	1		
project_id	int(11)	null	projects id	projects name
parent_id	int(11)	null	enumerations id	enumerations name
position_name	varchar(30)	null		

4. Impasse table 분석

Impasse는 Redmine의 plugin으로써 Redmine의 Database를 함께 사용한다. 그러나 table은 별도 구성되어 동작하며, 본 장에서는 Impasse 동작과 관련하여 문서화에 필요한 table에 대한 구조를 파악한다.



<Impasse Plugin DB Table 구조>

가. impasse_nodes

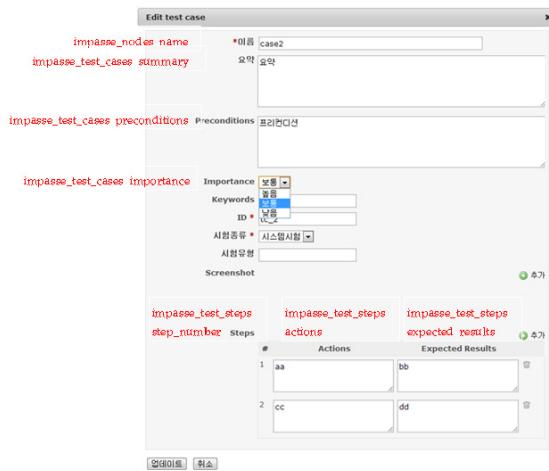
column	type	default	참조	표시	비고
id	int(11)	AI			TestSuite 또는 TestCase Id
name	varchar(255)	null			TestSuite 또는 TestCase 이름
node_type_id	int(11)	null	impasse_node_type_id	impasse_node_type_name	Node 종류 1 testproject 2 testsuite 3 testcase
parent_id	int(11)	null	impasse_node_id	impasse_node_name	부모 Project 또는 부모 TestSuite
node_order	int(11)	null			순서
path	varchar(255)	null			위치

나. impasse_node_type

column	type	default	참조	표시	비고
id	int(11)	AI			1,2,3
description	varchar(255)	null			1 testproject 2 testsuite 3 testcase

다. impasse_test_cases

column	type	default	참조	표시	비고
id	int(11)	AI	impasse_nodes id	impasse_nodes name	
summary	text				요약
preconditions	text				사전조건
importance	int(11)				높음:1 보통:2 낮음:3
active	tinyint(1)	1			활용 여부



라. `impasse_test_steps`

column	type	default	참조	표시	비고
id	int(11)	AI			
test_case_id	int(11)		impasse_test_cases id	impasse_test_cases name	
step_number	int(11)				스텝 번호
actions	text				실행
expected_results	text				예상 결과

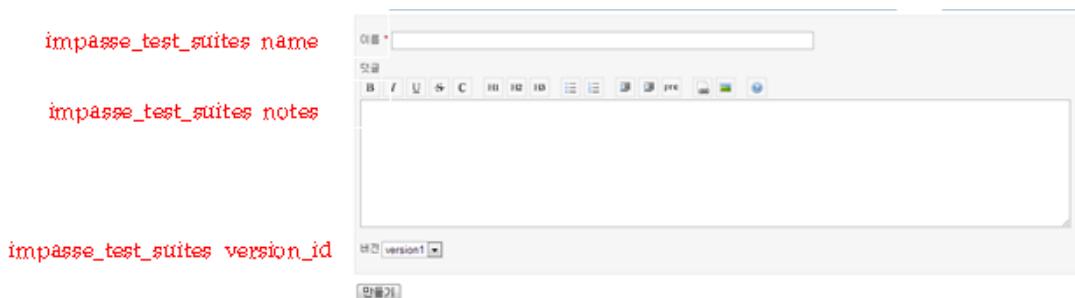
마. `impasse_test_suites`

column	type	default	참조	표시	비고
id	int(11)	AI	impasse_nodes id	impasse_nodes name	
details	text				설명

The screenshot shows a modal dialog titled "Edit test suite". It has two tabs: "Details" (which is selected) and "Keywords". In the "Details" tab, there is a field labeled "Name" with the placeholder "suite" and a "Description" field with the placeholder "디테일". Below the tabs are several small buttons for text styling (B, I, U, S, C, H1, H2, H3, etc.). At the bottom of the dialog are two buttons: "업데이트" (Update) and "취소" (Cancel).

비. impasso_test_plans

column	type	default	참조	표시	비고
id	int(11)	AI			
version_id	int(11)		version id	version name	버전
name	varchar(255)				이름
notes	text				설명
active	tinyint(1)				활용 여부



사. impasso_test_plan_cases

column	type	default	참조	표시	비고
id	int(11)	AI			
test_plan_id	int(11)		impasso_test_ plans id	impasso_test_ plans name	테스트 계획 ID
test_case_id	int(11)		impasso_test_ cases id	impasso_test_ cases summary	테스트 케이스 ID
node_order	int(11)				

아. impass_node_keywords

column	type	default	참조	표시	비고
id	int(11)	AI			
node_id	int(11)		impass_nodes id	impass_nodes name	
keyword_id	int(11)		impass_keywords id	impass_keywords name	

자. impass_keywords

column	type	default	참조	표시	비고
id	int(11)	AI			
keyword	varchar(255)	null			키워드
project_id	int(11)	null	project id	project name	

자. impass_executions

column	type	default	참조	일감 표시
id	int(11)	AI		
test_plan_case_id	int(11)		impass_test_plan_ cases id	
tester_id	int(11)	null	users id	users firstname, lastname
build_id	int(11)	null		
expected_date	date	null		

column	type	default	참조	일감 표시
status	varchar(255)	'0'		
execution_ts	datetime	null		
notes	text	null		
executor_id	int(11)	null	users id	users firstname, lastname
position_name	varchar(30)	null		

캬. impasse_requirement_issues

column	type	default	참조	표시	비고
id	int(11)	AI			
issue_id	int(11)	null	issue id	issue name	요구사항 ID
num_of_cases	int(11)	null			테스트 케이스 개수

탸. impasse_requirement_cases

column	type	default	참조	비고
id	int(11)	AI		
requirement_id	int(11)	null	impasse_requirement_issues id	요구사항 ID
test_case_id	int(11)	null	impasse_test_cases id	연관된 테스트 케이스 ID

5. 프로젝트 산출물 생성용 SQL

가. 요구사항 명세서

요구사항 명세서						
프로젝트명						
버전				작성일자		
요구사항 ID	요구사항 명	구분	설명	출처	제약사항	중요도

➤ 요구사항 명세서 View 생성 SQL

```

CREATE VIEW 요구사항_명세서 AS
SELECT
    (SELECT P.name FROM projects P where P.id = l.project_id) AS 프로젝트명,
    (SELECT V.name FROM versions V where V.id = l.fixed_version_id) AS 버전명,
    (SELECT CV.value FROM custom_fields CF INNER JOIN custom_values CV ON
    CF.id = CV.custom_field_id WHERE CF.name='REQ_ID' AND CV.customized_id=l.id)
    AS 요구사항ID,
    l.subject AS 요구사항명,
    (SELECT CV.value FROM custom_fields CF INNER JOIN custom_values CV ON
    CF.id = CV.custom_field_id WHERE CF.name='기능/비기능' AND CV.customized_id=l.id)
    AS 구분,
    l.description AS 설명,
    (SELECT CV.value FROM custom_fields CF INNER JOIN custom_values CV ON
    CF.id = CV.custom_field_id WHERE CF.name='출처' AND CV.customized_id=l.id)
    AS 출처,
    (SELECT CV.value FROM custom_fields CF INNER JOIN custom_values CV ON
    
```

```
CF.id = CV.custom_field_id WHERE CF.name='제약사항' AND CV.customized_id=I.id) AS 제약사항,  
(SELECT E.name FROM enumerations E WHERE E.id = I.priority_id) AS 중요도  
FROM issues I  
WHERE I.tracker_id = (SELECT T.id FROM trackers T where T.name='요구사항')  
ORDER BY 프로젝트명, 버전명, 요구사항ID;
```

➤ 사용 예제

```
SELECT *  
FROM 요구사항_명세서  
WHERE 프로젝트명='Project01' AND 버전명='v1.0';
```

나. 테스트 시나리오 및 결과서

시험 시나리오/결과서						
프로젝트명						
버전		작성일자				
시험유형						
관련 요구사항 ID						
시나리오 ID	시나리오 명	절차	만족기준	시험결과		

➤ 시험 시나리오 결과서 View 생성 SQL

```

CREATE VIEW 시험시나리오_결과서 AS
SELECT
  (SELECT P.name FROM projects P where P.id = l.project_id) AS 프로젝트명,
  (SELECT V.name FROM versions V where V.id = l.fixed_version_id) AS 버전명,
  (SELECT CV.value FROM custom_fields CF INNER JOIN custom_values CV ON
  CF.id = CV.custom_field_id WHERE CF.name='시험유형' AND CV.customized_
  id=TC.id) AS 시험유형,
  (SELECT CV.value FROM custom_fields CF INNER JOIN custom_values CV ON
  CF.id = CV.custom_field_id WHERE CF.name='REQ_ID' AND CV.customized_id=l.id)
  AS 관련요구사항ID,
  l.subject AS 요구사항명,
  TP.name AS 테스트플랜명,
  (SELECT CV.value FROM custom_fields CF INNER JOIN custom_values CV ON
  CF.id = CV.custom_field_id WHERE CF.name='ID' AND CV.customized_id=TC.id)

```

```

AS 시나리오ID,
N.name AS 시나리오명,
TS.step_number AS TS_STEP_NUM,
TS.actions AS 절차,
TS.expected_results AS 만족기준,
CASE E.status WHEN '0' THEN 'Not yet' WHEN '1' THEN 'OK' WHEN '2' THEN
'NG' WHEN '3' THEN 'Block' END AS 시험결과
FROM issues I
LEFT OUTER JOIN impasse_requirement_issues RI ON I.id = RI.issue_id
LEFT OUTER JOIN impasse_requirement_cases RC ON RI.id = RC.requirement_id
LEFT OUTER JOIN impasse_test_cases TC ON RC.test_case_id = TC.id
LEFT OUTER JOIN impasse_nodes N ON TC.id = N.id
LEFT OUTER JOIN impasse_test_steps TS ON TC.id = TS.test_case_id
LEFT OUTER JOIN impasse_test_plan_cases TPC ON TC.id = TPC.test_case_id
LEFT OUTER JOIN impasse_executions E ON TPC.id = E.test_plan_case_id
LEFT OUTER JOIN impasse_test_plans TP ON TP.id = TPC.test_plan_id
WHERE I.tracker_id = (SELECT T.id FROM trackers T WHERE T.name='요구사항')
ORDER BY 프로젝트명, 버전명, 시험유형, 관련요구사항ID, 테스트플랜명, 시나리
오ID, TS_STEP_NUM;

```

➤ 사용 예제

```

SELECT *
FROM 시험시나리오_결과서
WHERE 프로젝트명='Project01' AND 버전명='v1.0';

```

다. 요구사항 추적표

요구사항 추적표						
프로젝트명						
버전				작성일자		
요구사항 ID	유스케이스 ID	아키텍처 설계서	프로그램 코드	단위시험 결과서	통합시험 결과서	시스템시험 결과서

➤ 요구사항 추적표 View 생성 SQL

```

CREATE VIEW 요구사항_추적표 AS
SELECT
  (SELECT P.name FROM projects P where P.id = I.project_id) AS 프로젝트명,
  (SELECT V.name FROM versions V where V.id = I.fixed_version_id) AS 버전명,
  (SELECT CV.value FROM custom_fields CF INNER JOIN custom_values CV ON
  CF.id = CV.custom_field_id WHERE CF.name='REQ_ID' AND CV.customized_id=I.id)
  AS 요구사항ID,
  C.path AS 프로그램코드,
  TP.name AS 테스트플랜명,
  (SELECT CV.value FROM custom_fields CF INNER JOIN custom_values CV ON
  CF.id = CV.custom_field_id WHERE CF.name='ID' AND CV.customized_id=TC.id)
  AS 시나리오ID,
  N.name AS 시나리오명,
  CASE E.status WHEN '0' THEN 'Not yet' WHEN '1' THEN 'OK' WHEN '2' THEN
  'NG' WHEN '3' THEN 'Block' END AS 시험결과
FROM issues I
  
```

```
LEFT OUTER JOIN impasse_requirement_issues RI ON I.id = RI.issue_id  
LEFT OUTER JOIN impasse_requirement_cases RC ON RI.id = RC.requirement_id  
LEFT OUTER JOIN impasse_test_cases TC ON RC.test_case_id = TC.id  
LEFT OUTER JOIN impasse_nodes N ON TC.id = N.id  
LEFT OUTER JOIN impasse_test_plan_cases TPC ON TC.id = TPC.test_case_id  
LEFT OUTER JOIN impasse_executions E ON TPC.id = E.test_plan_case_id  
LEFT OUTER JOIN impasse_test_plans TP ON TP.id = TPC.test_plan_id  
LEFT OUTER JOIN changesets_issues CI ON CI.issue_id = I.id  
LEFT OUTER JOIN changes C ON C.changeset_id = CI.changeset_id  
WHERE I.tracker_id = (SELECT T.id FROM trackers T where T.name='요구사항')  
ORDER BY 프로젝트명, 버전명, 요구사항ID, 프로그램코드, 테스트플랜명, 시나리오ID;
```

➤ 사용 예제

```
SELECT *  
FROM 요구사항_추적표  
WHERE 프로젝트명='Project01' AND 버전명='v1.0';
```


VIII

사례

- 요구사항 명세서
- 테스트 명세서
- SW아키텍처 설계 문서 양식

요구사항 명세서

■ 문서정보

이 문서는 소프트웨어 공학센터에서 중소기업에서 요구사항 명세서를 작성할 때, 참조할 수 있는 예로 작성된 문서이다. 이 문서는 크게 2개의 Part로 구성되어 있다. Part I은 Case Tool에 입력한 형태의 요구사항 예를 보여주고 있으며, Part II는 문서로 작성한 형태의 요구사항 예를 보여주고 있다.

Part I은 개발자가 개발할 때 사용하는 시스템 요구사항 내용이 주로 정리 되어 있으면 Case Tool의 특성에 따라 요구사항 관리에 필요한 내용이 추가되어 있다. 여기서는 이슈관리 Tool인 Redmine을 요구사항 관리 Tool로 사용하고 있으면 Redmine에서 요구사항 관리로 필요한 내용을 추가로 정리되어 있다.

Part II는 과제에 관련된 모든 이해관계자의 의견을 반영하고 이해할 수 있는 형태의 예로 작성된 요구사항으로 비즈니스 요구사항, 사용자 요구사항, 그리고 시스템 요구사항까지 작성한 예이다. 특히 여기서는 사용자 요구사항과 시스템 요구사항을 더욱 구체화 위하여 Use Case Modeling, QAW 방법론을 사용하였다.

요구사항 명세서는 과제의 도메인 특성, 요구사항 분석 방법론 그리고 Case Tool에 따라 작성한 형태 및 내용에 차이가 있기 때문에 각각의 중소기업의 특성에 맞추어 요구사항 명세의 형태를 사용할 필요가 있다.

이 문서는 요구사항 문서의 사례이기 때문에 한 사례의 모든 요구사항을 기술하고 있지 않는다.

Part I Case Tool 관리 관점의 요구사항 작성 사례

Part I은 실제 시스템에서 요구사항을 등록할 때 사용되는 형식으로 사례를 정리하였다. 요구공학은 크게 요구사항 개발과 요구사항 관리 영역으로 나누어져 있다. 요구사항 개발은 도출, 분석, 명세화, 검증 및 확인으로 구성되어 있고 요구사항 관리는 요구사항 변경 관리, 추적 관리로 구성되어 있다. 이 Part I에서는 요구공학 내에서 요구사항 관리에 초점을 맞추어 정리된 부분이다.

Part I에서 각각의 요구사항들은 “요구사항 테이블 작성 형태”와 “Redmine Tool 관리 형태” 두 가지로 구성되어 있다. “요구사항 테이블 작성 형태”와 “Redmine Tool 관리 형태”는 동일한 형식으로 구성되어 있는데, “요구사항 테이블 작성 형태”에서는 독자가 읽기 쉽도록 정리 되어 있고, “Redmine Tool 관리 형태”에서는 “요구사항 테이블 작성 형태” 내용이 Redmine Tool의 이용하여 작성한 것을 보여 주고 있다.

“요구사항 테이블 작성 형태”的 구성은 내용과 관리로 크게 2가지로 구성되어 있다. 이해관계자들끼리 요구사항을 이해하고 협의하기 위하여 작성한 내용 부분은 “설명”, “선처리 조건”, “후처리 조건”, “제약사항” 그리고 “기타”에서 포함이 되어 있고, 요구사항을 관리하기 위한 부분은 “요구사항 ID”, “중요도” 그리고 “요구사항 변경 이력”에 포함이 되어 있다. Part I에서는 주로 요구사항 관리를 중심으로 내용을 작성되어 있기 때문에 요구사항이 개발 되는 과정을 이해하는데 Part I의 사례는 한계가 있다. 그래서 요구사항의 개발 과정을 이해하기 위해서는 Part II의 “문서 작성 과정의 요구사항 작성”을 참조할 필요가 있다.

Part I에서 사용된 예제는 파일시스템의 기능과 비기능 요구사항을 예제로 정리하여 기술하였다. 다양한 요구사항 중 파일시스템의 예제를 사용하여 기술한 이유는 개발자면 누구나 파일 시스템을 사용하기 때문에 이 시스템의 기능을 쉽게 알 수 있을 뿐만 아니라, 오래 전부터 만들고 정리되어 왔기 때문에 정형화된 형식을 가지고 있기 때문이다. 특히 기능 요구사항은 블룸 관리와 파일 관리의 요구사항을 일부 정리하였고, 비기능 요구사항은 성능에 관련된 요구사항을 정리하였다.

1. 볼륨 관리

1.1. 볼륨 생성

1.1.1. 정상적인 볼륨 생성

« 요구사항 테이블 작성 형태 »

요구사항 ID		요구사항 명		
RF-VOL-CREATE-BFLOW		정상적으로 볼륨 만들기		
유형	영역	흐름 분류	중요도	진행 과제
기능	볼륨 관리	기본 흐름	상	1차
설명	어플리케이션 개발자는 저장장치에 자신이 원하는 데이터를 기록하고 싶어한다. 저장장치에 데이터를 기록하고 위해서는 전자제품의 운영시스템이 저장 장치를 논리적인 이해할 수 있는 형태로 만들어야 하면, 이 논리적 형태를 가지고 어플리케이션 개발자는 자신이 데이터 기록을 할 수 있게 된다. 이와 같은 데이터를 저장 할 수 있는 논리적 기능을 만드는 것은 이 “볼륨 만들기” 기능이다.			
선처리 조건	전자제품의 운영 시스템은 저장장치가 있다는 것을 인지할 수 있어야 한다. 그리고 운영 시스템은 모든 저장장치를 등록하고 관리할 수 있어야 한다.			
후처리 조건	관련 사항 없음			
상세내역	1	어플리케이션 개발자가 생성하려고 하는 볼륨의 시작 위치를 정한다.		
	2	어플리케이션 개발자가 생성하려고 하는 볼륨의 크기를 정한다.		
	3	어플리케이션 개발자가 생성하려고 하는 볼륨의 모드를 선택한다.		
	4	어플리케이션 개발자가 볼륨 생성함수를 호출한다.		
	5	파일 시스템은 어플리케이션 개발자에게 볼륨 구분자를 준다		
제약사항	저장장치는 최소한 XXX GB까지 지원해야 한다.			
기타	여러 개의 장치가 붙어도 하나의 논리적인 공간으로 보이도록 만들어야 함 한 개의 장치가 있더라도 논리적으로 여러 개의 공간인 것처럼 보이게 하여야 함			
요구사항 변경 이력				

2. 파일 관리

2.1 파일 생성

2.1.1. 정상적인 파일 생성

« 요구사항 테이블 작성 형태 »

요구사항 ID		요구사항 명			
상세내역	유형	영역	흐름 분류	중요도	진행 과제
	기능	파일 관리	기본 흐름	상	1차
	설명	어플리케이션 개발자는 저장장치에 자신이 원하는 데이터를 기록할 수 있으며, 파일 시스템에서 파일 단위로 자신이 원하는 정보를 기록하고 관리한다. 파일시스템은 개발자에게 파일을 생성할 수 있는 기능을 함수로 제공하고 있다. 개발자가 파일 생성 함수를 사용하기 위해서는 파일의 이름, 파일 오픈 시 사용되는 모드 그리고 파일 접근 권한을 알아야 한다. 그리고 이 함수 통하여 개발자가 원하는 파일을 만들 수 있으면 컨트롤 할 수 있게 된다.			
	선처리 조건	논리적 저장 공간이 볼륨을 만들 수 있어야 하며, 이 볼륨을 관리 할 수 있어야 한다.			
	후처리 조건	개발자는 파일 생성 함수를 통하여 파일을 컨트롤 할 수 있는 파일 디스크립터를 얻을 수 있다. 이렇게 얻은 파일 디스크립터는 메모리 자원에 한계가 있기 때문에 사용한 파일 디스크립터는 다시 파일 시스템에 반납을 해야 한다.			
	시나리오	1	어플리케이션 개발자가 정해진 규칙에 따라 파일 이름을 생성한다.		
		2	어플리케이션 개발자가 파일 생성 함수를 호출한다.		
		3	파일 시스템은 어플리케이션 개발자에게 파일 디스크립터를 준다.		
	제약사항	파일 이름 생성 규칙파일 이름의 최대 길이는 255 특수 문자 “*,!” 사용불가 대소문자를 구별함 파일 오픈 모드			

	<p>APPEND : 쓰기전용 모드로 파일을 엽니다. 파일을 열때 파일이 없다면 생성합니다 쓰기전용 모드. 파일을 열 때 파일이 없다면 생성함</p> <p>READ : 읽기전용 모드. 파일이 반드시 있어야 함</p> <p>UPDATE : 읽기/쓰기 모드. 파일을 열 때 파일이 없다면 생성함</p> <p>WRITE : 쓰기전용 모드. 파일이 없다고 생성함. 파일이 있다면 덮어쓰기</p> <p>파일 접근 모드</p> <p>파일 접근은 3개의 그룹으로 이루어짐 (소유자, 특정 그룹, 그외 사람들)</p> <p>각각 읽고, 쓰고, 실행하기의 허가를 할 수 있음</p>
기타	정상적인 파일 생성의 시나리오에서는 파일 오픈 모드가 APPEND로 파일 접근 모드가 소유자로 Default로 설정이 된다.
요구사항 변경 이력	

⟨⟨ Redmine Tool 관리 형태⟩⟩

기능요구사항 #5

수정 작업시간 기록 ★ 기여보기 문서 삭제

« 목록 | 2/3 | 다음 »

[RF-FILE-CREATE-FLOW] 경상적인 파일 생성

User Name Admin01(기) 악 한시간 전에 추가함. 일본 미하 전에 수정됨.

상태:	정상	소요 시간:	-
우선 순위:	상	개발자:	개발자1 조
담당자:	-	제작 사항:	파일 이름, 생성 규칙파일 - 이름의 최대 길이는 255 - 둘째 문자는 사용불가 - 대소문자를 구별함
명주:	-	파일 오픈 모드:	- APPEND : 쓰기전용 모드로 파일을 엽니다. 파일을 열때 파일이 없다면 생성함 - READ : 읽기전용 모드. 파일이 반드시 있어야 함 - UPDATE : 읽기/쓰기 모드. 파일을 열 때 파일이 없다면 생성함 - WRITE : 쓰기전용 모드. 파일이 없다고 생성함. 파일이 있다면 덮어쓰기
목표奔전:	-	파일 접근 모드:	- 파일 접근은 3개의 그룹으로 이루어짐 (소유자, 특정 그룹, 그외 사람들) - 각각 읽고, 쓰고, 실행하기의 허가를 할 수 있음
명역:	파일 관리	경상적인 파일 생성의 시나리오에서는 파일 오픈 모드가 APPEND로 파일 접근 모드가 소유자로 Default로 설정이 된다.	

선처리 조건:

주의: 파일 접근 허가를 만들 수 있어야 하며, 이 허가를 관리 기준:

개발자는 파일 생성 험수를 통하여 파일을 험수를 할 수 있는 파일 디스크립터를 만들 수 있다. 이렇게 얻은 파일 디스크립터는 파일에 험수가 있기 때문에 디 사용한 파일 디스크립터는 다시 파일 시스템에 반납을 해야 한다.

후처리 조건:

주의: 파일 생성 험수를 사용하기 위해서는 파일의 이름, 파일 오픈 시 사용되는 모드 그리고 파일 접근 권한을 알아야 한다. 그리고 이 험수 통하여 개발자가 원하는 파일을 만들 수 있으므로 험수를 할 수 있게 된다.

설명

애플리케이션 개발자는 저장경치에 자신이 원하는 데이터를 기록할 수 있으며, 파일시스템에서 파일 단위로 자신이 원하는 정보를 기록하고 관리한다. 파일시스템은 개발자에게 파일을 생성할 수 있는 기능을 제공하고 있다. 개발자가 파일 생성 험수를 사용하기 위해서는 파일의 이름, 파일 오픈 시 사용되는 모드 그리고 파일 접근 권한을 알아야 한다. 그리고 이 험수 통하여 개발자가 원하는 파일을 만들 수 있으므로 험수를 할 수 있게 된다.

1. 애플리케이션 개발자가 경제적 규칙에 따라 파일 이름을 생성한다.
2. 애플리케이션 개발자가 파일 생성 험수를 호출한다.
3. 파일 시스템은 애플리케이션 개발자에게 파일 디스크립터를 준다.

댓글을 기록

2.1.2. 다양한 파일 오프 모드로 파일 생성

« 요구사항 테이블 작성 형태 »

요구사항 ID			요구사항 명					
RF-FILE-CREATE-AFLOW-1			다양한 파일 오프 모드로 파일 생성					
상세내역	유형	영역	흐름 분류	중요도	진행 과제			
	기능	파일 관리	대체 흐름	상	1차			
	설명	정상적인 파일 생성에서 대체 흐름으로 파일 오프 모드 선택이 있다. 파일 오프 모드에 따라 읽기 적용이 될 수도 있고 쓰기 적용이 될 수도 있다.						
	선처리 조건	RF-FILE-CREATE-BFLOW와 동일함						
	후처리 조건	RF-FILE-CREATE-BFLOW와 동일함						
	시나리오	1	어플리케이션 개발자가 정해진 규칙에 따라 파일 이름을 생성하고 오프 모드를 생성한다.					
		2	어플리케이션 개발자가 파일 생성 함수를 호출한다.					
		3	파일 시스템은 어플리케이션 개발자에게 오픈 모드에 알맞은 파일 디스크립터를 준다.					
	제약사항	RF-FILE-CREATE-BFLOW와 동일함						
	기타	특이사항 없음						
요구사항 변경 이력								

« Redmine Tool 관리 형태 »

기능요구사항 #6

[RF-FILE-CREATE-AFLOW-1] 다양한 파일 오프 모드로 파일 생성

User Name Admin[0](?) 4분 전에 추가함. 일본 이하 전역 수정됨.

상태:	생성	소요 시간:	-
무선 순위:	상		
담당자:	개발자1 조		
별주:	-		
목표 버전:	-		
영역:	파일 관리	제약사항:	RF-FILE-CREATE-BFLOW와 동일함
선처리 조건:	RF-FILE-CREATE-BFLOW와 동일함	기타:	특이 사항 없음
후처리 조건:	RF-FILE-CREATE-BFLOW와 동일함		
설명	정상적인 파일 오프 모드에서 대체 흐름으로 파일 오프 모드 선택이 있다. 파일 오프 모드에 따라 파일 이름을 생성하고 오프 모드를 생성한다. 1. 어플리케이션 개발자가 정해진 규칙에 따라 파일 이름을 생성하고 오프 모드를 생성한다. 2. 어플리케이션 개발자가 파일 생성 함수를 호출한다. 3. 파일 시스템은 어플리케이션 개발자에게 오픈 모드에 알맞은 파일 디스크립터를 준다.		

2.1.3. 다양한 파일 접근 모드로 파일 생성

« 요구사항 테이블 작성 형태 »

요구사항 ID		요구사항 명		
RF-FILE-CREATE-AFLOW-2		다양한 파일 접근 모드로 파일 생성		
	유형	영역	흐름 분류	중요도
	기능	파일 관리	대체 흐름	상
	설명	정상적인 파일 생성에서 대체 흐름으로 파일 접근 모드 선택이 있다. 파일 접근 모드에 따라 접근할 수 있는 그룹은 소유자, 특정 그룹, 그 외 사람들로 이루어졌다. 파일에 대한 접근 권한은 전자 제품이 사용하는 운영 시스템과 연계 되어 있다. 만약 운영 시스템에서 이 기능을 제공하지 않을 경우에는 이 기능은 비활성화 된다.		
상	선처리 조건	RF-FILE-CREATE-BFLOW와 동일함		
세	후처리 조건	RF-FILE-CREATE-BFLOW와 동일함		
내				
역				
	1	어플리케이션 개발자가 정해진 규칙에 따라 파일 이름을 생성하고 파일 접근 모드를 생성한다.		
	2	어플리케이션 개발자가 파일 생성 함수를 호출한다.		
	3	파일 시스템은 어플리케이션 개발자에게 파일 접근 모드에 알맞은 파일 디스크립터를 준다.		
	제약사항	RF-FILE-CREATE-BFLOW와 동일함		
	기타	운영 시스템에서 파일 접근 모드를 지원하지 않은 경우에는 이 기능은 비활성화 된다.		
	요구사항 변경 이력			

« Redmine Tool 관리 형태 »

기능요구사항 #7

기능요구사항 #5: [RF-FILE-CREATE-BFLOW] 정상적인 파일 생성
[RF-FILE-CREATE-AFLOW-2] 다양한 파일 접근 모드로 파일 생성

User Name Admin@(기) 일본 미하 전래 추가기합.

상태:	생성	소요 시간:	-
우선 순위:	상	개발자:	개발자2 조
담당자:	-	제약사항:	RF-FILE-CREATE-BFLOW와 동일함
별명:	-	기타:	운영 시스템에서 파일 접근 모드를 지원하지 않은 경우에는 이 기능은 비활성화 된다.
모표 버전:	-		
영역:	파일 관리		
선처리 조건:	RF-FILE-CREATE-BFLOW와 동일함		
후처리 조건:	RF-FILE-CREATE-BFLOW와 동일함		
설명	<p>정상적인 파일 생성에서 대체 흐름으로 파일 접근 모드 선택이 있다. 파일 접근 모드에 따라 접근할 수 있는 그룹은 소유자, 특정 그룹, 그 외 사람들로 이루어졌다. 만약 운영 시스템에서 이 기능을 제공하지 않을 경우에는 이 기능은 비활성화 된다.</p> <ol style="list-style-type: none"> 어플리케이션 개발자가 정해진 규칙에 따라 파일 접근 모드를 생성하고 파일 접근 모드를 생성한다. 어플리케이션 개발자가 파일 생성 함수를 호출한다. 파일 시스템은 어플리케이션 개발자에게 파일 접근 모드에 알맞은 파일 디스크립터를 준다. 		

Part II 문서 작성 관점의 요구사항 작성 사례

Part II은 실제 시스템에서 요구사항을 개발 할 때 사용되는 형식으로 사례를 정리하였다. Part I에서 기술 된 것과 같이 요구공학은 크게 요구사항 개발과 요구사항 관리 영역으로 나누어져 있다. 이 Part II에서는 요구공학 내에서 요구사항 관리에 개발에 초점이 맞추어 정리된 부분이다.

Part II는 있는 요구사항 사례를 볼 때, 두 가지 관점에 대하여 고려해야 한다. 첫째는 어떤 이해관계자를 위한 요구사항을 만든 것인가와 둘째는 요구사항을 도출하는데 있어 어떤 방법론을 사용할 것인 가이다. 어떤 이해관계자를 위한 요구사항을 만들 것인가에 대해서는 요구사항이 비즈니스 요구사항인지, 사용자 요구사항인지 아니면 시스템 요구사항인지를 구분하게 한다. 그리고 요구사항을 도출하기 위하여 어떤 방법론을 사용에 대한 부분은 이 사례에서는 Use Case Modeling, QAW 방법론을 사용하고 있다.

요구사항을 이해하는 관계자의 특성에 따른 요구사항은 비즈니스 요구사항, 사용자 요구사항 그리고 시스템 요구사항이 있다. 비즈니스 요구사항은 시스템의 요구하는 이해관계자나 조직이 가지고 있는 비전이나 범위를 기술하는 것으로 주고 경영진, 기획 또는 마케팅 부서에서 원하는 요구사항들이 정리 되어 있다. 그리고 사용자 요구사항은 시스템이 제공하는 서비스와 운영 제약사항을 자연어와 다이어그램으로 기술한 것으로 모든 이해관계자를 위하여 작성된 것이다. 마지막으로 시스템 요구사항은 시스템의 기능과 서비스와 운영 제약 사항을 구체적으로 설정한 체계적인 부분을 말하면 주로 개발자를 대상으로 작성한 것이다. 그래서 시스템 요구사항은 도메인의 언어를 사용하여 작성하는 경우가 많다.

요구사항을 도출하는데 여러 가지 방법론이 존재하다. 여기 사례에서 사용한 방법론은 Use Case Modeling, QAW 방법론을 사용하였다. Use Case Modeling은 Actor를 중심으로 Use Case를 도출하여 요구사항을 추출하는 방법으로 주로 사용자가 명확한 시스템에서 사용한다. QAW는 아키텍처를 작성할 때, 필요한 품질 속성에 대하여 정리한 방법론으로 여기에서는 비기능 요구사항 중에 품질 속성의 요소를 도출하기 위하여

사용하였다. 이와 같이 요구사항을 도출하기 위해서는 각각의 도메인의 특성이나 사항에 따라 요구사항 도출 방법론을 사용해야 한다.

여기서도 Part I에서 사용한 예제인 파일 시스템의 요구사항을 사용하여 정리하였다. Part I과 Part II의 사례의 통일성을 주어 사례를 이해하는데 효율성을 높이기 위함이다.

1. 과제 배경

1.1 비즈니스 목표

IT기술의 발전으로 인하여 모든 전자제품의 융합이 이루어지고 있다. 이로 인하여 전자제품마다 컴퓨팅을 할 수 있는 능력이 생겨날 뿐만 아니라, 대용량의 저장시스템이 필요하게 되었다. 이에 어떤 전자 제품에서도 사용할 수 있는 파일 시스템이 이번 과제의 목적이다. 이번 과제에서 만들고자 하는 파일 시스템은 다양한 전자 제품에 대하여 각각 전자 제품의 특성에 맞추어 품질의 특정을 변경하여 적용하려고 할 뿐만 아니라 또한 전자 제품이 가지고 있는 컴퓨팅 능력에 맞추어 변경 가능한 파일 시스템을 만드는 것을 목적으로 하고 있다.

1.2 범위

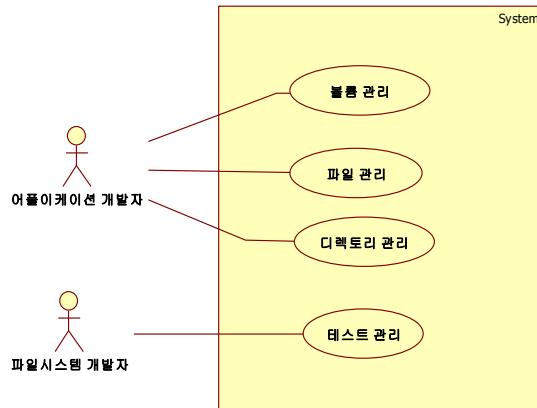
이번 과제는 스마트폰, 고화질 TV 그리고 고성능 노트북을 대상으로 하는 저장매체에 들어가는 파일 시스템이다. 이 파일 시스템은 현재는 위 3개의 제품군을 대상으로 개발을 진행하지만 지속적으로 적용할 수 있는 전자 제품군을 확대할 것이다. 그리고 현재 이번 과제의 개발 기간은 1년으로 진행될 예정이다. 이와 같은 범위와 일정으로 진행되는 파일 시스템은 “A 파일 시스템”이라고 호칭한다.

1.3 용어 정의

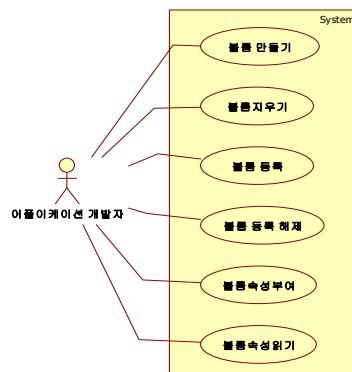
용어	설명
A 파일 시스템	이번 과제에서 개발하고자 하는 파일 시스템으로 모든 전자제품에서 사용할 수 있는 범용 파일 시스템
API (Application Program Interface)	애플리케이션을 개발하기 위하여 만들어진 인터페이스
POSIX (Portable Operating System Interface)	서로 다른 UNIX OS의 공통 API를 정리하여 이식성이 높은 유닉스 응용 프로그램을 개발하기 위한 목적으로 IEEE가 책정한 애플리케이션 인터페이스 규격
볼륨	사용자가 집적 접근하는 사용할 수 있는 논리적인 공간

2. 개요

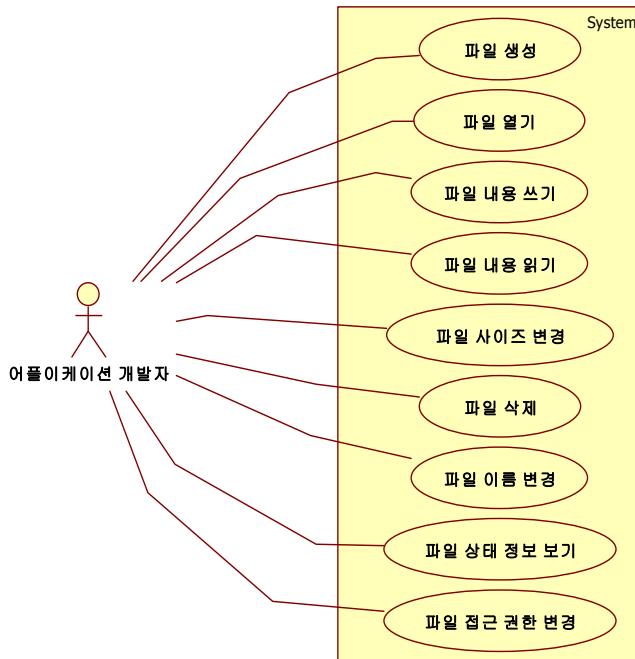
2.1 Use Case 다이어그램



A파일 시스템의 액터는 어플리케이션 개발자와 파일 시스템 개발자이다. 어플리케이션 개발자는 이 파일 시스템을 사용하여 각각의 전자제품에 필요한 어플리케이션을 개발하는 사람으로 실제 파일 시스템의 기능을 사용한다. 그리고 파일 시스템 개발자는 파일 시스템을 개발하면서 파일 시스템의 품질을 높이기 위하여 어플리케이션 개발자가 사용하는 기능 외 추가적인 기능을 사용하는 액터로 보면 된다. 그리고 A파일 시스템에는 크게 4개의 Use Case가 있다. 볼륨 관리, 파일 관리, 디렉토리 관리 그리고 테스트 관리가 있으면 각각의 Use Case는 아래의 그림과 같이 상세한 각각 상세한 Use Case로 나누어 질 수 있다.



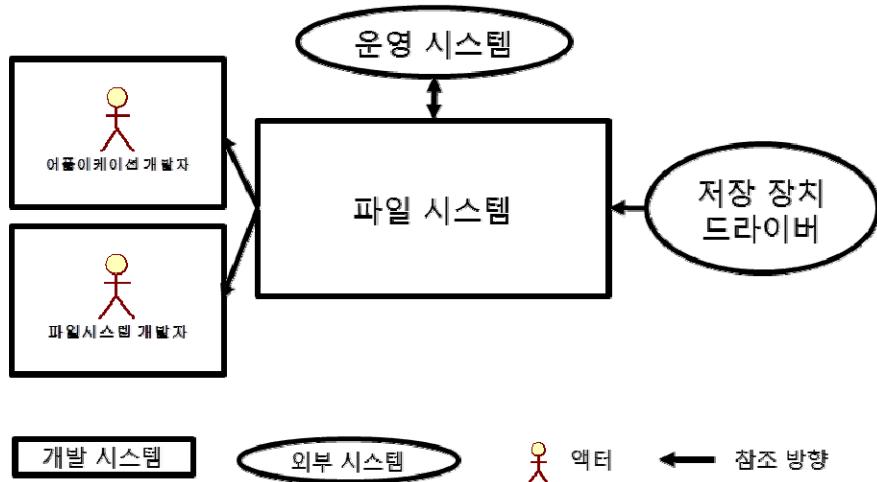
볼륨 관리의 상세 Use Case는 앞에 그림과 같다. 볼륨 만들기, 볼륨 등록, 볼륨 등록 해제, 볼륨 지우기, 볼륨 속성 부여 그리고 볼륨 속성 읽기이다. 볼륨 만들기는 저장장치에서 저장할 수 있는 형태의 공간을 만드는 것이다. 볼륨 등록은 시스템에 사용자가 사용할 수 있는 논리적 공간을 등록하여 사용자가 저장장치의 논리적인 공간을 볼 수 있도록 하는 기능이다. 볼륨 등록 해제는 등록된 볼륨을 해지하는 기능이다. 볼륨 지우기는 만들어진 저장 형태의 공간을 삭제하고 저장장치가 관리하는 형태로 공간을 반납하는 것이다. 그리고 볼륨 속성 부여는 사용자가 볼륨을 사용할 때, 읽기 전용인지, 쓰기 전용인지, 읽기 쓰기 모드인지를 설정하는 기능이다. 볼륨 속성 읽기는 볼륨이 현 상태의 정보를 읽어 사용자가 볼륨의 속성을 알 수 있게 해 주는 기능이다.



파일 관리의 상세 Use Case는 앞의 그림과 같다. 파일 생성, 파일 열기, 파일 내용 쓰기, 파일 내용 읽기, 파일 사이즈 변경, 파일 삭제, 파일 이름 변경, 파일 상태 정보 보기 그리고 파일 접근 권한 변경 등이 있다. 파일 생성은 저장 매체의 정보 관리의 기본 단위인 파일을 생성하는 기능이다. 파일 열기는 생성된 파일을 열어 파일 내용을 접근할 수 있는 권한이다. 파일 내용 쓰기는 생성된 파일 내에서 저장하고 싶은 정보

를 쓸 수 있는 기능이다. 파일 내용 읽기는 파일에 저장 된 내용을 읽는 기능이다. 파일 사이즈 변경은 생성된 파일에 대하여 내가 원하는 크기로 파일 크기를 변경하는 기능이다. 파일 삭제는 생성된 파일을 저장 매체에서 지우는 기능이다. 파일 이름 변경은 생성된 파일의 이름을 변경하는 기능이다. 파일 상태 정보 보기는 생성된 파일의 이름, 크기, 생성 날짜, 접근 권한, 변경 날짜 등에 대한 정보를 볼 수 있는 기능이다. 마지막으로 파일 접근 권한 변경은 파일을 접속할 수 있는 권한을 변경할 수 있는 권한이다.

2.2 시스템 Context 다이어그램



A 파일 시스템은 다양한 전자 제품에 돌아가는 시스템이기 때문에 전자 제품이 가지고 있는 운영 시스템을 사용해야 한다. 그리고 다양한 저장 장치를 받아 드릴 수 있도록 다양한 저장 장치의 인터페이스를 수용해야 한다. 그리고 어플리케이션 개발자와 파일 시스템 개발자가 쉽게 개발을 할 수 있도록 표준화된 API를 사용해야 한다.

3. 외부 인터페이스

3.1 사용자 인터페이스

사용자는 어플리케이션 개발자와 파일시스템 개발자 두 명의 액터가 존재함

어플리케이션 개발자는 POSIX규약을 따라 어플리케이션을 개발하여도 무리가 없어야 함

POSIX 규약 내 파일 시스템의 부분에 대하여 다음 문서의 내용을 따름

- 1003.1-2001 – IEEE Standard for IEEE Information Technology – Portable Operating System Interface (POSIX(R))
- 1003.1-2008 – IEEE Standard for Information Technology – Portable Operating System Interface (POSIX(R))
- 1003.1-2001/Cor 1-2002 – IEEE Standard for IEEE Std 1003.1-2001/Cor 1 2002 Standard for Information Technology -- Portable Operating System Interface (POSIX(R)) -- Technical Corrigendum Number 1
- 1003.1-2001/Cor 2-2004 – IEEE Standard for Information Technology – Portable Operating System Interface (POSIX) – Technical Corrigendum Number 2

파일시스템 개발자는 파일시스템의 특성상 빠른 테스트 및 디버그를 수행할 수 개발자를 위한 API를 제공하여야 함

3.2 하드웨어 인터페이스

- 저장 장치 드라이버에서 제공하는 일반적인 Interface를 따름
- 저장 장치 특성에 맞는 특별한 기능을 사용할 수 있도록 특별한 Interface에 대해서도 사용할 수 있도록 하여야 함

3.3 소프트웨어 인터페이스

- 전자 제품이 가진 운영 시스템을 사용하여도 문제가 되지 않도록 구현되어야 함

4. 제약 사항

- 스마트폰, 고화질 TV 그리고 고성능 노트북을 대상으로 하는 저장매체에 들어가는 파일 시스템을 구현함
- 위에서 정의한 범위는 1년(XXXX.XX.XX ~ XXXX.XX.XX)에 구현 예정임
- 이 이후에 다른 제품군에 대한 내용이 추가 될 수 있도록 할 것
- 기존 전자 장치에서 구현한 어플리케이션이 사용하는데 무리가 없도록 구현이 되어야 함

5. 요구사항 목록

분류	항목	ID	설명
기능	볼륨	RF-VOL-CREATE	볼륨 만들기
		RF-VOL-DELETE	볼륨 지우기
		RF-VOL-MOUNT	볼륨 등록
		RF-VOL-UNMOUNT	볼륨 등록 해지
		RF-VOL-GETSTATE	볼륨 속성 읽기
		RF-VOL-SETSTATE	볼륨 속성 부여
	파일	RF-FILE-CREATE	파일 생성
		RF-FILE-OPEN	파일 열기
		RF-FILE-READ	파일 내용 읽기
		RF-FILE-WRITE	파일 내용 쓰기
		RF-FILE-TRUNCATE	파일 사이즈 변경
		RF-FILE-DELETE	파일 삭제
		RF-FILE-NAME	파일 이름 변경

분류	항목	ID	설명
디렉터리	RF-FILE-GETSTATE	파일 상태 정보 보기	
		RF-FILE-CHMOD	파일 접근 권한 변경
	RF-DIR-OPEN RF-DIR-CHDIR RF-DIR-MARK RF-DIR-UNLINK RF-DIR-RMDIR RF-DIR-CREATE	RF-DIR-OPEN	디렉터리 열기
		RF-DIR-CHDIR	디렉터리 변경
		RF-DIR-MARK	디렉터리 접근 권한 변경
		RF-DIR-UNLINK	디렉터리 요소 삭제
		RF-DIR-RMDIR	디렉터리 삭제
		RF-DIR-CREATE	디렉터리 생성
비기능	사용성	RNF-USER-POSIX	POSIX 규약을 준수
	성능	RNF-PER-WRITEFILE	정해진 시간 내 파일 내용을 쓰기
		RNF-PER-READREAD	정해진 시간 내 파일 내용을 읽기
		RNF-PER-CREATEDIR	정해진 시간 내 디렉터리를 만들기
	가용성	RNF-AVA-LIFE	저장 장치의 수명이 다할 때까지 에러 없이 동작
	변경용이성	RNF-MAIN-ANYSTORAGE	어떤 저장 장치도 이 파일 시스템을 사용 가능
		RNF-MAIN-ANYOS	어떤 OS도 이 파일 시스템을 사용 가능
	확장성	RNF-PER-INFINITEFILE	저장 장치의 용량이 될 때까지 무한하게 파일을 생성
		RNF-PER-INFINITEFDIR	저장 장치의 용량이 될 때까지 무한하게 파일을 생성

6. 기능적 요구 사항

6.1 볼륨 관리

6.1.1 볼륨 만들기

제목	볼륨 만들기	ID	RF-VOL-CREATE	우선순위	상
설명	저장 매체를 접근하여 사용자가 사용할 수 있는 논리적 공간을 만드는 기능				
선조건	시스템에서 저장 장치 드라이버를 사용할 수 있어야 함				
후조건	없음				
액터		내용			
기본 흐름(B)	어플리케이션 개발자	생성하려고 하는 볼륨의 시작 위치를 정한다.			
	어플리케이션 개발자	생성하려고 하는 볼륨의 크기를 정한다.			
	어플리케이션 개발자	생성하려고 하는 볼륨의 모드를 선택한다.			
	어플리케이션 개발자	볼륨 생성함수를 호출한다.			
	어플리케이션 개발자	볼륨 구분자를 얻는다.			
선택 흐름(A)	-	-			
예외 흐름(E)	어플리케이션 개발자	생성하려고 하는 볼륨의 시작 위치가 잘못 입력한다.			
	어플리케이션 개발자	생성하려고 하는 볼륨의 크기를 잘못 입력한다.			
	어플리케이션 개발자	생성하려고 하는 볼륨의 모드를 잘못 입력한다.			
시나리오	1	어플리케이션 개발자가 생성하려고 하는 볼륨의 시작 위치를 정한다.			
	E_1	생성하려고 하는 볼륨의 시작 위치를 잘못 입력한다.			
	2	어플리케이션 개발자가 생성하려고 하는 볼륨의 크기를 정한다.			
	E_2	생성하려고 하는 볼륨의 크기를 잘못 입력한다.			
	3	어플리케이션 개발자가 생성하려고 하는 볼륨의 모드를 선택한다.			
	E_3	생성하려고 하는 볼륨의 모드를 잘못 입력한다.			
	4	어플리케이션 개발자가 볼륨 생성함수를 호출한다.			
	5	파일 시스템은 어플리케이션 개발자에게 볼륨 구분자를 준다.			
	E_1	정의되지 않는 볼륨 구분자를 호출하고, Error 함수에 Error값을 리턴한다.			
	E_2	정의되지 않는 볼륨 구분자를 호출하고, Error 함수에 Error값을 리턴한다.			
	E_3	정의되지 않는 볼륨 구분자를 호출하고, Error 함수에 Error값을 리턴한다.			
기타	볼륨의 모드는 3가지가 있다. 읽기 전용 쓰기 전용 읽기/쓰기 모드				

6.1.2 볼륨 지우기

제목	볼륨 지우기	ID	RF-VOL-DELETE	우선순위	상
설명	생성된 볼륨을 삭제하고 사용자가 사용할 수 있는 논리적 공간을 저장 장치에 반납하는 기능				
선조건	볼륨을 정상적으로 생성할 수 있어야 함				
후조건	없음				
	액터	내용			
기본 흐름(B)	어플리케이션 개발자	삭제하고자 하는 볼륨의 구별자를 정한다.			
	어플리케이션 개발자	볼륨 삭제함수를 호출한다.			
	어플리케이션 개발자	볼륨 삭제 함수에서 성공 메시지를 준다.			
선택 흐름(A)	-	-			
예외 흐름(E)	어플리케이션 개발자	삭제하고자 하는 볼륨의 구별자를 잘못 입력한다.			
시나리오	1	어플리케이션 개발자가 삭제하고자 하는 볼륨의 구별자를 정한다.			
	E_1	삭제하고자 하는 볼륨의 구별자를 잘못 입력한다.			
	2	어플리케이션 개발자가 볼륨 삭제 함수를 호출한다.			
	3	파일 시스템은 어플리케이션 개발자에게 볼륨이 정상적으로 삭제 되었다고 하는 성공 메시지를 준다.			
	E_1	함수에서 '-1'값을 리턴하고, Error 함수에 Error값을 리턴한다.			
기타	-				

6.1.3 볼륨 등록

제목	볼륨 등록	ID	RF-VOL-MOUNT	우선순위	상			
설명	생성된 볼륨을 파일 시스템에 등록하는 기능							
선조건	파일 시스템에서 볼륨 생성과 볼륨 삭제가 정상적으로 동작할 수 있어야 함							
후조건	등록된 볼륨을 파일 시스템에서 해제함							
	액터		내용					
기본 흐름(B)	어플리케이션 개발자	등록하고자 하는 볼륨 등록자를 정한다.						
	어플리케이션 개발자	등록하고자 하는 볼륨과 연결할 디렉토리를 정한다.						
	어플리케이션 개발자	볼륨 등록 함수를 호출한다.						
	어플리케이션 개발자	볼륨 등록 성공 메시지를 얻는다.						
선택 흐름(A)	-	-						
예외 흐름(E)	어플리케이션 개발자	등록하고자 하는 볼륨의 구별자를 잘못 입력한다.						
	어플리케이션 개발자	연결할 디렉토리를 잘못 입력한다.						
시나리오	1	어플리케이션 개발자가 등록하고자 하는 볼륨의 구별자를 정한다.						
	E_1	E_1. 등록하고자 하는 볼륨의 구별자를 잘못 입력한다.						
	2	어플리케이션 개발자가 등록하고자 하는 볼륨과 연결할 디렉토리를 정한다.						
	E_2	E_2. 연결할 디렉토리를 잘못 입력한다.						
	3	어플리케이션 개발자가 볼륨 등록 함수를 호출한다.						
	4	파일 시스템은 어플리케이션 개발자에게 볼륨이 정상적으로 등록되었다고 하는 성공 메시지를 준다.						
	E_1	함수에서 '-1'값을 리턴하고, Error 함수에 Error값을 리턴한다.						
	E_2	함수에서 '-1'값을 리턴하고, Error 함수에 Error값을 리턴한다.						
	기타							

6.1.4 볼륨 등록 해지

제목	볼륨 등록 해지	ID	RF-VOL-UMOUNT	우선순위	상
설명	파일 시스템에 등록된 볼륨을 해지하는 기능				
선조건	파일 시스템이 저장 장치를 볼륨으로 등록할 수 있어야 함				
후조건	-				
	액터	내용			
기본 흐름(B)	어플리케이션 개발자	해지하고자 하는 볼륨의 디렉토리 명을 입력한다.			
	어플리케이션 개발자	볼륨 등록 해지 함수를 호출한다.			
	어플리케이션 개발자	볼륨 등록 해지 성공 메시지를 얻는다.			
선택 흐름(A)	-	-			
예외 흐름(E)	어플리케이션 개발자	볼륨 등록을 해지할 수 없는 디렉토리 명을 입력한다.			
시나리오	1	어플리케이션 개발자가 해지하고자 하는 볼륨의 디렉토리 명을 입력한다.			
	E_1	볼륨 등록을 해지할 수 없는 디렉토리 명을 입력한다.			
	2	어플리케이션 개발자가 볼륨 등록 해지 함수를 호출한다.			
	3	파일 시스템은 어플리케이션 개발자에게 볼륨이 정상적으로 해지 되었다고 하는 성공 메시지를 준다.			
	E_1	함수에서 '-1'값을 리턴하고, Error 함수에 Error값을 리턴한다.			
기타					

6.1.5 볼륨 속성 부여

제목	볼륨 속성 부여	ID	RF-VOL-SETSTATE	우선순위	중
설명	파일 시스템에 등록된 볼륨의 속성을 변경할 수 있는 기능				
선조건	파일 시스템이 저장 장치를 볼륨으로 등록할 수 있어야 함				
후조건	-				
	액터	내용			
기본 흐름(B)	어플리케이션 개발자	속성을 변경하고자 하는 볼륨에 해당되는 디렉토리 명을 입력한다.			
	어플리케이션 개발자	변경하고자 하는 속성을 입력한다.			
	어플리케이션 개발자	볼륨 속성 부여 함수를 호출한다.			
	어플리케이션 개발자	볼륨 속성 변경에 대한 성공 메시지를 얻는다.			
선택 흐름(A)	-	-			
예외 흐름(E)	어플리케이션 개발자	볼륨 속성을 변경할 수 없는 디렉토리 명을 입력한다.			
	어플리케이션 개발자	볼륨 속성에 대한 모드를 잘못 입력한다.			
시나리오	1	어플리케이션 개발자가 속성을 변경하고자 하는 볼륨에 해당되는 디렉토리 명을 입력한다.			
	E_1	볼륨 속성을 변경할 수 없는 디렉토리 명을 입력한다.			
	2	어플리케이션 개발자가 변경하고자 하는 속성을 입력한다.			
	E_2	볼륨 속성에 대한 모드를 잘못 입력한다.			
	3	어플리케이션 개발자가 볼륨 속성 부여 함수를 호출한다.			
	4	파일 시스템은 어플리케이션 개발자에게 볼륨의 속성이 정상적으로 변경되었다는 성공 메시지를 얻는다.			
	E_1	함수에서 '-1'값을 리턴하고, Error 함수에 Error값을 리턴한다.			
	E_2	함수에서 '-1'값을 리턴하고, Error 함수에 Error값을 리턴한다.			
기타	볼륨 속성 부여 모드 읽기 전용 : R 쓰기 전용 : W 읽기/쓰기 모드 : RW				

6.1.6 볼륨 속성 읽기

제목	볼륨 속성 읽기	ID	RF-VOL-GETSTATE	우선순위	종
설명	파일 시스템에 등록된 볼륨의 속성을 읽는 기능				
선조건	파일 시스템이 저장 장치를 볼륨으로 등록할 수 있어야 함				
후조건	-				
액터		내용			
기본 흐름(B)	어플리케이션 개발자	볼륨의 속성을 읽을 볼륨의 디렉토리 명을 입력한다.			
	어플리케이션 개발자	볼륨 속성 읽기 함수를 호출한다.			
	어플리케이션 개발자	볼륨 속성 읽기 메시지를 얻는다.			
선택 흐름(A)	-	-			
예외 흐름(E)	어플리케이션 개발자	볼륨 속성을 읽을 수 있는 볼륨의 디렉토리 명을 잘못 입력한다.			
시나리오	1	어플리케이션 개발자가 볼륨 속성을 읽을 수 있는 볼륨의 디렉토리 명을 입력한다.			
	E_1	볼륨 속성을 읽을 수 있는 볼륨의 디렉토리 명을 잘못 입력한다.			
	2	어플리케이션 개발자가 볼륨 속성 읽기 함수를 호출한다.			
	3	파일 시스템은 어플리케이션 개발자에게 볼륨의 정보를 읽을 수 있는 위치 정보를 가르쳐 준다.			
	E_1	함수에서 '-1'값을 리턴하고, Error 함수에 Error값을 리턴한다.			
기타	볼륨 정보 리스트 시작 위치 볼륨의 크기 볼륨의 파일시스템 종류 볼륨의 접근 모드 볼륨과 접속하고 있는 디렉토리 명				

6.2 파일 관리

6.2.1 파일 생성

제목	파일 생성	ID	RF-FILE-CREATE	우선순위	상
설명	저장 매체의 정보 관리의 기본 단위인 파일을 생성하는 기능				
선조건	시스템의 저장 장치가 정상적으로 등록이 되어야 함				
후조건	파일을 조절하기 위하여 만든 파일 디스크립터를 반납함				
		액터	내용		
기본 흐름(B)	어플리케이션 개발자	파일이름을 생성한다.			
	어플리케이션 개발자	파일생성 함수를 호출한다.			
	파일 시스템	파일 디스크립터를 얻는다.			
선택 흐름(A)	어플리케이션 개발자	파일 오픈 모드를 선택한다.			
	어플리케이션 개발자	파일 접근 모드를 선택한다.			
예외 흐름(E)	어플리케이션 개발자	주어진 규칙에 따라 파일 이름을 생성하지 않는다.			
	어플리케이션 개발자	주어진 규칙에 따라 파일 오픈 모드를 기입하지 않는다.			
	어플리케이션 개발자	주어진 규칙에 따라 파일 접근 모드를 기입하지 않는다.			
시나 리오	1	어플리케이션 개발자가 파일 이름을 생성한다.			
	A_1	오픈 모드를 선택한다.			
	A_2	파일 접근 모드를 선택한다.			
	E_1	주어진 규칙에 따라 파일 이름을 생성하지 않았다.			
	E_2	주어진 규칙에 따라 파일 오픈 모드를 기입하지 않았다.			
	E_3	주어진 규칙에 따라 파일 접근 모드를 기입하지 않았다.			
	2	어플리케이션 개발자가 파일 생성 함수를 호출한다.			
	3	파일 시스템은 어플리케이션 개발자에게 파일 디스크립터를 준다.			
	A_1	오픈 모드에 맞는 파일 디스크립터를 준다.			
	A_2	파일 접근 모드에 맞는 파일 디스크립터를 준다.			
	E_1	정의되지 않는 파일 디스크립터가 호출되고 Error함수에서 Error값을 리턴한다.			
	E_2	정의되지 않는 파일 디스크립터가 호출되고 Error함수에서 Error값을 리턴한다.			
	E_3	정의되지 않는 파일 디스크립터가 호출되고 Error함수에서 Error값을 리턴한다.			
기타	파일 이름 생성 규칙				

제목	파일 생성	ID	RF-FILE-CREATE	우선순위	상
	<p>파일 이름의 최대 길이는 255 특수 문자 “*,!” 사용불가 대소문자를 구별함</p> <p><u>파일 오픈 모드</u> APPEN : 쓰기전용 모드로 파일을 엽니다. 파일을 열때 파일이 없다면 생성합니다 쓰기 전용 모드. 파일을 열 때 파일이 없다면 생성함 READ : 읽기전용 모드. 파일이 반드시 있어야 함 UPDATE : 읽기/쓰기 모드. 파일을 열 때 파일이 없다면 생성함 WRITE : 쓰기전용 모드. 파일이 없다고 생성함. 파일이 있다면 덮어쓰기</p> <p><u>파일 접근 모드</u> 파일 접근은 3개의 그룹으로 이루어짐 (소유자, 특정 그룹, 그외 사람들) 각각 읽고, 쓰고, 실행하기의 허가를 할 수 있음</p>				

6.2.2 파일 열기

제목	파일 열기	ID	RF-FILE-OPEN	우선순위	상
설명	생성된 파일을 열어 파일을 조절할 수 있는 디스크립터를 얻는 기능				
선조건	시스템의 저장 장치가 정상적으로 등록이 되어야 함				
후조건	파일을 조절하기 위하여 만든 파일 디스크립터를 반납함				
	액터	내용			
기본 흐름(B)	어플리케이션 개발자	오픈할 파일이름을 생성한다.			
	어플리케이션 개발자	파일 오픈 함수를 호출한다.			
	어플리케이션 개발자	파일 디스크립터를 얻는다.			
선택 흐름(A)	어플리케이션 개발자	파일 오픈 모드를 선택한다.			
	어플리케이션 개발자	파일 접근 모드를 선택한다.			
예외 흐름(E)	어플리케이션 개발자	오픈할 파일이 존재하지 않는다.			
	어플리케이션 개발자	주어진 규칙에 따라 파일 오픈 모드를 기입하지 않는다.			
	어플리케이션 개발자	주어진 규칙에 따라 파일 접근 모드를 기입하지 않는다.			
시나리오	1	어플리케이션 개발자가 오픈할 파일이름을 생성한다.			
	A_1	오픈 모드를 선택한다.			
	A_2	파일 접근 모드를 선택한다.			
	E_1	존재하지 않는 파일 이름을 생성한다.			
	E_2	주어진 규칙에 따라 파일 오픈 모드를 기입하지 않았다.			
	E_3	주어진 규칙에 따라 파일 접근 모드를 기입하지 않았다.			
	2	어플리케이션 개발자가 파일 오픈 함수를 호출한다.			
	3	파일 시스템은 어플리케이션 개발자에게 오픈할 파일 디스크립터를 준다.			
	A_1	오픈 모드에 맞는 파일 디스크립터를 준다.			
	A_2	파일 접근 모드에 맞는 파일 디스크립터를 준다.			
	E_1	정의되지 않는 파일 디스크립터가 호출되고 Error함수에서 Error값을 리턴한다.			
	E_2	정의되지 않는 파일 디스크립터가 호출되고 Error 함수에서 Error 값을 리턴한다.			
기타	파일 오픈 모드와 파일 접근 모드는 “RF-VOL-CREATE” 내용과 동일하게 적용됨				

6.2.3 파일 내용 쓰기

제목	파일 내용 쓰기	ID	RF-FILE-WRITE	우선순위	상
설명	파일에 원하는 Data를 쓸 수 있는 기능				
선조건	시스템의 장치가 정상적으로 등록이 되어야 함 정상적인 파일이 생성되어야 함				
후조건	파일을 조절하기 위하여 만든 파일 디스크립터를 반납함				
	액터		내용		
기본 흐름(B)	어플리케이션 개발자		파일을 오픈한다.		
	어플리케이션 개발자		오픈한 파일의 디스크립터를 얻는다.		
	어플리케이션 개발자		파일에 쓸 Data를 정한다.		
	어플리케이션 개발자		파일에 쓸 Data의 크기를 정한다.		
	어플리케이션 개발자		파일에 Data를 쓸 있는 함수를 호출한다.		
	어플리케이션 개발자		파일에 쓰여진 Data의 길이를 얻는다.		
선택 흐름(A)	어플리케이션 개발자		-		
예외 흐름(E)	어플리케이션 개발자		존재하지 않는 파일 디스크립터를 사용한다.		
	어플리케이션 개발자		읽기 전용 모드의 파일 디스크립터를 사용한다.		
	어플리케이션 개발자		파일에 쓸 Data와 쓸 Data의 크기가 일치하지 않는다.		
시나리오	1		어플리케이션 개발자는 파일을 오픈한다.		
	E_2		읽기 전용 모드로 파일을 오픈 한다.		
	2		어플리케이션 개발자는 오픈한 파일의 디스크립터를 얻는다.		
	E_1		존재하지 않는 파일 디스크립터를 사용한다.		
	3		어플리케이션 개발자는 파일에 쓸 Data를 정한다.		
	4		어플리케이션 개발자는 파일에 쓸 Data의 크기를 정한다.		
	E_		파일에 쓸 Data와 쓸 Data의 길이가 일치하지 않는다.		
	5		어플리케이션 개발자는 파일에 Data를 쓸 수 있는 함수를 호출한다.		
	6		파일 시스템은 어플리케이션 개발자는 파일에 쓰여진 Data의 길이를 준다. 그리고 파일에 쓰여진 Data는 파일을 읽었을 때 일치해야 한다.		
	E_1		'0'을 리턴하고 Error함수에서 Error값을 리턴한다.		
	E_2		'0'을 리턴하고 Error함수에서 Error값을 리턴한다.		
	E_3		쓰 Data의 길이와 동일한 값을 리턴한다. (기타 참조)		
기타	파일에 쓸 실제 Data의 길이와 쓸 Data의 크기가 일치하지 않을 때 파일에 쓸 실제 Data의 길이 > 쓸 Data의 크기 : 쓸 Data의 크기만큼 Data를 파일에 기록한다. 파일에 쓸 실제 Data의 길이 < 쓸 Data의 크기 : 파일에 쓸 실제 Data를 기입하고 쓸 Data의 크기와 파일에 쓸 실제 Data의 길이 차에 대한 나머지 부분은 0으로 기입한다.				

6.2.4 파일 내용 읽기

제목	파일 내용 읽기	ID	RF-FILE-READ	우선순위	상
설명	파일에 있는 Data를 읽는 기능				
선조건	시스템의 장치가 정상적으로 등록이 되어야 함 정상적인 파일이 생성하고 파일에 내용을 쓸 수 있어야 함				
후조건	파일을 조절하기 위하여 만든 파일 디스크립터를 반납함				
	액터		내용		
기본 흐름(B)	어플리케이션 개발자	Data가 들어 있는 파일을 오픈한다.			
	어플리케이션 개발자	오픈한 파일의 디스크립터를 얻는다.			
	어플리케이션 개발자	파일 Data를 저장할 수 있는 공간을 만든다.			
	어플리케이션 개발자	파일 Data를 읽을 사이즈를 입력한다.			
	어플리케이션 개발자	파일 읽기 함수를 호출한다.			
	어플리케이션 개발자	파일에서 읽은 Data의 길이를 얻는다.			
선택 흐름(A)	어플리케이션 개발자	-			
예외 흐름(E)	어플리케이션 개발자	존재하지 않는 파일 디스크립터를 사용한다.			
	어플리케이션 개발자	쓰기 전용 모드의 파일 디스크립터를 사용한다.			
	어플리케이션 개발자	파일에 읽을 Data를 저장할 공간 길이와 실제 읽을 사이즈가 일치하지 않는다			
시나리오	1	어플리케이션 개발자는 Data가 들어 있는 파일을 오픈한다.			
	E_2	쓰기 전용 모드의 파일 읽기 전용 모드로 파일을 오픈 한다.			
	2	어플리케이션 개발자는 오픈한 파일의 디스크립터를 얻는다.			
	E_1	존재하지 않는 파일 디스크립터를 사용한다.			
	3	어플리케이션 개발자는 파일 Data를 저장할 수 있는 공간을 만든다.			
	4	어플리케이션 개발자는 파일 Data를 읽을 사이즈를 입력한다.			
	E_3	파일에 읽을 Data를 저장할 공간 길이와 실제 읽을 사이즈가 일치하지 않는다.			
	5	어플리케이션 개발자는 파일에 Data를 읽을 수 있는 함수를 호출한다.			
	6	파일 시스템은 어플리케이션 개발자는 파일에 읽혀진 Data의 길이를 준다. 그리고 파일에 쓰여진 Data는 파일을 읽었을 때 일치해야 한다.			
	E_1	'0'을 리턴하고 Error함수에서 Error값을 리턴한다.			
	E_2	'0'을 리턴하고 Error함수에서 Error값을 리턴한다.			
	E_3	읽혀진 Data의 길이와 동일한 값을 리턴한다. (기타 참조)			
기타	파일에 읽을 Data를 저장할 공간 길이와 실제 읽을 사이즈가 일치하지 않을 때, 파일에 읽을 Data를 저장할 공간 길이 > 실제 읽을 사이즈 : 실제 읽은 사이즈 크기만큼 Data 읽고 Data를 저장할 공간의 길이 만큼 저장한다. 파일에 읽을 Data를 저장할 공간 길이 > 실제 읽을 사이즈 : 실제 읽은 사이즈 크기만큼 Data 읽고 Data를 저장할 공간의 길이 만큼 저장한다.				

7. 비기능적 요구 사항

7.1 사용성

7.1.1 POSIX 규약을 준수

제목	POSIX 규약 준수	ID	RNF-USER-POSIX	우선순위	상
구분	자극원	POSIX 규약을 알고 있는 어플리케이션 개발자			
	자극	어플리케이션을 개발할 때			
	대상	A 파일 시스템의 API			
	환경	어플리케이션을 구현하고 있는 단계			
	응답	어플리케이션 개발자가 어플리케이션을 만드는데 어려움이 없어야 함			
	응답 측정	A 파일 시스템의 API가 POSIX 규약에 일치하는지를 확인함			
시나리오	POSIX 규약을 알고 있는 어플리케이션 개발자가 A 파일 시스템의 API를 가지고 어플리케이션을 구현할 때, 어플리케이션 개발자는 어플리케이션을 만드는데 어려움이 없다. 그리고 이를 검증하기 위해서 A파일 시스템의 API가 POSIX 규약에 일치해야 한다.				
기타	A 파일 시스템의 아래의 POSIX 규약을 따라야 함 1003.1-2001 - IEEE Standard for IEEE Information Technology - Portable Operating System Interface (POSIX(R)) 1003.1-2008 - IEEE Standard for Information Technology - Portable Operating System Interface (POSIX(R)) 1003.1-2001/Cor 1-2002 - IEEE Standard for IEEE Std 1003.1-2001/Cor 1 2002 Standard for Information Technology -- Portable Operating System Interface (POSIX(R)) -- Technical Corrigendum Number 1 1003.1-2001/Cor 2-2004 - IEEE Standard for Information Technology - Portable Operating System Interface (POSIX) - Technical Corrigendum Number 2				

7.2 성능

7.2.1 정해진 시간 내 파일 내용을 쓰기

제목	정해진 시간 내 파일 내용을 쓰기		ID	RNF-PER-WRITEFILE	우선순위	상
구분	자극원	전자제품 사용자				
	자극	어플리케이션이 저장 장치에 Data를 쓰는 경우				
	대상	A 파일 시스템으로 구현한 어플리케이션				
	환경	정상적으로 동작할 때				
	응답	정해진 시간 내 파일에 Data가 쓰여져야 함				
	응답 측정	A Bench Mark Tool을 사용하여 B사 제품보다 10% 향상된 쓰기 속도가 나와야 함				
시나리오	전자제품을 사용하는 사용자가 전자제품에 있는 어플리케이션으로 Data를 저장할 때, 사용자가 정해진 시간 내까지 파일에 Data가 쓰여져야 한다. 이를 검증하기 위하여 A Bench Mark Tool을 사용하여 B사 제품보다 10% 향상된 쓰기 속도가 나와야 한다.					
기타	제품의 성능을 위하여 전자제품의 하드웨어를 개발하는 팀과 상의 후에 제품의 성능을 다시 조정 될 수 있음					

7.2.2 정해진 시간 내 파일 내용을 읽기

제목	정해진 시간 내 파일 내용을 읽기		ID	RNF-PER-READFILE	우선순위	상
구분	자극원	전자제품 사용자				
	자극	어플리케이션이 저장 장치에 Data를 읽을 경우				
	대상	A 파일 시스템으로 구현한 어플리케이션				
	환경	정상적으로 동작할 때				
	응답	정해진 시간 내 파일에 Data가 읽혀야 함				
	응답 측정	A Bench Mark Tool을 사용하여 B사 제품보다 10% 향상된 읽기 속도가 나와야 함				
시나리오	전자제품을 사용하는 사용자가 전자제품에 있는 어플리케이션으로 Data를 읽을 때, 사용자가 정해진 시간 내까지 파일에 Data가 읽혀야 한다. 이를 검증하기 위하여 A Bench Mark Tool을 사용하여 B사 제품보다 10% 향상된 읽기 속도가 나와야 한다.					
기타	제품의 성능을 위하여 전자제품의 하드웨어를 개발하는 팀과 상의 후에 제품의 성능을 다시 조정 될 수 있음					

7.2.3 정해진 시간 내 디렉터리를 만들기

제목	정해진 시간 내 파일 내용을 쓰기		ID	RNF-PER-CREATEDIR	우선순위	중
구분	자극원	전자제품 사용자				
	자극	A 파일 시스템으로 구현한 어플리케이션이 저장 장치에 디렉터리를 만드는 경우				
	대상	A 파일 시스템으로 구현한 어플리케이션				
	환경	정상적으로 동작할 때				
	응답	정해진 시간 내 디렉터리를 만들어야 함				
	응답 측정	A Bench Mark Tool을 사용하여 B사 제품보다 10% 향상된 디렉터리 만드는 속도가 나와야 함				
시나리오	전자제품을 사용하는 사용자가 전자제품에 있는 어플리케이션으로 디렉터리를 만들 때, 사용자가 정해진 시간 내까지 디렉터리를 만들어야 한다. 이를 검증하기 위하여 A Bench Mark Tool을 사용하여 B사 제품보다 10% 향상된 디렉터리 만드는 속도가 나와야 한다.					
기타	제품의 성능을 위하여 전자제품의 하드웨어를 개발하는 팀과 상의 후에 제품의 성능을 다시 조정 될 수 있음					

7.3 가용성

7.3.1 저장 장치의 수명이 다할 때까지 에러 없이 동작

제목	저장 장치의 수명이 다할 때까지 에러 없이 동작		ID	RNF-AVA-LIFE	우선순위	상
구분	자극원	보통의 전자 제품 사용자				
	자극	A 파일 시스템의 기능을 사용할 때				
	대상	A 파일 시스템을 사용하는 어플리케이션				
	환경	정상적으로				
	응답	전자 제품의 수명이 다할 때까지 사용하여도 A 파일 시스템이 붕괴되지 않아야 함				
	응답 측정	전자 제품을 5년 동안 사용하여도 A 파일 시스템이 붕괴되지 않아야 함				
시나리오	보통의 전자 제품 사용자가 A 파일 시스템의 기능을 정상적인 동작 환경에서 5년 정도 사용하여도 A 파일 시스템이 붕괴 되지 않아야 한다.					
기타						

7.4 변경용이성

7.4.1 어떤 저장 장치도 이 파일 시스템을 사용 가능

제목	어떤 저장 장치도 이 파일 시스템을 사용 가능	ID	RNF-MAIN-ANYSTORAGE	우선순위	상
구분	자극원	전자 제품에 A 파일 시스템 포팅하는 개발자			
	자극	새로운 전자 제품에 포팅 할 때			
	대상	A 파일 시스템			
	환경	전자 제품을 개발할 때			
	응답	새로운 전자 제품에 파일 시스템 포팅하는 개발자가 A 파일 시스템을 쉽게 포팅 할 수 있어야 함			
	응답 측정	새로운 전자 제품에 파일 시스템 포팅하는 개발자가 A 파일 시스템을 2일 안에 포팅 할 수 있어야 함			
시나리오	새로운 전자 제품에 A 파일 시스템을 포팅 할 경우 포팅하는 개발자가 포팅하는데 2일 안 에 포팅을 할 수 있어야 한다.				
기타	사람의 능력에서 문서 형태로 변경 가능한지에 대하여 체크할 것				

8. 요구사항 추적표

요구사항 ID	설명	시나리오			테스트 케이스 ID		
			Phone	TV	Com		
RF-VOL-CREATE-BFLOW	볼륨 만들기	1	어플리케이션 개발자가 생성하려고 하는 볼륨의 시작 위치를 정한다.				
		2	어플리케이션 개발자가 생성하려고 하는 볼륨의 크기를 정한다.				
		3	어플리케이션 개발자가 생성하려고 하는 볼륨의 모드를 선택한다.				
		4	어플리케이션 개발자가 볼륨 생성함수를 호출한다.				
		5	파일 시스템은 어플리케이션 개발자에게 볼륨 구분자를 준다.				
RF-VOL-CREATE-EFLOW-1	볼륨의 시작 위치를 잘못 입력하는 경우	1	어플리케이션 개발자가 생성하려고 하는 볼륨의 시작 위치를 잘못 정한다.				
		2	어플리케이션 개발자가 생성하려고 하는 볼륨의 크기를 정한다.				
		3	어플리케이션 개발자가 생성하려고 하는 볼륨의 모드를 선택한다.				
		4	어플리케이션 개발자가 볼륨 생성함수를 호출한다.				
		5	파일 시스템은 어플리케이션 개발자에게 정의되지 않는 볼륨 구분자를 호출하고, Error 함수에 Error값을 리턴한다.				
RF-VOL-CREATE-EFLOW-2	볼륨의 크기 값을 잘못 입력하는 경우	1	어플리케이션 개발자가 생성하려고 하는 볼륨의 시작 위치를 정한다.				
		2	어플리케이션 개발자가 생성하려고 하는 볼륨의 크기를 잘못 입력한다.				
		3	어플리케이션 개발자가 생성하려고 하는 볼륨의 모드를 선택한다.				
		4	어플리케이션 개발자가 볼륨 생성함수를 호출한다.				
		5	파일 시스템은 어플리케이션 개발자에게 정의되지 않는 볼륨 구분자를 호출하고, Error 함수에 Error값을 리턴한다.				
RF-VOL-CREATE-EFLOW-3	볼륨의 모드를 잘못 입력하는 경우	1	어플리케이션 개발자가 생성하려고 하는 볼륨의 시작 위치를 정한다.				
		2	어플리케이션 개발자가 생성하려고 하는 볼륨의 크기를 정한다.				
		3	어플리케이션 개발자가 생성하려고 하는 볼륨의 모드를 잘못 입력한다.				

요구사항 ID	설명	시나리오		테스트 케이스 ID		
		Phone	TV	Com		
		4	어플리케이션 개발자가 볼륨 생성함수를 호출한다.			
		5	파일 시스템은 어플리케이션 개발자에게 정의되지 않는 볼륨 구분자를 호출하고, Error 함수에 Error값을 리턴한다.			
RF-VOL-DELETE -BFLOW	볼륨 지우기	1	어플리케이션 개발자가 삭제하고자 하는 볼륨의 구별자를 정한다.			
		2	어플리케이션 개발자가 볼륨 삭제 함수를 호출한다.			
		3	파일 시스템은 어플리케이션 개발자에게 볼륨이 정상적으로 삭제 되었다고 하는 성공 메시지를 준다.			
RF-VOL-DELETE -EFLOW-1	볼륨을 지울 때, 볼륨의 구별자를 잘못 입력하는 경우	1	어플리케이션 개발자가 삭제하고자 하는 볼륨의 구별자를 잘못 입력한다.			
		2	어플리케이션 개발자가 볼륨 삭제 함수를 호출한다.			
		3	파일 시스템은 어플리케이션 개발자에게 함수에서 '-1'값을 리턴하고, Error 함수에 Error값을 리턴한다.			
RF-VOL-MOUNT -BFLOW	볼륨 등록	1	어플리케이션 개발자가 등록하고자 하는 볼륨의 구별자를 정한다.			
		2	어플리케이션 개발자가 등록하고자 하는 볼륨과 연결할 디렉토리를 정한다.			
		3	어플리케이션 개발자가 볼륨 등록 함수를 호출한다.			
		4	파일 시스템은 어플리케이션 개발자에게 볼륨이 정상적으로 등록 되었다고 하는 성공 메시지를 준다.			
RF-VOL-MOUNT -EFLOW-1	볼륨을 등록할 때, 볼륨의 구별자를 잘못 입력하는 경우	1	어플리케이션 개발자가 등록하고자 하는 볼륨의 구별자를 잘못 입력한다.			
		2	어플리케이션 개발자가 등록하고자 하는 볼륨과 연결할 디렉토리를 정한다.			
		3	어플리케이션 개발자가 볼륨 등록 함수를 호출한다.			
		4	파일 시스템은 어플리케이션 개발자에게 함수에서 '-1'값을 리턴하고, Error 함수에 Error값을 리턴한다			
RF-VOL-MOUNT -EFLOW-2	볼륨을 등록할 때, 볼륨의 연결 디렉터리를 잘못 입력하는 경우	1	어플리케이션 개발자가 등록하고자 하는 볼륨의 구별자를 정한다.			
		2	어플리케이션 개발자가 등록하고자 하는 볼륨과 연결할 디렉토리를 잘못 입력한다.			
		3	어플리케이션 개발자가 볼륨 등록 함수를 호출한다.			
		4	파일 시스템은 어플리케이션 개발자에게 함수에서 '-1'값을 리턴하고, Error 함수에 Error값을 리턴한다			

SW개발 품질관리 매뉴얼

요구사항 ID	설명	시나리오			테스트 케이스 ID		
		Phone	TV	Com			
RF-VOL-UNMO UNT-BFLOW	볼륨 등록 해 지	1	어플리케이션 개발자가 해지하고자 하는 볼륨의 디렉토리 명을 입력한다.				
		2	어플리케이션 개발자가 볼륨 등록 해지 함수를 호출한다.				
		3	파일 시스템은 어플리케이션 개발자에게 볼륨이 정상적으로 해지 되었다고 하는 성공 메시지를 준다.				
RF-VOL-UNMO UNT-EFLOW-1	볼륨 등록 해 지 할 때, 볼 륨의 연결 디 렉터리를 잘못 입력하는 경우	1	어플리케이션 개발자가 볼륨 등록을 해지할 수 없는 디렉토리 명을 입력한다.				
		2	어플리케이션 개발자가 볼륨 등록 해지 함수를 호출한다.				
		3	파일 시스템은 어플리케이션 개발자에게 볼륨이 정상적으로 해지 되었다고 하는 성공 메시지를 준다.				
RF-VOL- SETSTATE -BFLOW	볼륨 속성 부여	1	어플리케이션 개발자가 속성을 변경하고자 하는 볼륨에 해당되는 디렉토리 명을 입력한다.				
		2	어플리케이션 개발자가 변경하고자 하는 속성을 입력한다.				
		3	어플리케이션 개발자가 볼륨 속성 부여 함수를 호출한다.				
		4	파일 시스템은 어플리케이션 개발자에게 볼륨의 속성이 정상적으로 변경되었다는 성공 메시지를 얻는다.				
RF-VOL- SETSTATE -EFLOW-1	볼륨 속성을 부여할 때, 디 렉터리명이 잘 못 입력되는 경우	1	어플리케이션 개발자가 볼륨 속성을 변경할 수 없는 디렉토리 명을 입력한다.				
		2	어플리케이션 개발자가 변경하고자 하는 속성을 입력한다.				
		3	어플리케이션 개발자가 볼륨 속성 부여 함수를 호출한다.				
		4	파일 시스템은 어플리케이션 개발자에게 함수에서 '-1'값을 리턴하고, Error 함수에 Error값을 리턴한다.				
RF-VOL- SETSTATE -EFLOW-2	볼륨 속성을 부여할 때, 속 성부여 모드를 잘못 입력하는 경우	1	어플리케이션 개발자가 속성을 변경하고자 하는 볼륨에 해당되는 디렉토리 명을 입력한다.				
		2	어플리케이션 개발자가 볼륨 속성에 대한 모드를 잘못 입력한다.				
		3	어플리케이션 개발자가 볼륨 속성 부여 함수를 호출한다.				
		4	파일 시스템은 어플리케이션 개발자에게 함수에서 '-1'값을 리턴하고, Error 함수에 Error값을 리턴한다.				
RF-VOL-GETST ATE-BFLOW	볼륨 속성 읽기	1	어플리케이션 개발자가 볼륨 속성을 읽을 수 있는 볼륨의 디렉토리 명을 입력한다.				
		2	어플리케이션 개발자가 볼륨 속성 읽기 함수를 호출한다.				
		3	파일 시스템은 어플리케이션 개발자에게 볼륨의 정보를 읽을 수 있는 위치 정보를 가르쳐 준다.				

요구사항 ID	설명	시나리오			테스트 케이스 ID		
			Phone	TV	Com		
RF-VOL-SETSTA TE-EFLOW-1	볼륨 속성을 읽을 때, 디렉터리명이 잘못 입력되는 경우	1 어플리케이션 개발자가 볼륨 속성을 읽을 수 있는 볼륨의 디렉토리 명을 잘못 입력한다.					
		2 어플리케이션 개발자가 볼륨 속성 읽기 함수를 호출한다.					
		3 파일 시스템은 어플리케이션 개발자에게 함수에서 '-1'값을 리턴하고, Error 함수에 Error값을 리턴한다.					
RF-FILE-CREATE -BFLOW	파일 생성	1 어플리케이션 개발자가 파일 이름을 생성한다.					
		2 어플리케이션 개발자가 파일 생성 함수를 호출한다.					
		3 파일 시스템은 어플리케이션 개발자에게 파일 디스크립터를 준다.					
RF-FILE-CREATE -AFLOW-1	파일 오픈 모드를 선택하여 파일 생성	1 어플리케이션 개발자가 파일 이름을 생성한다. 그리고 오픈 모드를 선택한다.					
		2 어플리케이션 개발자가 파일 생성 함수를 호출한다.					
		3 파일 시스템은 어플리케이션 개발자에게 오픈 모드에 맞는 파일 디스크립터를 준다.					
RF-FILE-CREATE -AFLOW-2	파일 접근 모드를 선택하여 파일 생성	1 어플리케이션 개발자가 파일 이름을 생성한다. 그리고 파일 접근 모드를 선택한다.					
		2 어플리케이션 개발자가 파일 생성 함수를 호출한다.					
		3 파일 시스템은 어플리케이션 개발자에게 파일 접근 모드에 맞는 파일 디스크립터를 준다.					
RF-FILE-CREATE -EFLOW-1	주어진 규칙에 따라 파일 이름을 생성하지 않고 파일을 생성	1 어플리케이션 개발자가 주어진 규칙에 따라 파일 이름을 생성하지 않았다.					
		2 어플리케이션 개발자가 파일 생성 함수를 호출한다.					
		3 파일 시스템은 어플리케이션 개발자에게 정의되지 않는 파일 디스크립터가 호출되고 Error 함수에서 Error값을 리턴한다.					
RF-FILE-CREATE -EFLOW-2	주어진 규칙에 따라 파일 오픈 모드를 기입하지 않고 파일을 생성	1 어플리케이션 개발자가 파일 이름을 만든다. 그리고 주어진 규칙에 따라 파일 오픈 모드를 기입하지 않았다.					
		2 어플리케이션 개발자가 파일 생성 함수를 호출한다.					
		3 파일 시스템은 어플리케이션 개발자에게 정의되지 않는 파일 디스크립터가 호출되고 Error 함수에서 Error값을 리턴한다.					
RF-FILE-CREATE -EFLOW-3	주어진 규칙에 따라 파일 접근 모드를 기입하지 않고 파일을 생성	1 어플리케이션 개발자가 파일 이름을 생성한다. 그리고 주어진 규칙에 따라 파일 접근 모드를 기입하지 않았다.					
		2 어플리케이션 개발자가 파일 생성 함수를 호출한다.					
		3 파일 시스템은 어플리케이션 개발자에게 정의되지 않는 파일 디스크립터가 호출되고 Error 함수에서 Error값을 리턴한다.					

SW개발 품질관리 매뉴얼

요구사항 ID	설명	시나리오			테스트 케이스 ID		
			Phone	TV	Com		
RF-FILE-OPEN-B FLOW	파일 오픈	1 어플리케이션 개발자가 오픈할 파일이름을 생성한다.					
		2 어플리케이션 개발자가 파일 오픈 함수를 호출한다.					
		3 파일 시스템은 어플리케이션 개발자에게 오픈할 파일 디스크립터를 준다.					
RF-FILE-OPEN-A FLOW-1	파일 오픈 모드를 선택하여 파일 열기	1 어플리케이션 개발자가 오픈할 파일이름을 생성한다.					
		2 그리고 오픈 모드를 선택한다.					
		3 어플리케이션 개발자가 파일 오픈 함수를 호출한다.					
		4 파일 시스템은 어플리케이션 개발자에게 오픈 모드에 맞는 파일 디스크립터를 준다.					
RF-FILE-OPEN-A FLOW-2	파일 접근 모드를 선택하여 파일 열기	1 어플리케이션 개발자가 오픈할 파일이름을 생성한다.					
		2 그리고 파일 접근 모드를 선택한다.					
		3 어플리케이션 개발자가 파일 오픈 함수를 호출한다.					
		4 파일 시스템은 어플리케이션 개발자에게 파일 접근 모드에 맞는 파일 디스크립터를 준다.					
RF-FILE-OPEN-E FLOW-1	존재하지 않는 파일 열기	1 어플리케이션 개발자가 존재하지 않는 파일 이름을 생성한다.					
		2 어플리케이션 개발자가 파일 오픈 함수를 호출한다.					
		3 파일 시스템은 어플리케이션 개발자에게 정의되지 않는 파일 디스크립터가 호출되고 Error함수에서 Error값을 리턴한다.					
RF-FILE-OPEN-E FLOW-2	주어진 규칙에 따라 파일 오픈 모드를 기입하지 않고 파일을 열기	1 어플리케이션 개발자가 오픈할 파일이름을 생성한다.					
		2 그리고 주어진 규칙에 따라 파일 오픈 모드를 기입하지 않았다.					
		3 어플리케이션 개발자가 파일 오픈 함수를 호출한다.					
		4 파일 시스템은 어플리케이션 개발자에게 정의되지 않는 파일 디스크립터가 호출되고 Error 함수에서 Error값을 리턴한다.					
RF-FILE-OPEN-E FLOW-3	주어진 규칙에 따라 파일 접근 모드를 기입하지 않고 파일을 열기	1 어플리케이션 개발자가 오픈할 파일이름을 생성한다.					
		2 그리고 주어진 규칙에 따라 파일 접근 모드를 기입하지 않았다.					
		3 어플리케이션 개발자가 파일 오픈 함수를 호출한다.					
		4 파일 시스템은 어플리케이션 개발자에게 정의되지 않는 파일 디스크립터가 호출되고 Error 함수에서 Error값을 리턴한다.					
RF-FILE-WRITE-B FLOW	파일 내용 쓰기	1 어플리케이션 개발자는 파일을 오픈한다.					
		2 어플리케이션 개발자는 오픈한 파일의 디스크립터를 얻는다.					

요구사항 ID	설명	시나리오		테스트 케이스 ID		
		Phone	TV	Com		
RF-FILE-WRITE-EFLOW-1	존재하지 않는 파일 디스크립터로 파일 내용 쓰기	3	어플리케이션 개발자는 파일에 쓸 Data를 정한다.			
		4	어플리케이션 개발자는 파일에 쓸 Data의 크기를 정한다.			
		5	어플리케이션 개발자는 파일에 Data를 쓸 수 있는 함수를 호출한다.			
		6	파일 시스템은 어플리케이션 개발자는 파일에 쓰여진 Data의 길이를 준다. 그리고 파일에 쓰여진 Data는 파일을 읽었을 때 일치해야 한다.			
		1	어플리케이션 개발자는 파일을 오픈한다.			
		2	어플리케이션 개발자는 존재하지 않는 파일 디스크립터를 사용한다.			
RF-FILE-WRITE-EFLOW-2	읽기 전용 모드의 파일 디스크립터로 파일 내용 쓰기	3	어플리케이션 개발자는 파일에 쓸 Data를 정한다.			
		4	어플리케이션 개발자는 파일에 쓸 Data의 크기를 정한다.			
		5	어플리케이션 개발자는 파일에 Data를 쓸 수 있는 함수를 호출한다.			
		6	파일 시스템은 어플리케이션 개발자는 '0'을 리턴하고 Error함수에서 Error값을 리턴한다.			
		1	어플리케이션 개발자는 읽기 전용 모드로 파일을 오픈 한다.			
		2	어플리케이션 개발자는 오픈한 파일의 디스크립터를 얻는다.			
RF-FILE-WRITE-EFLOW-3	파일 쓸 Data와 쓸 Data의 크기가 일치하지 않을 때 파일 내용 쓰기	3	어플리케이션 개발자는 파일에 쓸 Data를 정한다.			
		4	어플리케이션 개발자는 파일에 쓸 Data와 쓸 Data의 길이가 일치하지 않는다.			
		5	어플리케이션 개발자는 파일에 Data를 쓸 수 있는 함수를 호출한다.			
		6	파일 시스템은 어플리케이션 개발자는 '0'을 리턴하고 Error함수에서 Error값을 리턴한다.			

테스트 명세서

■ 문서정보

이 문서는 소프트웨어 공학센터에서 중소기업에서 테스트 케이스 명세서를 작성할 때, 참조할 수 있는 예로 작성된 문서이다. 이 문서는 크게 2개의 Part로 구성되어 있다. Part I은 테스트 케이스와 Test 결과 정리서가 통합된 형태와 Part II는 테스트 케이스와 Test 결과 정리서가 분리된 형태의 예를 보여 주고 있다.

Part I는 테스트 케이스와 Test 결과 정리서가 하나의 형태로 테스트 케이스의 관리보다는 빠른 결과에 대한 피드백이 필요한 과제에서 사용되는 것으로 많은 문서를 만들지 않아도 되는 장점이 있다. 그러나 테스트 케이스가 지속적으로 관리되지 않기 때문에 제품을 만들면서 발생할 수 있는 특성들이 테스트 케이스에 반영하는데 어려움이 있을 수 있다.

Part II은 테스트 케이스와 Test 결과 정리서가 분리한 형태로 동일한 제품에 대하여 동일한 내용의 Test를 자주 반복할 때 사용하는 사례로 테스트 케이스를 지속적으로 관리할 수 있는 장점이 있다. 그러나 테스트 케이스 작성시 작성해야 할 내용이 많아 테스트 케이스 작성시 많이 시간이 소요되는 단점이 있다.

테스트 케이스 명세서는 과제의 도메인 특성, Test 반복 횟수 정도 그리고 자동화 도구에 따라 작성한 형태 및 내용에 차이가 있기 때문에 각각의 중소기업의 특성에 맞추어 테스트 케이스 명세서 형태를 정의하여 사용할 필요가 있다.

이 문서는 테스트 케이스 명세서의 사례이기 때문에 한 사례의 모든 테스트 케이스를 기술하고 있지 않는다.

Part I

테스트 케이스 명세서와 Test 수행 결과 정리서가 통합되는 경우

Part I은 테스트 케이스 명세서와 Test 수행 결과 정리서가 통합된 사례를 정리하였다. ISO/IEC 29119에서 Test 설계를 하기 위하여 6단계를 정의하였다. 1단계 Test 대상을 파악하고, 2단계, Test 조건을 파악하고, 3단계 Test 커퍼리지 아이템을 정리하고, 4단계 테스트 케이스를 만들고 5단계 Test 집합을 만들고 6단계 Test 수행 절차를 정의한다. 그리고 여기에서는 3개의 산출물을 정의하고 있는데, Test 설계 문서, 테스트 케이스 문서 그리고 Test 순서 문서이다. Part I에서는 Test의 수행의 용의성을 위하여 테스트 케이스 문서가 중심으로 되어 있는 사례이다.

Part I에서 크게 “테스트 케이스 명세서” 와 “Test 수행 결과 정리서” 가 통합된 형태로 정리된 부분과 테스트 케이스를 효율적으로 관리하기 위하여 Redmine의 Plugin Tool인 Impasses에 정리된 형태로 작성된 테스트 케이스가 있다. 실제 도구에서 테스트 케이스를 관리하여 정리하는 샘플을 확인할 수 있으면 도구에 있는 샘플의 실제적인 내용은 통합된 형태로 정리된 부분에서 확인 할 수 있다.

Test 수행 결과서는 Test의 전체 내용, Test 수행 과정과 결과 그리고 Test 수행 결과가 포함되어 있습니다. Test 프로젝트 개요에 Test의 전체 내용이 포함되어 있으며, 각각의 Test 아이템 별로 Test 수행 과정과 결과가 정리되어 있으며 마지막으로 Test 수행 결과를 Test 수행 결과에서 정리하고 있다. 그리고 Impasses에서의 Test 관리를 통하여 실제 도구에서 테스트 케이스를 관리하는 사례를 보여주고 있다.

Part I에서 사용된 예제는 파일시스템의 Test 수행 예제로 정리하고 기술하고 있다. 다양한 사례 중 파일시스템의 예제를 사용하여 기술한 이유는 개발자면 누구나 파일시스템을 사용하기 때문에 이 시스템의 기능을 쉽게 알 수 있을 뿐만 아니라, 오래 전부터 만들고 정리되어 왔기 때문에 정형화된 형식을 가지고 있기 때문이다. 특히 기능 요구사항인 볼륨 관리와 파일 관리의 테스트 케이스를 일부 정리하였다 이 테스트 케이스들은 공학센터에서 이미 만든 요구사항 사례와 연계하여 사용할 수 있으면 이를 통하여 더 쉽게 테스트 케이스 작성 사례를 이해할 수 있을 것이다.

1. Test 프로젝트 개요

« 테스트 케이스 문서화 형태 »

Test 제목	파일 시스템 초기 기능 Test
프로젝트 이름	AFilesystem
Test 수행 날짜	2013.04.15 ~ 2013.04.29
Test 개요	파일 시스템 개발 후 1차 초기 Test
테스트 케이스	TC-VOL-CREATE, TC-FILE-CREATE
Tester	Test 수행 범위
A Tester	파일 시스템 볼륨에 관한 기능 Test 파일 시스템 파일에 관한 기능 Test

« Impasse Tool 관리 형태 »

1. Test Plan 정리 형태 : Impasse Tool은 버전에 여러 개의 Test Plan이 붙는 형식으로 되어 있다. 그리고 프로젝트는 여러 개의 버전을 가질 수 있다. 그래서 프로젝트에 여러 개의 Test Plan을 가지고 Test를 진행할 수 있다.

Tests > test plans list

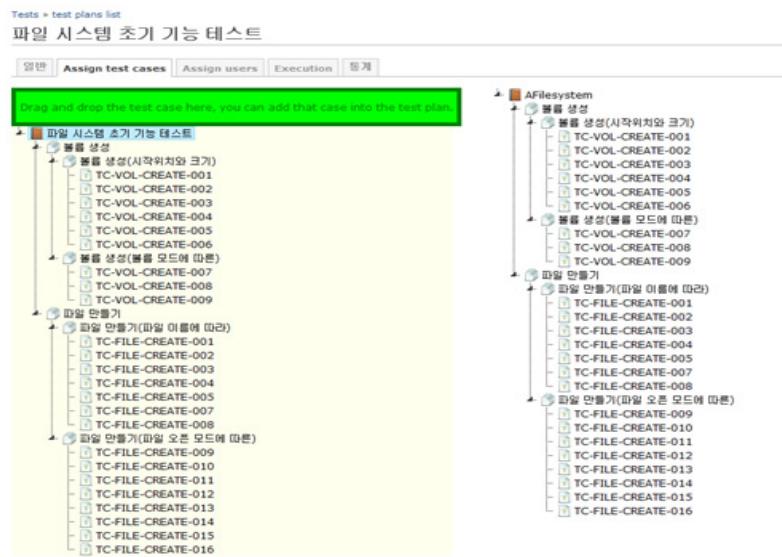
파일 시스템 초기 기능 테스트

일반 Assign test cases Assign users Execution 통계

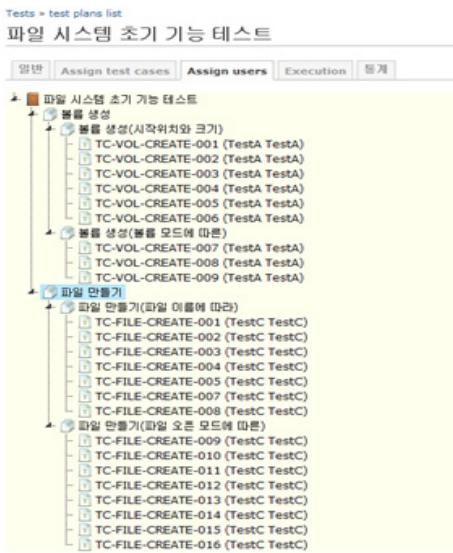
덧글

프로젝트 이름 : AFilesystem
 테스트 수행 날짜 : 2013.04.15 ~ 2013.04.29
 테스트 개요 : 파일 시스템 개발 후 1차 초기 테스트
 테스트 케이스 : TC-VOL-CREATE, TC-FILE-CREATE
 테스터 : A테스터 (파일 시스템 볼륨에 관한 기능 테스트, 파일 시스템 파일에 관한 기능 테스트)

2. Test Plan과 테스트 케이스의 연계 : Test Plan을 작성하면 Test Plan에 Test가 필요한 테스트 케이스를 연계하여 관리할 수 있다.



3. 테스트 케이스에 테스터 할당 : Test Plan에 할당된 테스트 케이스는 Tester에게 각각 할당할 수 있다.



2. 볼륨에 관련된 Test

2.1 볼륨 생성

⟨⟨ 테스트 케이스 문서화 형태 ⟩⟩

분류	테스트 케이스 ID	설명	No	조건	시나리오	기대 값	결과 값	성공/실패/대기	관련 요구 사항
볼륨 생성(시작 위치와 크기)	TC-VOL-CR EATE-001	시작위치와 크기가 정상적인 경우	1	-	생성하려고 하는 볼륨의 시작 위치를 1로 정한다.	-			RF-VOL-CREATE-BFLOW
			2	-	생성하려고 하는 볼륨의 크기를 1로 정한다.	-			
			3	-	생성하려고 하는 볼륨의 모드를 읽기 전용으로 선택한다.	-			
			4	-	볼륨을 생성함수를 호출하고 볼륨구분자를 얻는다.	정수 값			
	TC-VOL-CR EATE-002	시작위치와 크기가 정상적인 경우	1	-	생성하려고 하는 볼륨의 시작 위치를 1로 정한다.	-			RF-VOL-CREATE-BFLOW
			2	-	생성하려고 하는 볼륨의 크기를 MAX-1로 정한다.	-			
			3	-	생성하려고 하는 볼륨의 모드를 읽기 전용으로 선택한다.	-			
			4	-	볼륨을 생성함수를 호출하고 볼륨구분자를 얻는다.	정수 값			
	TC-VOL-CR EATE-003	시작위치와 크기가 정상적인 경우	1	-	생성하려고 하는 볼륨의 시작 위치를 MAX-1로 정한다.	-			RF-VOL-CREATE-BFLOW
			2	-	생성하려고 하는 볼륨의 크기를 1로 정한다.	-			
			3	-	생성하려고 하는 볼륨의 모드를 읽기 전용으로 선택한다.	-			
			4	-	볼륨을 생성함수를 호출하고 볼륨구분자를 얻는다.	정수 값			
	TC-VOL-CR EATE-004	시작위치가 정상이지 않는 경우	1	-	생성하려고 하는 볼륨의 시작 위치를 -1로 정한다.	-			RF-VOL-CREATE-EFLOW1
			2	-	생성하려고 하는 볼륨의 크기를 1로 정한다.	-			
			3	-	생성하려고 하는 볼륨의 모드를 읽기 전용으로 선택한다.	-			
			4	-	볼륨을 생성함수를 호출하고 볼륨구분자를 얻는다.	-1			
			5	-	시스템의 Error값을 확인한다.	에러값 호출			
	TC-VOL-CR EATE-005	크기가 정상적이지 않는 경우	1	-	생성하려고 하는 볼륨의 시작 위치를 1로 정한다.	-			RF-VOL-CREATE-EFLOW2
			2	-	생성하려고 하는 볼륨의 크기를 -1로 정한다.	-			
			3	-	생성하려고 하는 볼륨의 모드를 읽기 전용으로 선택한다.	-			
			4	-	볼륨을 생성함수를 호출하고 볼륨구분자를 얻는다.	-1			

분류	테스트 케이스 ID	설명	No	조건	시나리오	기대 값	결과 값	성공/실패/대기	관련 요구 사항
볼륨 생성(볼륨 모드에 따른)	TC-VOL-CREATE-006	시작위치와 크기가 정상적이지 않을 경우	5	-	시스템의 Error값을 확인한다.	에러값 호출			
			1	-	생성하려고 하는 볼륨의 시작 위치를 -1로 정한다.	-			RF-VOL-CREATE-EFLOW1 / RF-VOL-CREATE-EFLOW2
			2	-	생성하려고 하는 볼륨의 크기를 -1로 정한다.	-			
			3	-	생성하려고 하는 볼륨의 모드를 읽기 전용으로 선택한다.	-			
			4	-	볼륨을 생성함수를 호출하고 볼륨구분자를 얻는다.	-1			
			5	-	시스템의 Error값을 확인한다.	에러값 호출			
	TC-VOL-CREATE-007	읽기 모드에 따른 볼륨 생성	1	-	생성하려고 하는 볼륨의 시작 위치를 1로 정한다.	-			RF-VOL-CREATE-BFLOW
			2	-	생성하려고 하는 볼륨의 크기를 MAX-1로 정한다.	-			
			3	-	생성하려고 하는 볼륨의 모드를 읽기 전용으로 선택한다.	-			
			4	-	볼륨을 생성함수를 호출하고 볼륨구분자를 얻는다.	정수 값			
			5	-	생성된 볼륨에 파일을 생성한다.	-1			
	TC-VOL-CREATE-008	쓰기 모드에 따른 볼륨 생성	1	-	생성하려고 하는 볼륨의 시작 위치를 1로 정한다.	-			RF-VOL-CREATE-BFLOW
			2	-	생성하려고 하는 볼륨의 크기를 MAX-1로 정한다.	-			
			3	-	생성하려고 하는 볼륨의 모드를 쓰기 전용으로 선택한다.	-			
			4	-	볼륨을 생성함수를 호출하고 볼륨구분자를 얻는다.	정수 값			
			5	-	생성된 볼륨에 파일을 생성한다.	정수 값			
			6	-	생성한 볼륨에 파일에 내용을 읽는다.	-1			
	TC-VOL-CREATE-009	읽기/쓰기 모드에 따른 볼륨 생성	1	-	생성하려고 하는 볼륨의 시작 위치를 1로 정한다.	-			RF-VOL-CREATE-BFLOW
			2	-	생성하려고 하는 볼륨의 크기를 MAX-1로 정한다.	-			
			3	-	생성하려고 하는 볼륨의 모드를 읽기/쓰기 전용으로 선택한다.	-			
			4	-	볼륨을 생성함수를 호출하고 볼륨 구분자를 얻는다.	정수 값			
			5	-	생성된 볼륨에 파일을 생성한다.	정수 값			
			6	-	생성한 볼륨에 파일에 내용을 읽는다.	정수 값			

« Impasse Tool 관리 형태 »

1. 테스트 케이스 관리 형태 : Impasse Tool에서는 테스트 케이스는 Test Suite과 테스트 케이스로 관리된다. Test Suite은 Test Suite과 테스트 케이스의 묶음이면 테스트 케이스는 Test 시나리오를 담고 있다.

2. 테스트 케이스 수행 형태 : Test Plan과 Tester가 할당된 테스트 케이스는 Test를 수행하고 결과를 기록할 수 있다.

3. 파일에 관련된 Test

3.1 파일 만들기

⟨⟨ 테스트 케이스 문서화 형태 ⟩⟩

분류	테스트 케이스 ID	설명	No	조건	시나리오	기대값	결과 값	성공/실패/ 대기	관련 요구 사항
파일 만들기 (파일 이름에 따라)	TC-FILE-CRE ATE-001	파일이름이 NULL 인 경우	1	파일 시스템이 Format 되어 있어야함	파일이름에 NULL 를 넣는다.	-			RF-FILE-CREA TE-EFLOW-1
			2	-	파일 생성 함수를 호출한다.	에러값 호출			
			3	-	파일 디스크립터를 얻는다.	-1			
	TC-FILE-CRE ATE-002	파일이름이 A 인경우	1	파일 시스템이 Format 되어 있어야함	파일 이름에 A 를 넣는다.	-			RF-FILE-CREA TE-BFLOW
			2	-	파일 생성 함수를 호출한다.	-			
			3	-	파일 디스크립터를 얻는다.	정수값			
			4	-	파일을 닫는다.	정수값			
			5	-	생성된 파일을 오픈한다.	-			
	TC-FILE-CRE ATE-003	255 자 이상의 파일 이름을 만드는 경우	1	파일 시스템이 Format 되어 있어야함	255 자 이상의 파일 이름(ABCDEFG.....)을 만든다.	-			RF-FILE-CREA TE-BFLOW
			2	-	파일 생성 함수를 호출한다.	-			
			3	-	파일 디스크립터를 얻는다.	-			
			4	-	파일을 닫는다.	정수값			
			5	-	생성된 파일을 오픈한다.	정수값			
	TC-FILE-CRE ATE-004	256 자 이상의 파일 이름을 만드는 경우	1	파일 시스템이 Format 되어 있어야함	256 자 이상의 파일 이름(ABCDEFG.....)을 만든다.	-			RF-FILE-CREA TE-EFLOW-1
			2	-	파일 생성 함수를 호출한다.	에러값 호출			
			3	-	파일 디스크립터를 얻는다.	-1			
	TC-FILE-CRE ATE-005	파일 이름에 특수 문자가 추가 되는 경우	1	파일 시스템이 Format 되어 있어야함	파일이름에 A!를 넣는다.	-			RF-FILE-CREA TE-EFLOW-1
			2	-	파일 생성 함수를 호출한다.	에러값 호출			
			3	-	파일 디스크립터를 얻는다.	-1			
	TC-FILE-CRE ATE-007	대문자 이름으로 파일을 생성한 경우 소문자로 이름으로 파일 열기	1	파일 시스템이 Format 되어 있어야함	파일 이름에 ABCDE 를 넣는다.	-			RF-VOL-CREA TE-BFLOW
			2	-	파일 생성 함수를 호출한다.	-			
			3	-	파일 디스크립터를 얻는다.	-			
			4	-	파일을 닫는다.	정수값			

SW개발 품질관리 매뉴얼

분류	테스트 케이스 ID	설명	No	조건	시나리오	기대값	결과 값	성공/실패/ 대기	관련 요구 사항
파일 만들기(파일 오픈 모드에 따른)	TC-FILE-CRE ATE-008	소문자 이름으로 파일을 생성한 경우 대문자 이름으로 파일 열기	5	-	abcde 이름으로 파일을 오픈한다.	-1			RF-VOL-CREA TE-BFLOW
			6	-	Error 함수를 호출한다.	에러값 호출			
			1	파일 시스템이 Format 되어 있어야함	파일 이름에 abcde 를 넣는다.	-			
			2	-	파일 생성 함수를 호출한다.	-			
			3	-	파일 디스크립터를 얻는다.	-			
			4	-	파일을 닫는다.	정수값			
			5	-	ABCDE 이름으로 파일을 오픈한다.	-1			
			6	-	Error 함수를 호출한다.	에러값 호출			
	TC-FILE-CRE ATE-009	APPEND 모드로 파일 만들기	1	파일 시스템이 Format 되어 있어야함	파일의 이름은 File1 으로 설정한다.	-			RF-FILE-CREA TE-AFLOW-1
			2	-	파일을 APPEND 모드로 설정한다.	-			
			3	-	파일 생성 함수를 호출한다.	-			
			4	-	파일 디스크립터를 얻는다.	정수값			
			5	-	파일 오프셋 위치를 확인한다.	0			
			6	-	파일에 내용을 적는다.	파일에 내용을 적을 수 있음			
			7	-	파일 내용을 읽는다.	파일에 적은 내용을 읽을 수 있음			
	TC-FILE-CRE ATE-010	READ 모드로 파일 만들기	1	파일 시스템이 Format 되어 있어야함	파일의 이름은 File1 으로 설정한다.	-			RF-FILE-CREA TE-AFLOW-1
			2	-	파일을 READ 모드로 설정한다.	-			
			3	-	파일 생성 함수를 호출한다.	-			
			4	-	파일 디스크립터를 얻는다.	정수값			
			5	-	파일 오프셋 위치를 확인한다.	0			
			6	-	파일에 내용을 적는다.	파일에 내용을 적을 수 없음			
			7	-	파일 내용을 읽는다.	-			
	TC-FILE-CRE ATE-011	UPDATE 모드로 파일 만들기	1	파일 시스템이 Format 되어 있어야함	파일의 이름은 File1 으로 설정한다.	-			RF-FILE-CREA TE-AFLOW-1
			2	-	파일을 UPDATE 모드로	-			

분류	테스트 케이스 ID	설명	No	조건	시나리오	기대값	결과 값	성공/실패/ 대기	관련 요구 사항
TC-FILE-CRE ATE-012	WRITE 모드로 파일 만들기				설정한다.				RF-FILE-CREA TE-AFLOW-1
			3	-	파일 생성 함수를 호출한다.	-			
			4	-	파일 디스크립터를 얻는다.	정수값			
			5	-	파일 오프셋 위치를 확인한다.	0			
			6	-	파일에 내용을 적는다.	파일에 내용을 적을 수 있음			
			7	-	파일 내용을 읽는다.	파일에 내용을 읽을 수 있음			
			1	파일 시스템이 Format 되어 있어야함	파일의 이름은 File1 으로 설정한다.	-			
			2	-	파일을 WRITE 모드로 설정한다.	-			RF-FILE-CREA TE-AFLOW-1
			3	-	파일 생성 함수를 호출한다.	-			
			4	-	파일 디스크립터를 얻는다.	정수값			
			5	-	파일 오프셋 위치를 확인한다.	0			
			6	-	파일에 내용을 적는다.	파일에 내용을 적을 수 있음			
			7	-	파일 내용을 읽는다.	파일에 내용을 읽을 수 없음			
			1	파일 시스템이 Format 되어 있어야함	10byte 의 사이즈를 가지고 있는 File1 을 만든다.	-			
TC-FILE-CRE ATE-013	추가로 APPEND 모드로 파일 만들기		2	-	파일의 이름은 File1 으로 설정한다.	-			RF-FILE-CREA TE-AFLOW-1
			3	-	파일을 APPEND 모드로 설정한다.	-			
			4	-	파일 생성 함수를 호출한다.	-			
			5	-	파일 디스크립터를 얻는다.	정수값			
			6	-	파일 오프셋 위치를 확인한다.	10			
			7	-	파일에 내용을 적는다.	파일에 내용을 적을 수 있음			
			8	-	파일 내용을 읽는다.	파일에 적은 내용을 읽을 수 있음			

SW개발 품질관리 매뉴얼

분류	테스트 케이스 ID	설명	No	조건	시나리오	기대값	결과 값	성공/실패/ 대기	관련 요구 사항
TC-FILE-CRE ATE-014	추가로 READ 모드로 파일 만들기		1	파일 시스템이 Format 되어 있어야함	10byte 의 사이즈를 가지고 있는 File1 을 만든다.	-			RF-FILE-CREA TE-AFLOW-1
			2	-	파일의 이름은 File1 으로 설정한다.	-			
			3	-	파일을 READ 모드로 설정한다.	-			
			4	-	파일 생성 함수를 호출한다.	-			
			5	-	파일 디스크립터를 얻는다.	정수값			
			6	-	파일 오프셋 위치를 확인한다.	0			
			7	-	파일에 내용을 적는다.	파일에 내용을 적을 수 없음			
			8	-	파일 내용을 읽는다.	-			
TC-FILE-CRE ATE-015	추가로 UPDATE 모드로 파일 만들기		1	파일 시스템이 Format 되어 있어야함	10byte 의 사이즈를 가지고 있는 File1 을 만든다.	-			RF-FILE-CREA TE-AFLOW-1
			2	-	파일의 이름은 File1 으로 설정한다.	-			
			3	-	파일을 UPDATE 모드로 설정한다.	-			
			4	-	파일 생성 함수를 호출한다.	-			
			5	-	파일 디스크립터를 얻는다.	정수값			
			6	-	파일 오프셋 위치를 확인한다.	0			
			7	-	파일에 내용을 적는다.	파일에 내용을 적을 수 있음			
			8	-	파일 내용을 읽는다.	파일에 내용을 읽을 수 있음			

« Impasse Tool 관리 형태 »

1. 테스트 케이스 관리 형태

Tests

▶ 검색조건
✓ 적용 ⌂ 지우기

└ AFilesystem

- └ 파일 생성
- └ 파일 만들기
 - └ 파일 만들기(파일 이름에 따라)
 - TC-FILE-CREATE-001
 - TC-FILE-CREATE-002
 - TC-FILE-CREATE-003
 - TC-FILE-CREATE-004
 - TC-FILE-CREATE-005
 - TC-FILE-CREATE-006
 - TC-FILE-CREATE-007
 - TC-FILE-CREATE-008**
 - TC-FILE-CREATE-009
 - TC-FILE-CREATE-010
 - TC-FILE-CREATE-011
 - TC-FILE-CREATE-012
 - TC-FILE-CREATE-013
 - TC-FILE-CREATE-014
 - TC-FILE-CREATE-015
 - TC-FILE-CREATE-016

이름 TC-FILE-CREATE-008
요약
소문자 이름으로 파일을 생성한 경우 대문자 이름으로 파일 열기
Preconditions
1. 파일 시스템이 Format 되어 있어야함
Steps

#	Actions	Expected Results
1	파일 이름에 abcde를 넣는다.	-
2	파일 상상 입수를 호출한다.	-
3	파일 디스크립터를 얻는다.	-
4	파일을 닫는다.	정수값
5	ABCDE 이름으로 파일을 오픈한다.	-1
6	Error 함수를 호출한다.	에러값: 호출

이름 TC-FILE-CREATE-016
요약
후기로 WRITE 모드로 파일 만들기
Preconditions
1. 파일 시스템이 Format 되어 있어야함
Steps

#	Actions	Expected Results
1	10byte의 사이즈를 가지고 있는 File1을 만든다.	-
2	파일의 이름은 File1으로 설정한다.	-
3	파일을 WRITE 모드로 설정한다.	-
4	파일 생성 함수를 호출한다.	-
5	파일 디스크립터를 얻는다.	정수값
6	파일 오프셋 위치를 확인한다.	0
7	파일에 내용을 적는다.	파일에 내용을 적을 수 있음
8	파일 내용을 읽는다.	파일에 내용을 읽을 수 없음

2. 테스트 케이스 수행 형태

파일 시스템 초기 기능 테스트

▶ 일반 Assign test cases Assign users Execution 등록

▶ 검색조건
✓ 적용 ⌂ 지우기

└ 파일 시스템 초기 기능 테스트

- └ 파일 생성
- └ 파일 만들기
 - └ 파일 만들기(파일 이름에 따라)
 - TC-FILE-CREATE-001 (User Name Admin)
 - TC-FILE-CREATE-002 (User Name Admin)
 - TC-FILE-CREATE-003 (User Name Admin)
 - TC-FILE-CREATE-004 (User Name Admin)
 - TC-FILE-CREATE-005 (User Name Admin)
 - TC-FILE-CREATE-006 (User Name Admin)
 - TC-FILE-CREATE-007 (User Name Admin)
 - TC-FILE-CREATE-008 (User Name Admin)**
 - TC-FILE-CREATE-009 (User Name Admin)
 - TC-FILE-CREATE-010 (User Name Admin)
 - TC-FILE-CREATE-011 (User Name Admin)
 - TC-FILE-CREATE-012 (User Name Admin)
 - TC-FILE-CREATE-013 (User Name Admin)
 - TC-FILE-CREATE-014 (User Name Admin)
 - TC-FILE-CREATE-015 (User Name Admin)
 - TC-FILE-CREATE-016 (User Name Admin)

이름 TC-FILE-CREATE-009
요약 APPEND 모드로 파일 만들기
Preconditions 1. 파일 시스템이 Format 되어 있어야함
Steps

#	Actions	Expected Results
1	파일의 이름은 File1으로 설정한다.	-
2	파일을 APPEND 모드로 설정한다.	-
3	파일 생성 함수를 호출한다.	-
4	파일 디스크립터를 얻는다.	정수값
5	파일 오프셋 위치를 확인한다.	0
6	파일에 내용을 적는다.	파일에 내용을 적을 수 있음
7	파일 내용을 읽는다.	파일에 적은 내용을 읽을 수 있음

Execution status ⚡ Not yet ⚡ OK ⚡ NG ⚡ Block

결합 #7
댓글 Step7에서 Error가 발생함

4. Test 수행 결과

« 테스트 케이스 문서화 형태 »

전체 Test 수	
성공한 Test 수	
실패한 Test 수	
대기된 Test 수	

« Impasse Tool 관리 형태 »

Impasse는 Tester가 Test를 수행 후 결과를 자동으로 계산되어 관리 된다.

파일 시스템 초기 기능 테스트

일반	Assign test cases	Assign users	Execution	통계
Test suite 파일 시스템 초기 기능 테스트	total cases 24	executed cases 21	bugs 2	(open) 2 progress rate 87.50% bugs rate 8.33%

Part II 테스트 케이스 명세서와 Test 수행 결과 정리서가 분리되는 경우

Part II은 테스트 케이스 명세서와 Test 수행 결과 정리서를 분리하여 정리한 형식으로 사례를 정리하였다. 이미 Part I에서 설명한 것과 같이 ISO/IEC 29119의 Test 설계 절차는 다음과 같다. 1단계 Test 대상을 파악하고, 2단계, Test 조건을 파악하고, 3단계 Test 커퍼리지 아이템을 정리하고, 4단계 테스트 케이스를 만들고 5단계 Test 집합을 만들고 6단계 Test 수행 절차를 정의한다. Part II에서는 테스트 케이스 설계 과정을 보여주기 위하여 1단계에서 6단계까지의 내용을 포함하고 있다.

Part II는 크게 3개의 Chapter로 구성되어 있다. 먼저 테스트 케이스 설계는 ISO/IEC 29119의 Test 설계 절차에서 1단계와 2단계의 내용을 포함하고 있다. 이 부분은 Test 대상을 파악하고 Test 대상의 특성에 따라 Test 기법을 선택하는 것까지 진행한다. Test 기법의 선택은 Test 대상의 특성 및 조건에 의하여 결정되고, 이를 기반으로 Test 커퍼리지 및 테스트 케이스를 도출할 수 있다. 다음 Chapter인 테스트 케이스 명세는 3단계와 4단계의 내용을 담고 있다. 실제 Test를 수행하는 절차의 내용이 포함되어 있으며, Test를 수행할 때 필요한 환경 및 전제 조건도 포함하고 있다. 마지막 Chapter인 Test 수행 결과는 Test를 수행하고 정리하는 단계로 5단계와 6단계의 내용을 포함하고 있다.

Part II의 테스트 케이스 사례를 볼 때, 두 가지 관점에 대하여 고려해야 한다. 첫째는 Test 도출하기 위하여 어떤 설계 기법을 사용하는 것인가와 둘째는 테스트 케이스를 정리하기 위하여 변수값과 기대값을 어떤 도출할 것이고 실제 Test를 수행하면서 어떤 형식으로 사용할 것인지에 대하여 고려해야 한다.

테스트 케이스를 만들고 작성하는데 있어 여러 가지 형식으로 기술할 수 있다. Part I은 하나의 시나리오가 테스트 케이스 하나가 되는 형태이고, Part II는 시나리오와 변수값을 분리하여 하나의 시나리오가 하나의 테스트 케이스가 될 수 있고 하나의 변수가 하나의 테스트 케이스도 될 수 있는 형태이다. 테스트 케이스를 관리도 테스트 케이스를 만드는 조직의 형태에 따라 달라질 수 있으면 합리적인 테스트 케이스를 관리

하기 위해서 조직의 형태 및 성격을 판단하고 그에 맞추어 작성해야 한다.

Part II는 테스트 케이스의 설계 과정을 보여 주기 위하여 시나리오와 변수값이 분리된 형태로 테스트 케이스를 만들었으며, 변수값 별로 테스트 케이스를 두고 관리하고 있는 형태이다.

1. 테스트 케이스 설계

분류	항목	테스트 케이스 ID	설명	테스트 케이스 설계 기법
기능	볼륨	TC-VOL-CREATE	볼륨 만들기	- 경계값 분석 - 조합 테스팅
		TC-VOL-DELETE	볼륨 지우기	- 시나리오 테스팅 - 경계값 분석
		TC-VOL-MOUNT	볼륨 등록	- 시나리오 테스팅
		TC-VOL-UNMOUNT	볼륨 등록 해지	- 시나리오 테스팅
		TC-VOL-GETSTATE	볼륨 속성 읽기	- 시나리오 테스팅
		TC-VOL-SETSTATE	볼륨 속성 부여	- 시나리오 테스팅
	파일	TC-FILE-CREATE	파일 생성	- 시나리오 테스팅 - 조합 테스팅
		TC-FILE-OPEN	파일 열기	- 시나리오 테스팅 - 경계값 분석 - 조합 테스팅
		TC-FILE-READ	파일 내용 읽기	- 시나리오 테스팅 - 경계값 분석 - 조합 테스팅
		TC-FILE-WRITE	파일 내용 쓰기	- 시나리오 테스팅 - 경계값 분석 - 조합 테스팅
		TC-FILE-TRUNCATE	파일 사이즈 변경	- 시나리오 테스팅 - 경계값 분석 - 조합 테스팅
		TC-FILE-DELETE	파일 삭제	- 시나리오 테스팅
		TC-FILE-NAME	파일 이름 변경	- 시나리오 테스팅

분류	항목	테스트 케이스 ID	설명	테스트 케이스 설계 기법
비기능	디렉터리	TC-FILE-GETSTATE	파일 상태 정보 보기	- 시나리오 테스팅
		TC-FILE-CHMOD	파일 접근 권한 변경	- 시나리오 테스팅
		TC-DIR-OPEN	디렉터리 열리	- 시나리오 테스팅 - 경계값 분석
		TC-DIR-CHDIR	디렉터리 변경	- 시나리오 테스팅
		TC-DIR-MARK	디렉터리 접근 권한 변경	- 시나리오 테스팅
		TC-DIR-UNLINK	디렉터리 요소 삭제	- 시나리오 테스팅
		TC-DIR-RMDIR	디렉터리 삭제	- 시나리오 테스팅
		TC-DIR-CREATE	디렉터리 생성	- 시나리오 테스팅
	사용성	RNF-USER-POSIX	POSIX 규약을 준수	- 정적 Test
	성능	RNF-PER-WRITEFILE	파일 쓰기 성능	- 시나리오 테스팅
		RNF-PER-READREAD	파일 읽기 성능	- 시나리오 테스팅
		RNF-PER-CREATEDIR	디렉터리 생성 성능	- 시나리오 테스팅
	가용성	RNF-AVA-LIFE	저장장치 수명	- 시나리오 테스팅
	변경용이 성	RNF-MAIN-ANYSTORAGE	저장장치 호환성	- 정적 Test
		RNF-MAIN-ANYOS	운영체제 호환성	- 정적 Test
	확장성	RNF-PER-INFINITEFILE	파일 최대 생성	- 시나리오 테스팅
		RNF-PER-INFINITEFDIR	디렉터리 최대 생성	- 시나리오 테스팅

2. 테스트 케이스 명세

2.1 볼륨 관련 테스트 케이스

2.1.1 볼륨 만들기

테스트 케이스 ID	TC-VOL-CREATE-001	관련 요구사항	RF-VOL-CREATE
테스트 케이스 제목	시작 위치와 크기에 따른 볼륨 생성	테스트 케이스 분류	정상 수행/ 예외 사항 수행
테스트 케이스 설계 기법	<p>경계값 분석</p> <ul style="list-style-type: none"> - 볼륨의 시작 위치 :-1(저장장치 밖), 0(저장장치 경계), 1(저장장치 안), MAX-1(저장장치 안), MAX(저장장치 경계), MAX+1(저장장치 밖) - 볼륨의 크기 :-1(음수 크기), 0 (0 크기), 1 (정상 크기), MAX -1 (정상 크기), MAX (저장장치의 최대 크기), MAX+1(저장장치의 최대 크기보다 크기) <p>조합 테스팅 :시작 위치와 볼륨의 크기에 대한 조합 (allpairs 기법)</p> <p>기타의 allpairs Table 참조</p>		
테스트 케이스 설명	볼륨을 만드는데 있어 볼륨의 시작과 크기에 대한 정상 생성 여부 확인		
전처리 조건	시스템에서 저장 장치 드라이버를 사용할 수 있어야 함		
후처리 조건	없음		
No	시나리오 내용	변수	예상 결과
1	생성하려고 하는 볼륨의 시작 위치를 정한다.	시작 위치 : Start	-
2	생성하려고 하는 볼륨의 크기를 정한다.	크기 : Size	-
3	생성하려고 하는 볼륨의 모드를 선택한다.	모드 : 읽기 쓰기	-
4	볼륨 생성함수를 호출한다.	-	-
5	볼륨 구분자를 얻는다.		Return Value
6	시스템의 Error값을 호출함		Error Value

기타	MAX : 저장장치가 가질 수 있는 최대 크기			
	allpairs Table			
	Start	Size	Return value	Error value
	-1	-1	-1	에러값 호출
	-1	0	-1	에러값 호출
	-1	1	-1	에러값 호출
	-1	MAX-1	-1	에러값 호출
	-1	MAX	-1	에러값 호출
	-1	MAX+1	-1	에러값 호출
	0	-1	-1	에러값 호출
	0	0	-1	에러값 호출
	0	1	정수값	-
	0	MAX-1	정수값	-
	0	MAX	정수값	-
	0	MAX+1	-1	에러값 호출
	1	-1	-1	에러값 호출
	1	0	-1	에러값 호출
	1	1	정수값	-
	1	MAX-1	정수값	-
	1	MAX	정수값	-
	1	MAX+1	-1	에러값 호출
	MAX-1	-1	-1	에러값 호출
	MAX-1	0	-1	에러값 호출
	MAX-1	1	정수값	-
	MAX-1	MAX-1	-1	에러값 호출
	MAX-1	MAX	-1	에러값 호출
	MAX-1	MAX+1	-1	에러값 호출
	MAX	-1	-1	에러값 호출
	MAX	0	-1	에러값 호출
	MAX	1	-1	에러값 호출
	MAX	MAX-1	-1	에러값 호출
	MAX	MAX	-1	에러값 호출
	MAX	MAX+1	-1	에러값 호출
	MAX+1	-1	-1	에러값 호출
	MAX+1	0	-1	에러값 호출
	MAX+1	1	-1	에러값 호출
	MAX+1	MAX-1	-1	에러값 호출
	MAX+1	MAX	-1	에러값 호출
	MAX+1	MAX+1	-1	에러값 호출

SW개발 품질관리 매뉴얼

테스트 케이스 ID	TC-VOL-CREATE-002	관련 요구사항	RF-VOL-CREATE
테스트 케이스 제목	볼륨 모드에 따른 볼륨 생성	테스트 케이스 분류	정상 수행
테스트 케이스 설계 기법	시나리오 Test 볼륨 모드(읽기 전용, 쓰기 전용, 읽기/쓰기 모드)에 따른 시나리오 작성		
테스트 케이스 설명	볼륨을 만드는데 있어 모드에 따른 정상적으로 동작 여부 확인		
전처리 조건	시스템에서 저장 장치 드라이버를 사용할 수 있어야 함		
후처리 조건	없음		
No	시나리오 내용	변수	예상 결과
1	생성하려고 하는 볼륨의 시작 위치를 정상적으로 정한다.	시작 위치 : 1	-
2	생성하려고 하는 볼륨의 크기를 정상적으로 정한다.	크기 : 100	-
3	생성하려고 하는 볼륨의 모드를 선택한다.	볼륨 모드 테이블 참조	-
4	볼륨 생성함수를 호출한다.	-	-
5	볼륨 구분자를 얻는다.	-	Return Value
6	생성된 볼륨에 파일을 생성한다.	-	Activity 1
7	생성한 볼륨에 파일에 내용을 읽는다.	-	Activity 2
기타	볼륨 모드 테이블		
	볼륨모드	Return Value	Activity 1
	읽기	정수값	파일 생성할 수 없음
	쓰기	정수값	파일 생성할 수 있음
	읽기/쓰기	정수값	파일 생성할 수 없음
			파일을 읽을 수 있음

4.2 파일 관련 테스트 케이스

4.2.1 파일 만들기

테스트 케이스 ID	TC-FILE-CREATE-001	관련 요구사항	RF-FILE-CREATE			
테스트 케이스 제목	다양한 종류의 파일 이름 생성	테스트 케이스 분류	정상 수행/ 예외 사항 수행			
테스트 케이스 설계 기법	<p>경계값 분석</p> <ul style="list-style-type: none"> - 파일 이름의 길이 : 0개, 1개, 255개, 256개 - 시나리오 Test - 특수 문자 ! 추가된 경우 - 대문자 이름의 파일 이름 만들고 소문자로 파일 열기 - 소문자 이름의 파일 이름 만들고 대문자로 파일 열기 					
테스트 케이스 설명	파일 이름 생성 규칙에 맞추어 파일 이름을 생성하여 파일 이름 생성 규칙에 따라 생성되는지를 확인함					
전처리 조건	시스템의 저장 장치가 정상적으로 등록이 되어야 함					
후처리 조건	파일을 조절하기 위하여 만든 파일 디스크립터를 반납함					
No	시나리오 내용	변수	예상 결과			
1	파일을 생성한다.	파일 이름	-			
2	파일 생성 함수를 호출한다.	-	Error1			
3	파일 디스크립터를 얻는다.	-	File Descriptor1			
4	파일을 닫는다.	-				
5	생성된 파일을 오픈한다	-	File Descriptor2			
5_E1	대문자 이름으로 생성한 경우 소문자 이름으로 파일 열기		File Descriptor2			
5_E2	소문자 이름으로 생성한 경우 대문자 이름으로 파일 열기		File Descriptor2			
6	Error함수를 호출한다.		Error2			
기타	파일 이름 테이블					
	사항	파일 이름	Error1	File Descriptor1	File Descriptor2	Error2
	정상	0 개	에러값 호출	-1	-1	에러값 호출
	정상	1 개	-	정수값	정수값	-
	정상	255 개	-	정수값	정수값	-
	정상	256 개	에러값 호출	-1	-1	에러값 호출
	정상	특수 문자 ! 추가된 경우	에러값 호출	-1	-1	에러값 호출
	E1	대문자 이름으로 생성한 경우 소문자로 이름으로 파일 열기	-	정수값	-1	에러값 호출
	E2	소문자 이름으로 생성한 경우 대문자 이름으로 파일 열기	-	정수값	-1	에러값 호출

SW개발 품질관리 매뉴얼

테스트 케이스 ID	TC-FILE-CREATE-002	관련 요구사항	RF-FILE-CREATE																																																															
테스트 케이스 제목	파일 모드에 따른 파일 생성	테스트 케이스 분류	정상 수행																																																															
테스트 케이스 설계 기법	<p>시나리오 Test</p> <ul style="list-style-type: none"> - 파일 오픈 모드 <ul style="list-style-type: none"> ■ APPEN :쓰기전용모드파일을열때파일이없다면생성합니다 ■ READ :읽기전용 모드. 파일이 반드시 있어야 함 ■ UPDATE :읽기/쓰기 모드. 파일을 열 때 파일이 없다면 생성함 ■ WRITE : 쓰기전용 모드. 파일이 없다고 생성함. 파일이 있다면 덮어쓰기 																																																																	
테스트 케이스 설명	파일 오픈 모드에 맞추어 파일을 생성하고 파일 모드대로 생성되었는지를 확인 한다.																																																																	
전처리 조건	시스템의 저장 장치가 정상적으로 등록이 되어야 함																																																																	
후처리 조건	파일을 조절하기 위하여 만든 파일 디스크립터를 반납함																																																																	
No	시나리오 내용	변수	예상 결과																																																															
0_A	10바이트 크기를 가진 파일을 먼저 생성한다.	-	-																																																															
1	파일의 이름은 File1이다.	-	-																																																															
2	파일 오픈 모드를 선택한다.	파일 오픈 모드																																																																
3	파일 생성 함수를 호출한다.	-	Error1																																																															
4	파일 디스크립터를 얻는다.	-	File Descriptor1																																																															
5	파일 오프셋 위치를 확인한다.	-	File Offset																																																															
6	파일에 내용을 적는다.	-	Activity 1																																																															
7	파일의 내용을 읽는다.	-	Activity 2																																																															
기타	<p>파일 모드 테이블</p> <table border="1"> <thead> <tr> <th>사항</th><th>파일 오픈 모드</th><th>Error1</th><th>File Descriptor1</th><th>File Offset</th><th>Activity 1</th><th>Activity 2</th></tr> </thead> <tbody> <tr> <td>정상</td><td>APPEND</td><td>-</td><td>정수값</td><td>0</td><td>내용을 적을 수 있다.</td><td>내용을 읽을 수 있다.</td></tr> <tr> <td>정상</td><td>READ</td><td>-</td><td>정수값</td><td>0</td><td>내용을 적을 수 없다.</td><td>-</td></tr> <tr> <td>정상</td><td>UPDATE</td><td>-</td><td>정수값</td><td>0</td><td>내용을 적을 수 있다.</td><td>내용을 읽을 수 있다.</td></tr> <tr> <td>정상</td><td>WRITE</td><td>-</td><td>정수값</td><td>0</td><td>내용을 적을 수 없다.</td><td>내용을 읽을 수 없다.</td></tr> <tr> <td>A</td><td>APPEND</td><td>-</td><td>정수값</td><td>10</td><td>내용을 적을 수 있다.</td><td>내용을 읽을 수 있다.</td></tr> <tr> <td>A</td><td>READ</td><td>-</td><td>정수값</td><td>0</td><td>내용을 적을 수 없다.</td><td>내용을 읽을 수 있다.</td></tr> <tr> <td>A</td><td>UPDATE</td><td>-</td><td>정수값</td><td>0</td><td>내용을 적을 수 있다.</td><td>내용을 읽을 수 있다.</td></tr> <tr> <td>A</td><td>WRITE</td><td>-</td><td>정수값</td><td>0</td><td>내용을 적을 수 없다.</td><td>내용을 읽을 수 없다.</td></tr> </tbody> </table>			사항	파일 오픈 모드	Error1	File Descriptor1	File Offset	Activity 1	Activity 2	정상	APPEND	-	정수값	0	내용을 적을 수 있다.	내용을 읽을 수 있다.	정상	READ	-	정수값	0	내용을 적을 수 없다.	-	정상	UPDATE	-	정수값	0	내용을 적을 수 있다.	내용을 읽을 수 있다.	정상	WRITE	-	정수값	0	내용을 적을 수 없다.	내용을 읽을 수 없다.	A	APPEND	-	정수값	10	내용을 적을 수 있다.	내용을 읽을 수 있다.	A	READ	-	정수값	0	내용을 적을 수 없다.	내용을 읽을 수 있다.	A	UPDATE	-	정수값	0	내용을 적을 수 있다.	내용을 읽을 수 있다.	A	WRITE	-	정수값	0	내용을 적을 수 없다.	내용을 읽을 수 없다.
사항	파일 오픈 모드	Error1	File Descriptor1	File Offset	Activity 1	Activity 2																																																												
정상	APPEND	-	정수값	0	내용을 적을 수 있다.	내용을 읽을 수 있다.																																																												
정상	READ	-	정수값	0	내용을 적을 수 없다.	-																																																												
정상	UPDATE	-	정수값	0	내용을 적을 수 있다.	내용을 읽을 수 있다.																																																												
정상	WRITE	-	정수값	0	내용을 적을 수 없다.	내용을 읽을 수 없다.																																																												
A	APPEND	-	정수값	10	내용을 적을 수 있다.	내용을 읽을 수 있다.																																																												
A	READ	-	정수값	0	내용을 적을 수 없다.	내용을 읽을 수 있다.																																																												
A	UPDATE	-	정수값	0	내용을 적을 수 있다.	내용을 읽을 수 있다.																																																												
A	WRITE	-	정수값	0	내용을 적을 수 없다.	내용을 읽을 수 없다.																																																												

3. Test 수행 결과

3.1 Test 수행 개요

Test 제목	파일 시스템 초기 기능 Test
프로젝트 이름	AFilesystem
Test 수행 날짜	2013.04.15 ~ 2013.04.29
Test 개요	파일 시스템 개발 후 1차 초기 Test
테스트 케이스	TC-VOL-CREATE
테스터	Test 수행 범위
A 테스터	파일 시스템 볼륨에 관한 기능 Test 파일 시스템 파일에 관한 기능 Test 파일 시스템 디렉토리에 관한 기능 Test 파일 시스템 성능 Test

3.2 Test 결과표

테스트 케이스 ID	변수값		기대값		실제값		Pass/fail
	start	size	Return value	Error value	Return value	Error value	
TC-VOL-CREATE-001	start	size	Return value	Error value	Return value	Error value	
TC-VOL-CREATE-001-001	-1	-1	-1	예외값 호출			
TC-VOL-CREATE-001-002	-1	0	-1	예외값 호출			
TC-VOL-CREATE-001-003	-1	1	-1	예외값 호출			
TC-VOL-CREATE-001-004	-1	MAX-1	-1	예외값 호출			
TC-VOL-CREATE-001-005	-1	MAX	-1	예외값 호출			
TC-VOL-CREATE-001-006	-1	MAX+1	-1	예외값 호출			
TC-VOL-CREATE-001-007	0	-1	-1	예외값 호출			
TC-VOL-CREATE-001-008	0	0	-1	예외값 호출			
TC-VOL-CREATE-001-009	0	1	정수값	정상값 호출			
TC-VOL-CREATE-001-010	0	MAX-1	정수값	정상값 호출			
TC-VOL-CREATE-001-011	0	MAX	정수값	정상값 호출			
TC-VOL-CREATE-001-012	0	MAX+1	-1	예외값 호출			
TC-VOL-CREATE-001-013	1	-1	-1	예외값 호출			
TC-VOL-CREATE-001-014	1	0	-1	예외값 호출			
TC-VOL-CREATE-001-015	1	1	정수값	정상값 호출			
TC-VOL-CREATE-001-016	1	MAX-1	정수값	정상값 호출			
TC-VOL-CREATE-001-017	1	MAX	정수값	정상값 호출			
TC-VOL-CREATE-001-018	1	MAX+1	-1	예외값 호출			
TC-VOL-CREATE-001-019	MAX-1	-1	-1	예외값 호출			
TC-VOL-CREATE-001-020	MAX-1	0	-1	예외값 호출			
TC-VOL-CREATE-001-021	MAX-1	1	정수값	정상값 호출			

SW개발 품질관리 매뉴얼

테스트 케이스 ID	변수값		기대값		실제값		Pass/fail
TC-VOL-CREATE-001-022	MAX-1	MAX-1	-1	에러값 호출			
TC-VOL-CREATE-001-023	MAX-1	MAX	-1	에러값 호출			
TC-VOL-CREATE-001-024	MAX-1	MAX+1	-1	에러값 호출			
TC-VOL-CREATE-001-025	MAX	-1	-1	에러값 호출			
TC-VOL-CREATE-001-026	MAX	0	-1	에러값 호출			
TC-VOL-CREATE-001-027	MAX	1	-1	에러값 호출			
TC-VOL-CREATE-001-028	MAX	MAX-1	-1	에러값 호출			
TC-VOL-CREATE-001-029	MAX	MAX	-1	에러값 호출			
TC-VOL-CREATE-001-030	MAX	MAX+1	-1	에러값 호출			
TC-VOL-CREATE-001-031	MAX+1	-1	-1	에러값 호출			
TC-VOL-CREATE-001-032	MAX+1	0	-1	에러값 호출			
TC-VOL-CREATE-001-033	MAX+1	1	-1	에러값 호출			
TC-VOL-CREATE-001-034	MAX+1	MAX-1	-1	에러값 호출			
TC-VOL-CREATE-001-035	MAX+1	MAX	-1	에러값 호출			
TC-VOL-CREATE-001-036	MAX+1	MAX+1	-1	에러값 호출			

테스트 케이스 ID	변수값	기대값			실제값			Pass/fail
		Return	Activity 1	Activity 2	Return	Activity 1	Activity 2	
TC-VOL-CREATE-002	불룹 모드	Return	Activity 1	Activity 2				
TC-VOL-CREATE-002-001	읽기	정수값	파일 생성할 수 없음	-				
TC-VOL-CREATE-002-002	쓰기	정수값	파일 생성할 수 있음	파일 읽을 수 없음				
TC-VOL-CREATE-003-003	읽기/쓰기	정수값	파일 생성할 수 있음	파일 읽을 수 있음				

테스트 케이스 ID	변수값	기대값			실제값			Pass/fail	
		Error1	FD1	FD2	Error2	Error1	FD1	FD2	Error2
TC-FILE-CREATE-001	파일이름								
TC-FILE-CREATE-001-001	0개(NULL)	에러값 호출	-1	-1	에러값 호출				
TC-FILE-CREATE-001-002	1개(a)	-	정수 값	정수 값	-				
TC-FILE-CREATE-001-003	255개	-	정수 값	정수 값	-				
TC-FILE-CREATE-001-004	256개	에러값 호출	-1	-1	에러값 호출				
TC-FILE-CREATE-001-005	! 포함	에러값 호출	-1	-1	에러값 호출				
TC-FILE-CREATE-001-006	대문자 쓰고 소문자읽기	-	정수 값	-1	에러값 호출				
TC-FILE-CREATE-001-007	소문자 쓰고 대문자 읽기	-	정수 값	-1	에러값 호출				

테스트 케이스 ID	사전조건	변수값	기대값					실제값					Pass /fail
TC-FILE-CREATE-002	-	파일 오픈 모드	Error1	FD1	Offset	ACT1	ACT2	Error1	FD1	Offset	ACT1	ACT2	
TC-FILE-CREATE-002-001	-	APPEN	-	정수값	0	Write 가능	Read 가능						
TC-FILE-CREATE-002-002	-	READ	-	정수값	0	Write 불가능	-						
TC-FILE-CREATE-002-003		UPDATE	-	정수값	0	Write 가능	Read 가능						
TC-FILE-CREATE-002-004		WRITE	-	정수값	0	Write 가능	Read 불가능						
TC-FILE-CREATE-002-005	10바이트 파일 생성	APPEN	-	정수값	10	Write 가능	Read 가능						
TC-FILE-CREATE-002-006	10바이트 파일 생성	READ	-	정수값	0	Write 불가능	-						
TC-FILE-CREATE-002-007	10바이트 파일 생성	UPDATE	-	정수값	0	Write 가능	Read 가능						
TC-FILE-CREATE-002-008	10바이트 파일 생성	WRITE	-	정수값	0	Write 가능	Read 불가능						

3.3 Test 결과 요약 정리

전체 Test 수	
성공한 Test 수	
실패한 Test 수	
대기된 Test 수	

소프트웨어 아키텍처 설계 문서(양식서)

1. 소프트웨어 요구사항 분석

가. 요구사항 분석 절차(수정된)

요구사항 분석 과정의 배경과 목적, 수행 절차, 담당자별 역할, 활동별 수행 내용 등에 대한 내용을 소개한다.

1) 요구사항 분석 절차

수정 및 확정된 요구사항 분석 절차

2) 담당자별 역할

담당자별 역할 정의

3) 입력/출력물 정의

수정 및 확정된 입력/출력물 정의

4) 절차별 수행 방법

수정 및 확정된 요구사항 분석 절차 수행 방법

나. 비즈니스 관점의 시스템 환경

- 1) 배경, 목적 등
- 2) 중요한 비즈니스 목표 혹은 미션
- 3) 우선순위화된 비즈니스 목표 목록

비즈니스 목표의 정리 및 우선순위화 작업 시 아래의 서식을 이용하여 작성한다.

표 1. 비즈니스 목표 목록

기호	내용	중요도	구현성	합계
BG2	비즈니스 목표 1	5	1	6
BG1	비즈니스 목표 2	3	5	8
BG3	비즈니스 목표 3	1	5	6
	비즈니스 목표 4			
	:			

4) 마케팅/비즈니스 전략

- 가) 전략/전술 내용
- 나) 시장 경쟁력 분석 내용
- 다) 제품 차별화 내용

5) 시장 분석(환경) 및 출시 일정

- 가) 목표 시장
- 나) 공개 일정

다) 제품 포지셔닝 (목표로 하는 고객, 제품을 사야하는 이유, 제품 카테고리, 제품이 주는 주요 혜택, 주요 경쟁자, 주요 차별점 등)

6) 가용 예산 및 예상 소요 비용

7) 비즈니스 모델 및 수익 전망

다. 시스템 환경 이해

개발 초기의 상위 레벨 기능 요구사항, 시스템 제약사항, 기술적 환경 등을 파악하기 위하여 관련된 내용을 소개하고 이해한다. 특히 시스템(H/W)에 관련된 제반 사항에 대하여 소개한다.

1) 제품 명세 및 시스템 구조

2) 기술적 관점의 아키텍처 환경

3) 문제 도메인 관점의 아키텍처 환경

4) 기존에 존재하는 설계 자료(시스템 개념도, 시스템 구성도 등)

5) 상위 레벨 기능 요구사항 목록

표 2. 상위 레벨 기능 요구사항 목록

기호	내용	중요도	구현성	합계
HFR3	상위레벨 기능 요구사항 1	5	1	6
HFR1	상위레벨 기능 요구사항 2	3	5	8
HFR2	상위레벨 기능 요구사항 3	1	5	6
	상위레벨 기능 요구사항 4			
	:			

6) 시스템 제약사항

표 3. 제약사항 목록

기호	내용	출처
C1	제약사항 1	시스템
C2	제약사항 2	비즈니스
C3	제약사항 3	테스트
C4	제약사항 4	개발 환경
	:	

7) 위험 요소

8) 지원 모델 및 다른 제품과의 연동성

9) 과거 시스템과 연관성

10) 제품의 확장 범위

라. 중요 기능 요구사항 식별

기 작성된 기능 요구사항 문서 및 상위 레벨 요구사항을 기반으로 내용을 수정하거나 추가한 후, 중요도와 구현성 항목의 내용을 결정하여 중요 기능 요구사항을 식별한다.

1) 기능요구사항 우선순위화

표 4. 중요 기능 요구사항 목록

기호	내용	중요도	구현성	합계
FR3	기능 요구사항 1	5	1	6
FR1	기능 요구사항 2	3	5	8
FR2	기능 요구사항 3	1	5	6
	기능 요구사항 4			
	:			

마. 핵심 품질속성 식별

아키텍처 설계의 중심축이 되는 핵심 품질속성을 식별하고, 정리하여 우선순위화 한다. 핵심 품질속성은 앞에서 설명한 품질속성의 형태로 주로 표현되며 종종 비즈니스 목표, 상위 레벨 요구사항 등의 다양한 형태로도 표현이 가능하다. 아키텍처 설계에 핵심적으로 요구되는 품질속성을 관련된 참석자들이 이해하기 편하게 정의하고 우선순위화 하여 아키텍처 설계 및 시나리오 작성에 도움을 준다.

1) 핵심 품질속성 목록

표 5. 핵심 품질속성 목록

기호	내용	관련 항목	중요도	구현성	합계
QA3	품질속성 1	BG1, HFR1, C3	5	1	6
QA1	품질속성 2	HFR2, C1, FR1, FR2, BG4	3	5	8
QA2	품질속성 3	BG2, HFR3, C2	1	5	6
	품질속성 4	BG3, HFR4, FR3			
	:	(양식 편의상 기호로만 표시)			

바. 품질속성 시나리오 목록

여러 이해관계자들이 요구하는 품질속성의 구체적인 형태를 시나리오 형식으로 풀어내어 작성하는 것이다. 참석자들이 생각하는 품질속성의 구체적인 모습을 자유롭게 발표(브레인스토밍)하고 발표된 내용을 정리하여 시나리오 형식으로 작성한다.

1) 품질 속성 1

가) 파생 시나리오 1

표 6. 시나리오 목록

항목	내용
시나리오	- 아래의 6가지 항목을 포함하여 문장으로 서술
자극 유발원 (Source)	- 자극을 만들어내는 존재로써 사람, 컴퓨터 시스템, 기타 장치 등이다.
자극 (Stimulus)	- 시스템의 반응의 원인이 되는 조건으로 무언가가 시스템에 도달했을 때 고려해 볼 필요가 있는 것이다.

항목	내용
대상체 (Artifact)	- 자극에 의해 자극을 받는 대상으로 전체 시스템이 될 수도 있고 시스템의 일부가 될 수도 있다.
환경 (Environment)	- 자극은 특정 상황이나 특정한 환경에서 발생한다. 자극이 발생 했을 때 시스템은 과부하 상태로 존재할 수도 있으며 정상 상태에 있을 수도 있다. 해당 자극이 발생할 때나 혹은 다른 조건이 만족되었을 때 시스템의 상태를 말한다.
응답 (Response)	- 자극이 시스템에 도달한 이후에 취해지는 행위이다.
응답 측정 (Response Measure)	- 요구사항의 검증이 가능한 형태로, 응답이 발생할 때 측정이 가능한 대응의 결과값을 말한다.

나) 파생 시나리오 2

표 7. 시나리오 목록

항목	내용
시나리오	- 아래의 6가지 항목을 포함하여 문장으로 서술
자극 유발원 (Source)	- 자극을 만들어내는 존재로써 사람, 컴퓨터 시스템, 기타 장치 등이다.
자극 (Stimulus)	- 시스템의 반응의 원인이 되는 조건으로 무언가가 시스템에 도달했을 때 고려해 볼 필요가 있는 것이다.
대상체 (Artifact)	- 자극에 의해 자극을 받는 대상으로 전체 시스템이 될 수도 있고 시스템의 일부가 될 수도 있다.
환경 (Environment)	- 자극은 특정 상황이나 특정한 환경에서 발생한다. 자극이 발생 했을 때 시스템은 과부하 상태로 존재할 수도 있으며 정상 상태에 있을 수도 있다. 해당 자극이 발생할 때나 혹은 다른 조건이 만족되었을 때 시스템의 상태를 말한다.
응답 (Response)	- 자극이 시스템에 도달한 이후에 취해지는 행위이다.
응답 측정 (Response Measure)	- 요구사항의 검증이 가능한 형태로, 응답이 발생할 때 측정이 가능한 대응의 결과값을 말한다.

다) 등등

2) 품질속성 2

- 가) 파생 시나리오 1
- 나) 파생 시나리오 2
- 다) 등등

3) 품질속성 3

- 가) 파생 시나리오 1
- 나) 파생 시나리오 2
- 다) 등등

사. 시나리오 우선순위화

자유 토론을 통하여 도출된 시나리오를 투표를 통하여 우선순위화 한다. 참석자들의 투표를 통해서 결정된 중요한 시나리오를 바탕으로 설계 뷰 작성 및 설계 검증 과정을 수행한다.

1) 우선순위화 결과

표 8. 우선순위화된 시나리오 목록

기호	내용	중요도	구현성	합계
S3	시나리오 1 (서술되어 문장으로 표현된 시나리오로 작성)	5	1	6
S1	시나리오 2	3	5	8
S2	시나리오 3	1	5	6
	시나리오 4			
	:			

2) 상위 우선 순위 시나리오 정제

우선순위화된 시나리오 중 상위의 4~5개정도의 중요 시나리오를 대상으로 좀 더 상세하게 분석한다.

가) 핵심 시나리오 목록

표 9. 핵심 시나리오 정제 목록

기호	내용	관련 품질속성	관련 항목
S1	시나리오 S1(기호로 표시된 내용은 해당 내용으로 작성)	QA1	HFR2, C1, FR1, FR2, BG4
		QA2	FR2, BG4
S2	시나리오 S2	QA3	BG1, HFR1, C3
S3	시나리오 S3	QA4	BG2, HFR3, C2
S4	시나리오 S4	QA5	BG3, HFR4, FR3
	:	:	:

나) 핵심 시나리오 정제 목록

(1) 1순위 시나리오

표 10. 정제 시나리오

항목	내용
시나리오	아래의 항목을 포함하여 문장으로 서술
자극 유발원 (Source)	- 자극을 만들어내는 존재로써 사람, 컴퓨터 시스템, 기타 장치 등이다.
정제 1	- 자극을 발생시킬 때 자극원의 구체적인 상황
자극 (Stimulus)	- 시스템의 반응의 원인이 되는 조건으로 무언가가 시스템에 도달했을 때 고려해 볼 필요가 있는 것이다.
정제 2	- 자극이 발생했을 때의 구체적인 시스템의 상황

항목	내용
대상체 (Artifact)	- 자극에 의해 자극을 받는 대상으로 전체 시스템이 될 수도 있고 시스템의 일부가 될 수도 있다.
정제 3	- 자극을 받았을 때 대상체의 구체적인 상황
환경 (Environment)	- 자극은 특정 상황이나 특정한 환경에서 발생한다. 자극이 발생 했을 때 시스템은 과부하 상태로 존재할 수도 있으며 정상 상태에 있을 수도 있다. 해당 자극이 발생할 때나 혹은 다른 조건이 만족되었을 때 시스템의 상태를 말한다.
정제 4	- 여러 환경(서브 시스템// 외부 시스템)이 가지는 조건이나 상태를 나열
응답 (Response)	- 자극이 시스템에 도달한 이후에 취해지는 행위이다.
정제 5	- 자극으로부터 결과를 측정하기 위한 수행되어야 하는 행동
응답 측정 (Response Measure)	- 요구사항의 검증이 가능한 형태로, 응답이 발생할 때 측정이 가능한 대응의 결과값을 말한다.
정제 6	- 결과값을 측정하기 위한 방법을 명시

(2) 2순위 시나리오

표 11. 정제 시나리오

항목	내용
시나리오	아래의 항목을 포함하여 문장으로 서술
자극 유발원 (Source)	- 자극을 만들어내는 존재로써 사람, 컴퓨터 시스템, 기타 장치 등이다.
정제 1	- 자극을 발생시킬 때 자극원의 구체적인 상황
자극 (Stimulus)	- 시스템의 반응의 원인이 되는 조건으로 무언가가 시스템에 도달했을 때 고려해 볼 필요가 있는 것이다.
정제 2	- 자극이 발생했을 때의 구체적인 시스템의 상황

항목	내용
대상체 (Artifact)	- 자극에 의해 자극을 받는 대상으로 전체 시스템이 될 수도 있고 시스템의 일부가 될 수도 있다.
정제 3	- 자극을 받았을 때 대상체의 구체적인 상황
환경 (Environment)	- 자극은 특정 상황이나 특정한 환경에서 발생한다. 자극이 발생 했을 때 시스템은 과부하 상태로 존재할 수도 있으며 정상 상태에 있을 수도 있다. 해당 자극이 발생할 때나 혹은 다른 조건이 만족되었을 때 시스템의 상태를 말한다.
정제 4	- 여러 환경(서브 시스템/ 외부 시스템)이 가지는 조건이나 상태를 나열
응답 (Response)	- 자극이 시스템에 도달한 이후에 취해지는 행위이다.
정제 5	- 자극으로부터 결과를 측정하기 위한 수행되어야 하는 행동
응답 측정 (Response Measure)	- 요구사항의 검증이 가능한 형태로, 응답이 발생할 때 측정이 가능한 대응의 결과값을 말한다.
정제 6	- 결과값을 측정하기 위한 방법을 명시

(3) 3순위 시나리오

(4) 4순위 시나리오

(5) 그 외 (반복)

2. 설계 뷰 작성

가. 아키텍처 요구사항 검토

요구사항 분석 과정에서 도출된 산출물 혹은 추가로 작성되거나 수집된 자료를 대상으로 아키텍처 설계에 적용 가능한지 확인한다. 설계 팀의 주도로 입력물에 해당하는 문서나 자료를 확인하는 활동으로 입력물 각각의 항목에 따라 올바르게 작성되었는지 확인하고, 원하는 품질 수준에 모자라는 자료는 보완하거나 추가한다. 만약, 분석과정의 참여인원의 대부분이 중복하여 뷰 작성 과정을 수행한다면 생략하는 것을 권장한다.

1) 체크리스트 목록

표 12. 아키텍처 요구사항 체크리스트

기호	내용	품질 점수	결과
1	점검 검토 내용 1	1	불만족
2	점검 검토 내용 2	9	만족
3	점검 검토 내용 3	10	만족
4	점검 검토 내용 4		
	:		

나. 기능 요구사항 검토

요구사항 분석 과정에서 도출된 기능 요구사항 혹은 추가로 작성되거나 수집된 기능 요구사항 자료를 대상으로 확인한다.

표 13. 기능 요구사항 체크리스트

기호	내용	품질 점수	결과
1	점검 검토 내용 1	1	불만족
2	점검 검토 내용 2	9	만족
3	점검 검토 내용 3	10	만족
4	점검 검토 내용 4		
	:		

다. 아키텍처 드라이버 식별

분할 대상을 선정하고 그 대상에 대한 아키텍처 드라이버를 선정한다. 처음으로 분할을 수행할 경우 시스템 전체를 대상으로 드라이버를 식별한다. 아키텍처 드라이버는 아키텍처에 중요한 요인으로 작용하는 모든 품질 요구사항을 대상으로 도출되며, 개발하는 시스템 혹은 모듈 설계에 영향을 미치는 품질 요구사항을 선정하게 된다.

1) 우선순위화된 아키텍처 드라이버 목록

표 14. 우선순위화된 아키텍처 드라이버 목록

기호	아키텍처 드라이버	관련항목	중요도	구현성	합계
AD3	아키텍처 드라이버 1	S1	5	1	6
AD1	아키텍처 드라이버 2	C1	3	5	8
AD2	아키텍처 드라이버 3	FR1	1	5	6
	아키텍처 드라이버 4	S2			
	:				

라. 아키텍처 패턴 및 설계전술(tactic) 선정

분할 대상의 아키텍처 드라이버를 기반으로 전반적인 아키텍처 구조를 수립하는 것이다. 아키텍처 품질 요구사항을 만족시키기 위하여 선정된 설계전술(tactic)들의 집합으로 볼 수 있는 패턴(pattern/style)을 선택한다. 패턴의 선택 시에는 후보 패턴들을 나열하고 그 중 아키텍처 드라이버를 가장 잘 만족시키면서 다른 요소의 품질속성과 트레이드오프(trade-off)가 가장 효율적인 패턴을 선정한다. 만약 패턴을 적용시키기 힘들 경우 설계전술의 결합체나 하나의 설계전술을 구현하는 구조를 선정한다.

1) 후보 패턴 및 설계전술

표 15. 후보 패턴 정리

후보 패턴	세부 설계전술	관련 품질 요구사항	관련 드라이버	트레이드오프	요소 값
후보 패턴 1	설계전술 1	S1, C1, FR1	AD1, AD4	AD1++, AD4-	재시작 시간, 상호의존 개수 등
	설계전술 2	C2, FR2	AD1, AD2	AD1+, AD2+	
	:				
후보 패턴 2	설계전술 1				
	설계전술 2				
	:				
후보 패턴 3	:				
:					

2) 선정 패턴 정리

표 16. 선정 패턴 정리

기호	선정 패턴 1				선정 패턴 2				패턴 3 ⋮
	강점	약점	트레이드오프	선정 사유	강점	약점	트레이드오프	선정 사유	
AD1	장점	단점	++	이유	장점	단점	+	이유	⋮
AD2							+		⋮
AD3									⋮
AD4			-						⋮
⋮									⋮

마. 모듈 분할 및 책임 할당

아키텍처 품질 요구사항을 만족시키기 위하여 선정된 패턴의 구조에 실질적인 기능을 할당하고, 그에 대한 책임까지 부여한다. 즉, 패턴의 일반화 되어 있는 구조에 실질적인 기능 요구사항, 비기능 요구사항, 제약사항 등을 할당하여 구체적인 모듈 분할을 완성하고, 과제의 특정 상황에 맞게 인스턴스화한다. 패턴과 설계전술로 모듈의 타입을 결정하고, 결정된 모듈 타입에 책임을 할당하여 실제적으로 인스턴스화한다.

1) 모듈 분할 및 책임 할당

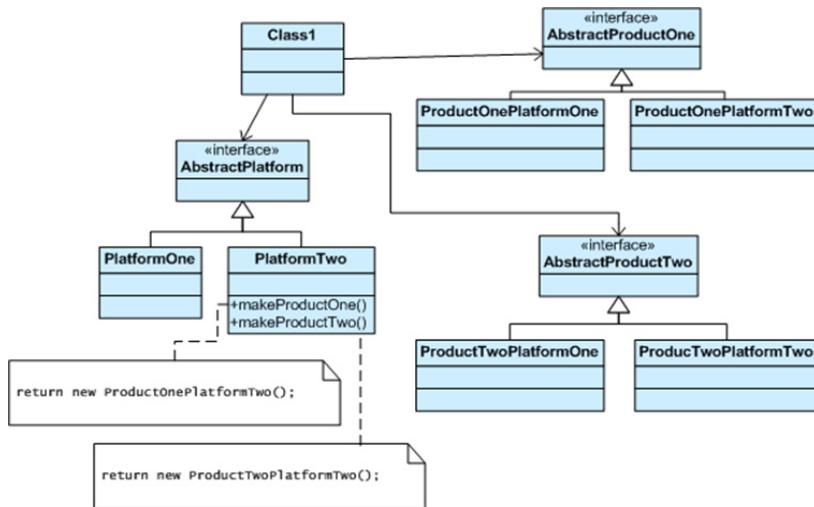


그림 1. Abstract Factory pattern 적용 시 모듈 분할 예시

표 17. 모듈 분할 및 책임 할당 목록

기호	모듈/ 컴포넌트	책임 할당
AD3	모듈/ 컴포넌트 1	QA1, HFR2, C1, FR1, FR2, BG1
AD1	모듈/ 컴포넌트 2	QA2, FR3,, FR5 BG4, C3
AD2	모듈/ 컴포넌트 3	QA3, BG1, FR4, HFR1, C3
	모듈/ 컴포넌트 4	QA4, BG2, HFR3, C2, FR6, FR7
	:	QA5, BG3, HFR4, FR8

바. 아키텍처 뷰 작성 (1차)

기본 설계 구조가 완성되어 실체화 되어있는 아키텍처를 여러 가지 뷰로 표현한다. 모듈 뷰는 런타임(runtime)이 아닐 경우 시스템의 특성과 아키텍처 결정사항을 표현하기에 적합하고, C&C 뷰는 런타임시 시스템의 특성과 아키텍처 결정사항을 표현하기에 적합하며, 할당 뷰는 SW과 비SW의 상호관계를 표현하기에 적합하다. 세 가지의 관점에서 표현하기 어려울 경우 결합 뷰를 사용하여 표현할 수 있다. 이러한 여러 가지 관점의 뷰를 바탕으로 이해관계자의 궁금증을 만족 시킬 수 있는 설계 뷰를 작성한다. 만약, 작성해야하는 범위가 작거나 새롭게 작성되는 부분이 적을 경우 모듈 분할 및 책임 할당 활동에서 수행하는 것을 권장한다. [아래 표 참고]

- * 각 반복 수행 시 모두 작성할지 최종 반복 이후 해당 내용을 전부 작성할지 결정한다. 높은 품질이 필요 할 경우 모든 반복 절차에서 작성하는 것을 추천한다.

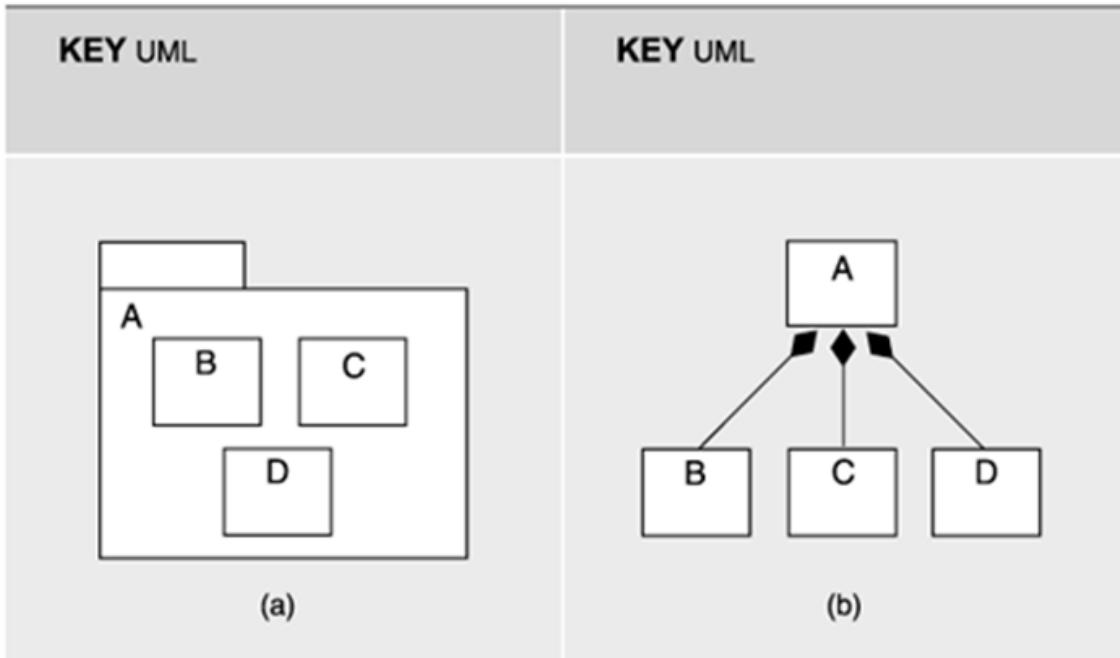
표 18. 뷰 정리 우선순위 [A: 매우상세, B: 약간 상세, C: 개략적]

이해관계자 집단	모듈 뷰			C&C 뷰		할당 뷰			결합 뷰
	분할	사용	계층	프로세스 통신	공유 데이터	배치	구현	작업 할당	:
프로젝트 관리자	B	B	B	-	C	A	C	A	:
개발팀원	A	A	A	A	A	B	B	A	:
유지보수자	A	A	A	A	A	B	B	B	:
테스터	-	A	-	A	B	B	A	-	:
시스템 통합	-	A	-	A	B	B	A	-	:
분석가	A	A	A	A	B	A	C	C	:
아키텍트	A	A	A	A	A	A	B	B	:
:									:

- 1) 모듈 뷰: 개발팀원, 테스터, 시스템 통합담당자, 유지보수자, 분석가, 현임/후임 아키텍트

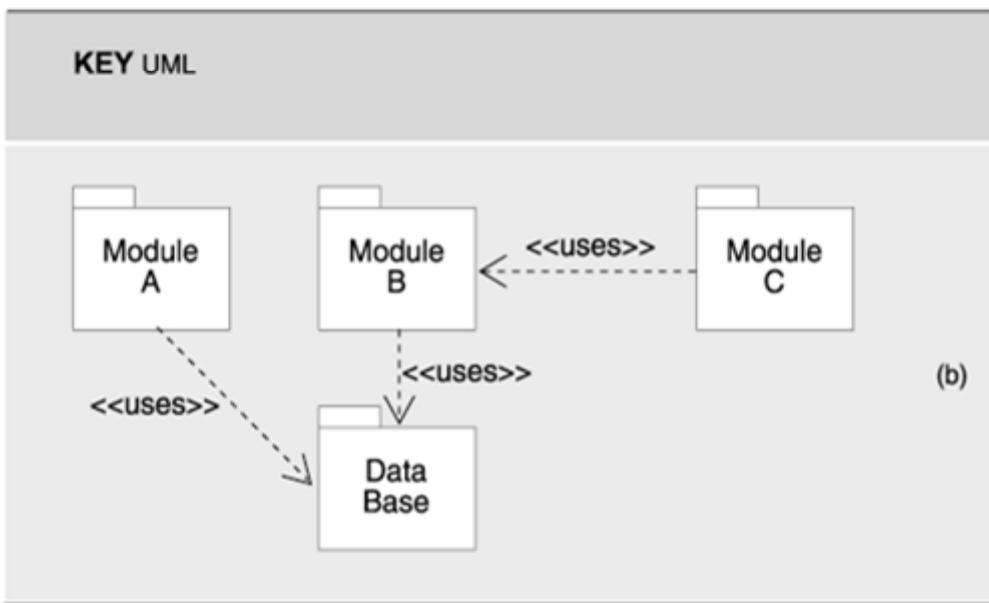
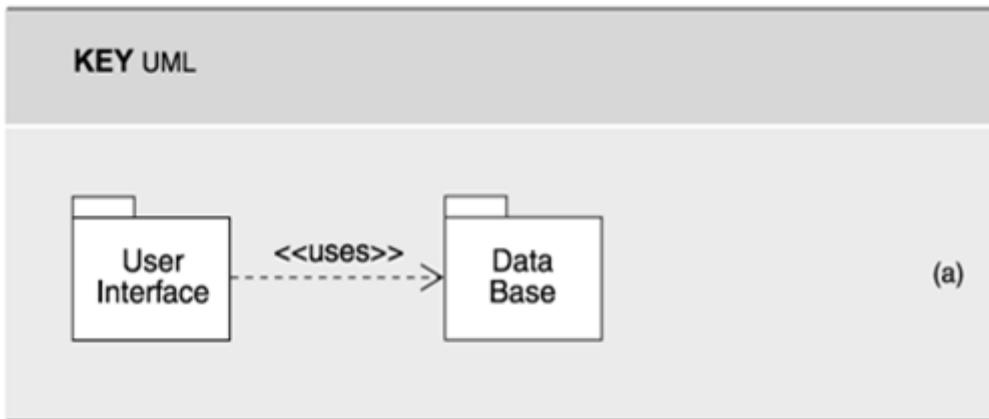
모듈은 시스템의 주요한 구현 단위(implementation unit)이며, 각 모듈들은 기능적 책임을 갖는다. 분할(Decomposition), 사용(Uses), 클래스(Class), 일반화(Generalization), 계층(Layered) 등

가) 분할 (Decomposition)



- (1) 동작(operations)의 문법적 설명
- (2) 동작(operations)의 의미적 설명
- (3) 변경되는 정보(Data)
- (4) 각각의 구성요소에 관계되는 품질 요구사항(관련 요소)
- (5) 예러 핸들링(handling)
- (6) 정보의 생산자와 소비자(데이터 흐름)
- (7) 모듈이 서비스를 제공하거나 사용할 때 필요한 상호작용
- (8) 서비스를 제공하거나 제공받는 모듈의 인터페이스와 연결된 스레드간의 상호 작용
- (9) 활성 컴포넌트에 대한 정보
- (10) 컴포넌트가 호출을 통하여 동기화되는지, 순차화하는지, 차단하고 있는지에 대한 정보
- (11) 특수 목적 하드웨어 등의 하드웨어 요구사항 등

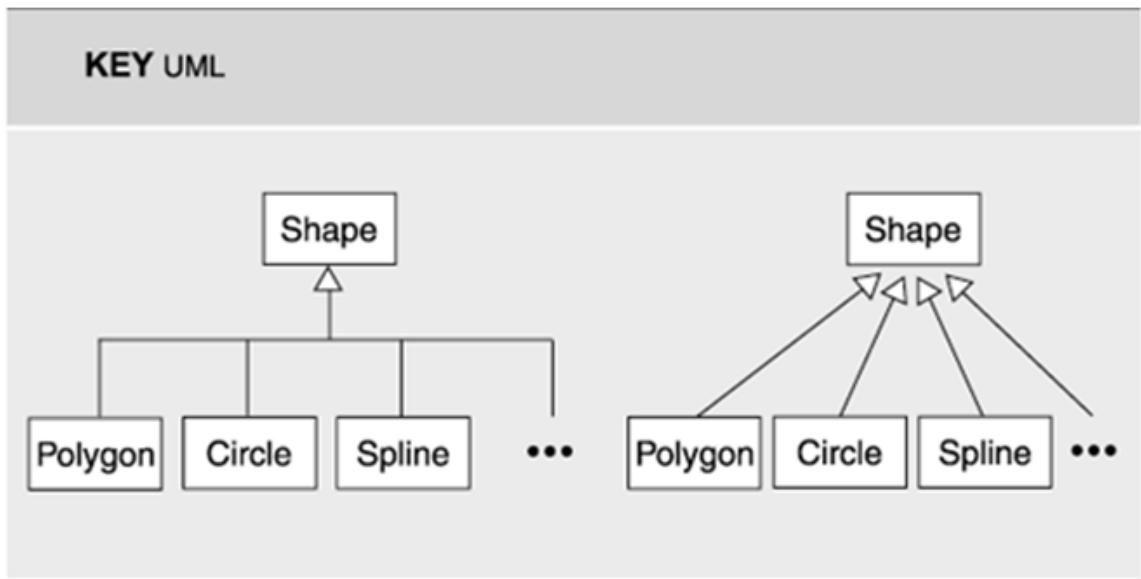
나) 사용(Uses)



- (1) 동작(operations)의 문법적 설명
- (2) 동작(operations)의 의미적 설명
- (3) 변경되는 정보(Data)
- (4) 각각의 구성요소에 관계되는 품질 요구사항(관련 요소)
- (5) 에러 핸들링(handling)
- (6) 정보의 생산자와 소비자(데이터 흐름)

- (7) 모듈이 서비스를 제공하거나 사용할 때 필요한 상호작용
- (8) 서비스를 제공하거나 제공받는 모듈의 인터페이스와 연결된 스레드간의 상호작용
- (9) 활성 컴포넌트에 대한 정보
- (10) 컴포넌트가 호출을 통하여 동기화되는지, 순차화하는지, 차단하고 있는지에 대한 정보
- (11) 특수 목적 하드웨어 등의 하드웨어 요구사항 등

다) 일반화(Generalization)

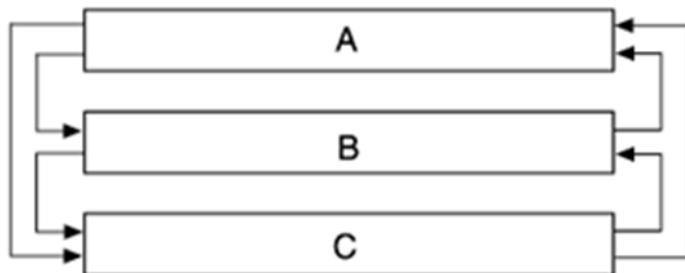


- (1) 동작(operations)의 문법적 설명
- (2) 동작(operations)의 의미적 설명
- (3) 변경되는 정보(Data)
- (4) 각각의 구성요소에 관계되는 품질 요구사항(관련 요소)
- (5) 에러 핸들링(handling)
- (6) 정보의 생산자와 소비자(데이터 흐름)
- (7) 모듈이 서비스를 제공하거나 사용할 때 필요한 상호작용

- (8) 서비스를 제공하거나 제공받는 모듈의 인터페이스와 연결된 스레드간의 상호작용
- (9) 활성 컴포넌트에 대한 정보
- (10) 컴포넌트가 호출을 통하여 동기화되는지, 순차화하는지, 차단하고 있는지에 대한 정보
- (11) 특수 목적 하드웨어 등의 하드웨어 요구사항 등

라) 계층(Layered)

KEY (informal notation)  Layer $x \rightarrow y$ x is allowed to use y



- (1) 동작(operations)의 문법적 설명
- (2) 동작(operations)의 의미적 설명
- (3) 변경되는 정보(Data)
- (4) 각각의 구성요소에 관계되는 품질 요구사항(관련 요소)
- (5) 에러 핸들링(handling)
- (6) 정보의 생산자와 소비자(데이터 흐름)
- (7) 모듈이 서비스를 제공하거나 사용할 때 필요한 상호작용
- (8) 서비스를 제공하거나 제공받는 모듈의 인터페이스와 연결된 스레드간의 상호작용
- (9) 활성 컴포넌트에 대한 정보
- (10) 컴포넌트가 호출을 통하여 동기화되는지, 순차화하는지, 차단하고 있는지에

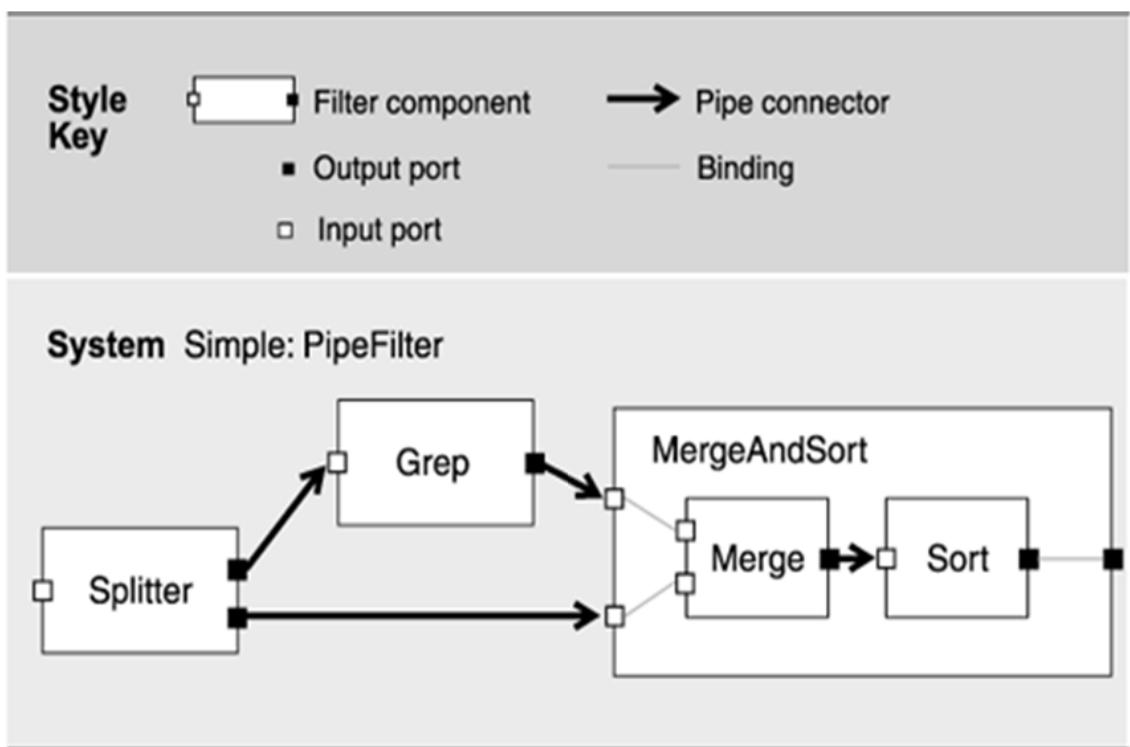
대한 정보

(11) 특수 목적 하드웨어 등의 하드웨어 요구사항 등

2) C&C 뷰: 개발팀원, 유지보수자, 현임/후임 아키텍트

런타임 컴포넌트와 커넥터로 시스템의 실행단위를 기술한다. *Pipe-and-Filter*, 공유 데이터(Shared Data), Client-Server, Peer-to-Peer, 프로세스간 통신(Communicating-Processes) 등

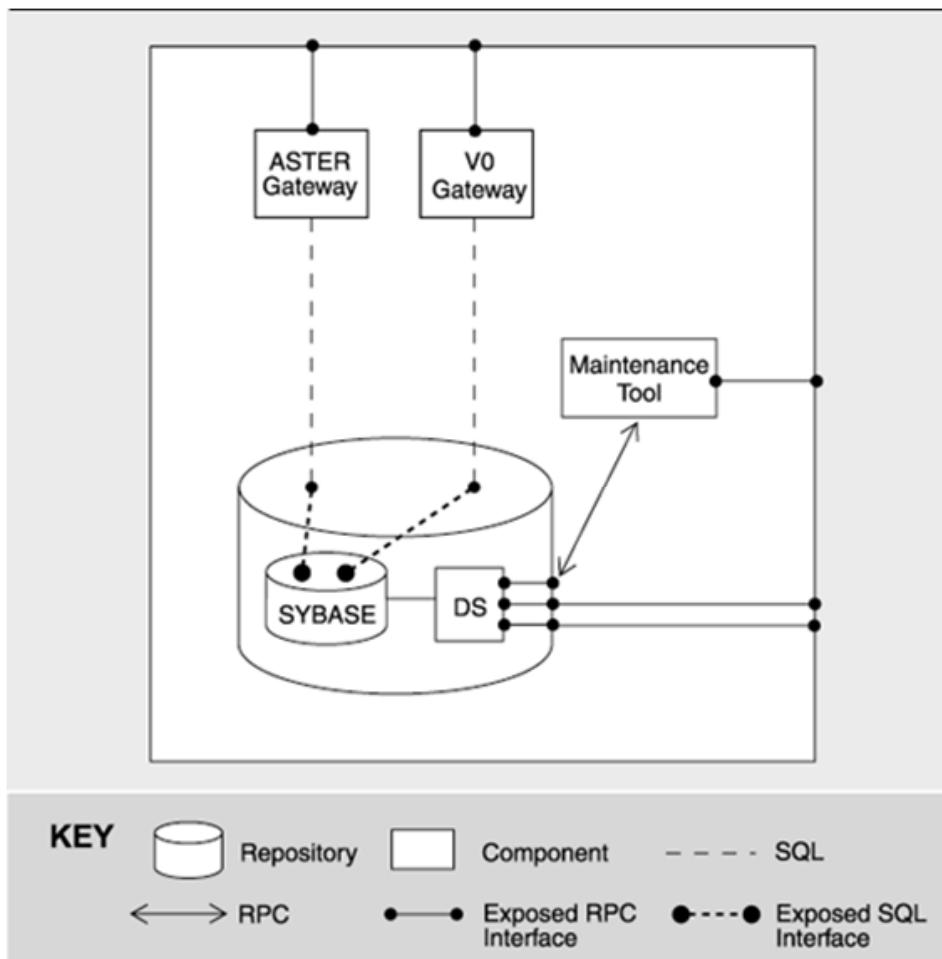
가) Pipe-and-Filter



- (1) 동작(operations)의 문법적 설명
- (2) 동작(operations)의 의미적 설명
- (3) 변경되는 정보(Data)
- (4) 각각의 구성요소에 관계되는 품질 요구사항(관련 요소)
- (5) 에러 핸들링(handling)

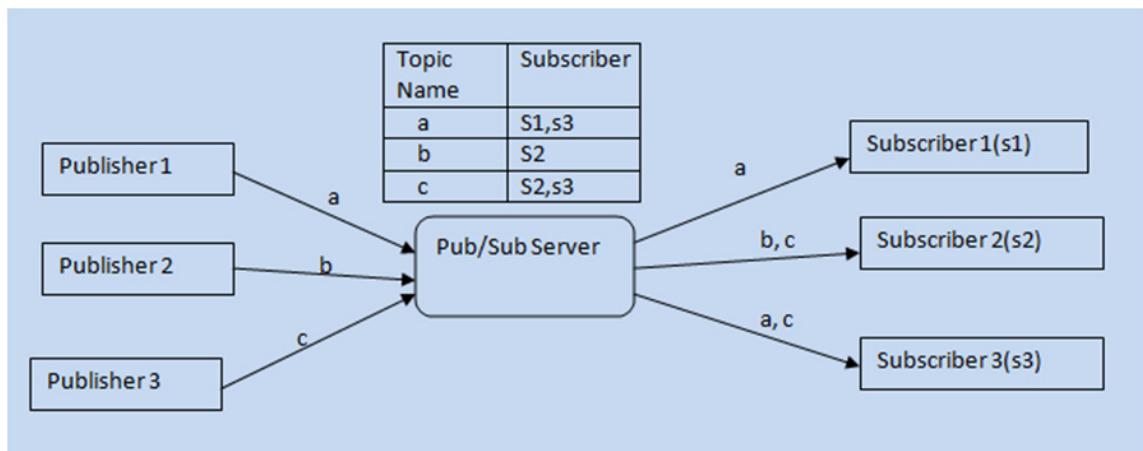
- (6) 정보의 생산자와 소비자(데이터 흐름)
- (7) 모듈이 서비스를 제공하거나 사용할 때 필요한 상호작용
- (8) 서비스를 제공하거나 제공받는 모듈의 인터페이스와 연결된 스레드간의 상호작용
- (9) 활성 컴포넌트에 대한 정보
- (10) 컴포넌트가 호출을 통하여 동기화되는지, 순차화하는지, 차단하고 있는지에 대한 정보
- (11) 특수 목적 하드웨어 등의 하드웨어 요구사항 등

나) 공유 데이터(Shared Data)



- (1) 동작(operations)의 문법적 설명
- (2) 동작(operations)의 의미적 설명
- (3) 변경되는 정보(Data)
- (4) 각각의 구성요소에 관계되는 품질 요구사항(관련 요소)
- (5) 에러 핸들링(handling)
- (6) 정보의 생산자와 소비자(데이터 흐름)
- (7) 모듈이 서비스를 제공하거나 사용할 때 필요한 상호작용
- (8) 서비스를 제공하거나 제공받는 모듈의 인터페이스와 연결된 스레드간의 상호작용
- (9) 활성 컴포넌트에 대한 정보
- (10) 컴포넌트가 호출을 통하여 동기화되는지, 순차화하는지, 차단하고 있는지에 대한 정보
- (11) 특수 목적 하드웨어 등의 하드웨어 요구사항 등

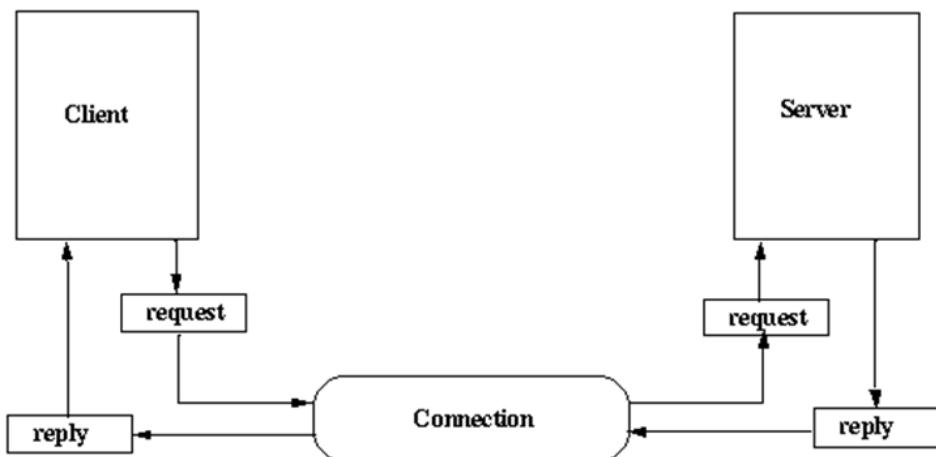
다) 발행-구독 (Publish-subscribe)



- (1) 동작(operations)의 문법적 설명
- (2) 동작(operations)의 의미적 설명
- (3) 변경되는 정보(Data)
- (4) 각각의 구성요소에 관계되는 품질 요구사항(관련 요소)
- (5) 에러 핸들링(handling)

- (6) 정보의 생산자와 소비자(데이터 흐름)
- (7) 모듈이 서비스를 제공하거나 사용할 때 필요한 상호작용
- (8) 서비스를 제공하거나 제공받는 모듈의 인터페이스와 연결된 스레드간의 상호작용
- (9) 활성 컴포넌트에 대한 정보
- (10) 컴포넌트가 호출을 통하여 동기화되는지, 순차화하는지, 차단하고 있는지에 대한 정보
- (11) 특수 목적 하드웨어 등의 하드웨어 요구사항 등

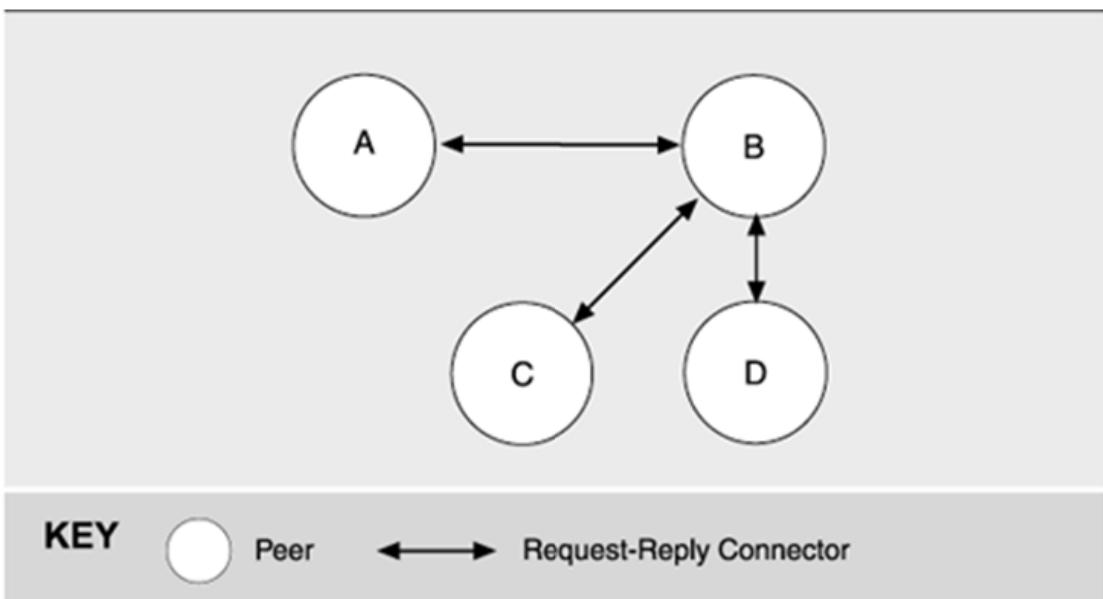
라) Client–Server



- (1) 동작(operations)의 문법적 설명
- (2) 동작(operations)의 의미적 설명
- (3) 변경되는 정보(Data)
- (4) 각각의 구성요소에 관계되는 품질 요구사항(관련 요소)
- (5) 에러 핸들링(error handling)
- (6) 정보의 생산자와 소비자(데이터 흐름)
- (7) 모듈이 서비스를 제공하거나 사용할 때 필요한 상호작용

- (8) 서비스를 제공하거나 제공받는 모듈의 인터페이스와 연결된 스레드간의 상호작용
- (9) 활성 컴포넌트에 대한 정보
- (10) 컴포넌트가 호출을 통하여 동기화되는지, 순차화하는지, 차단하고 있는지에 대한 정보
- (11) 특수 목적 하드웨어 등의 하드웨어 요구사항 등

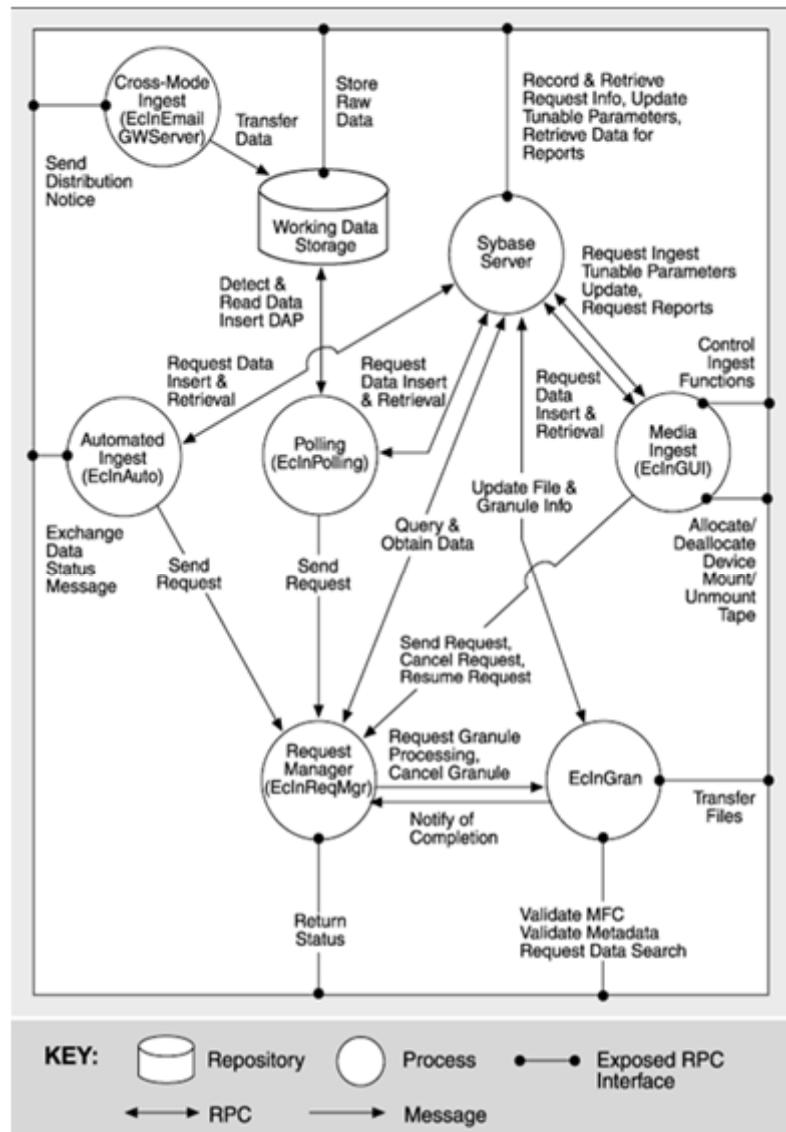
마) Peer-to-Peer



- (1) 동작(operations)의 문법적 설명
- (2) 동작(operations)의 의미적 설명
- (3) 변경되는 정보(Data)
- (4) 각각의 구성요소에 관계되는 품질 요구사항(관련 요소)
- (5) 에러 핸들링(handling)
- (6) 정보의 생산자와 소비자(데이터 흐름)
- (7) 모듈이 서비스를 제공하거나 사용할 때 필요한 상호작용
- (8) 서비스를 제공하거나 제공받는 모듈의 인터페이스와 연결된 스레드간의 상호작용

- (8) 활성 컴포넌트에 대한 정보
- (9) 컴포넌트가 호출을 통하여 동기화되는지, 순차화하는지, 차단하고 있는지에 대한 정보
- (10) 특수 목적 하드웨어 등의 하드웨어 요구사항 등

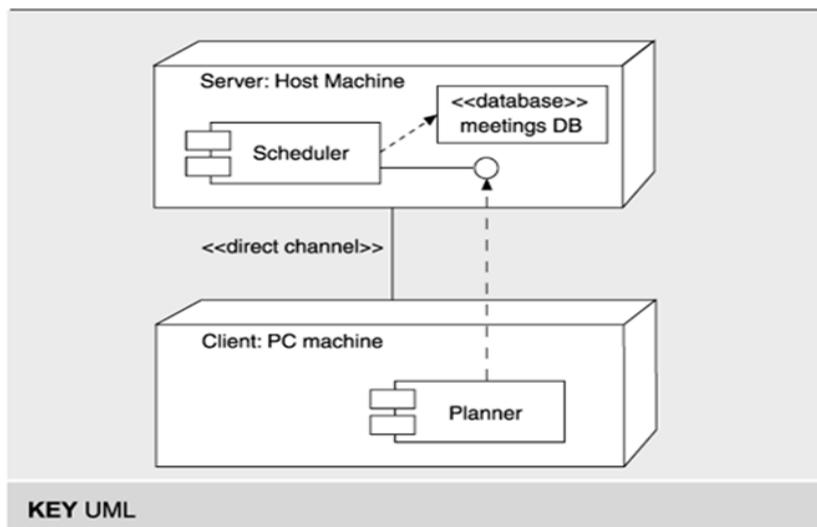
바) 프로세스간 통신(Communicating-Processes)



- (1) 동작(operations)의 문법적 설명
- (2) 동작(operations)의 의미적 설명
- (3) 변경되는 정보(Data)
- (4) 각각의 구성요소에 관계되는 품질 요구사항(관련 요소)
- (5) 에러 핸들링(handling)
- (6) 정보의 생산자와 소비자(데이터 흐름)
- (7) 모듈이 서비스를 제공하거나 사용할 때 필요한 상호작용
- (8) 서비스를 제공하거나 제공받는 모듈의 인터페이스와 연결된 스레드간의 상호작용
- (9) 활성 컴포넌트에 대한 정보
- (10) 컴포넌트가 호출을 통하여 동기화되는지, 순차화하는지, 차단하고 있는지에 대한 정보
- (11) 특수 목적 하드웨어 등의 하드웨어 요구사항 등

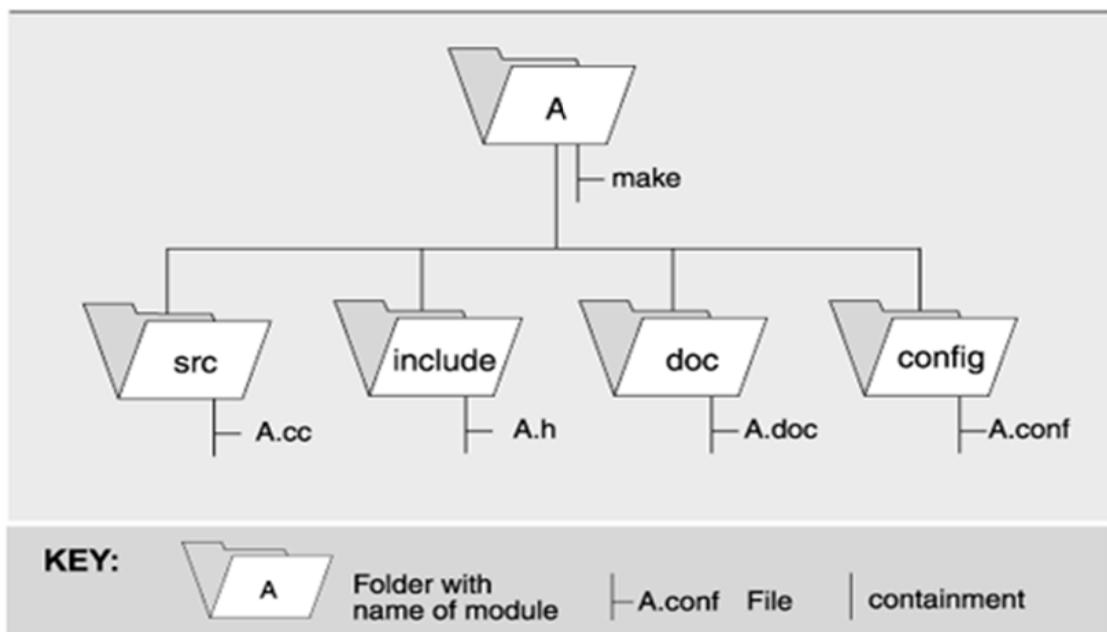
3) 할당 뷰: 프로젝트 관리자, 분석가, 기반 구조 지원 담당자, 현임/후임 아키텍트 시스템의 소프트웨어 구성요소와 소프트웨어가 생성되고 실행되는 외부 환경사이의 관계를 기술한다. 배치(deployment), 구현(Implementation), 작업 할당(Work assignment) 등

가) 배치(deployment)



- (1) 동작(operations)의 문법적 설명
- (2) 동작(operations)의 의미적 설명
- (3) 변경되는 정보(Data)
- (4) 각각의 구성요소에 관계되는 품질 요구사항(관련 요소)
- (5) 정보의 생산자와 소비자(데이터 흐름)
- (6) 모듈이 서비스를 제공하거나 사용할 때 필요한 상호작용
- (7) 서비스를 제공하거나 제공받는 모듈의 인터페이스와 연결된 스레드간의 상호작용
- (8) 활성 컴포넌트에 대한 정보
- (9) 컴포넌트가 호출을 통하여 동기화되는지, 순차화하는지, 차단하고 있는지에 대한 정보
- (10) 특수 목적 하드웨어 등의 하드웨어 요구사항 등

나) 구현(Implementation)



- (1) 해당 내용 작성

다) 작업 할당(Work assignment)

ECS Element (Module)		
Segment	Subsystem	Organizational Unit
Science Data Processing Segment (SDPS)	Client	Science team
	Interoperability	Prime contractor team 1
	Ingest	Prime contractor team 2
	Data Management	Data team
	Data Processing	Data team
	Data Server	Data team
	Planning	Orbital vehicle team
Communications and System Management Segment (CSMS)	System Management	Infrastructure team
	Communications	
	Internetworking	
Flight Operations Segment (FOS)	Planning and Scheduling	Orbital vehicle team
	Data Management	Database team
	Command Management	Orbital vehicle team
	Commanding	Orbital vehicle team
	Resource Management	Prime contractor team 3
	Telemetry	Orbital vehicle team
	User Interface	User interface team
	Analysis	Orbital vehicle team

(1) 해당 내용 작성

- 4) 결합 뷰: 개발팀원, 테스터, 시스템 통합담당자, 유지보수자, 분석가, 협임/후임 아키텍트

혼성(hybrid), 중첩(overlay)등의 방법을 활용하며 위의 뷰 형식에 따라 작성한다.

사. 아키텍쳐 뷰 작성 (2차, 3차, 4차 ...)

반복 작업을 수행함에 따라 아래의 내용을 상기의 양식을 따라 반복적으로 작성한다.

- 1) 아키텍처 드라이버 식별
- 2) 아키텍처 패턴 및 설계전술(tactic) 선정
- 3) 모듈 분할 및 책임 할당
- 4) 아키텍처 뷰 작성

3. 아키텍처 검증

가. 아키텍처 검증 절차(수정된)

아키텍처 검증 과정의 배경과 목적, 수행 절차, 담당자별 역할, 활동별 수행 내용 등에 대한 내용을 소개한다.

1) 검증 절차

수정 및 확정된 검증 절차

2) 담당자별 역할

담당자별 역할 정의

3) 입력/출력물 정의

수정 및 확정된 입력/출력물 정의

4) 절차별 수행 방법

수정 및 확정된 아키텍처 검증 절차 수행 방법

나. 비즈니스/아키텍처 목표 소개

표 19. 비즈니스/아키텍처 목표 소개 자료 항목

비즈니스/아키텍처 목표 소개 자료 항목
•사업 환경과 역사, 시장 차별화, 핵심 요구사항, 이해관계자, 현재 필요사항과 평가대상 시스템이 이러한 필요/요구사항을 어떻게 만족시키는지에 대한 설명
•비즈니스 제약사항에 대한 설명(예: 적시출시, 고객 요구, 표준, 비용 등)
•기술적 제약사항에 대한 설명(예: COTS(commercial off-the-shelf) 제품, 타 시스템과의 연동, 지정된 HW 또는 SW 플랫폼, 레거시 코드의 재사용 등)
•품질속성 요구사항과 어떤 비즈니스의 필요에 의해 해당 품질속성이 도출됐는지에 대한 설명
•용어설명

다. 작성된 아키텍처 소개

표 20. 작성된 아키텍처 소개 항목

작성된 아키텍처 소개 항목
• 아키텍처 요구사항, 요구사항과 관련된 측정 가능한 수치와 이를 달성하는 데 필요한 기준 표준/모델/접근방법
• 상위 수준 아키텍처 뷰
• <i>Module View</i> <ul style="list-style-type: none"> - 모듈은 시스템의 주요한 구현 단위(implementation unit)이며, 각 모듈들은 기능적 책임을 갖는다. - 분할(Decomposition), 사용(Uses), 레이어(Layered), 클래스(Class) 또는 일반화(Generalization)
• <i>Component and Connector View</i> <p>런타임 컴포넌트와 커넥터로 시스템의 실행단위를 기술한다.</p> <ul style="list-style-type: none"> - <i>Pipe-and-Filter</i>: 데이터 스트림을 연속적으로 변환한다. - 공유 데이터(Shared Data, or repository): 이 구조는 영구적인 데이터를 생성, 저장, 접근하는 컴포넌트와 커넥터로 구성된다. - 발행 구독(Publish-Subscribe): 이벤트의 공표를 통하여 컴포넌트들이 상호작용한다. - <i>Client-Server</i> : 이 구조에서는 클라이언트와 서버가 컴포넌트이며, 프로토콜과 메시지가 커넥터가 된다. - <i>Peer-to-Peer</i>: 컴포넌트들이 동등한 입장에서 서비스를 교환하며 상호작용한다. - 프로세스 통신(Communicating processes): 모든 C&C구조와 마찬가지로, 이 구조도 모듈 기반 구조와 orthogonal하며 구동(running) 시스템의 동적인 면을 다룬다. 이 구조에서는 프로세스나 쓰레드가 구성 유닛이 된다.
• <i>Allocation View</i> <p>시스템의 소프트웨어 구성요소와 소프트웨어가 생성되고 실행되는 외부 환경사이의 관계를 기술한다.</p> <ul style="list-style-type: none"> - 배치(Deployment) : 배치 구조는 소프트웨어가 하드웨어와 통신 요소에 할당되는 내용을 나타낸다. 이 구조의 요소는 소프트웨어(주로 C&C 뷰에서의 프로세스), 하드웨어

작성된 아키텍처 소개 항목	
어(프로세서), 통신 경로등이다.	
- 구현(Implementation) : 이 구조는 소프트웨어 요소(주로 모듈)와 시스템 개발, 통합, 형상관리 환경에서 파일 구조와의 맵핑관계를 나타낸다.	
- 작업 할당(Work assignment): 이 구조는 모듈의 구현 및 통합에 대한 책임을 적절한 개발 팀에 할당한다.	
• 달성하고자 한 품질속성에 따라 적용한 아키텍처 접근방법, 스타일, 메커니즘에 대한 설명	
• 아키텍처에 사용된 COTS(commercial off-the-shelf) 종류와 선정 사유, 타 구성요소와의 연계에 대한 설명	
• 가장 중요한 1~3개 정도의 유스케이스 시나리오 추적을 통해 각 시나리오에서 사용하는 실행시간 자원에 대한 설명	
• 가장 중요한 1~3개 정도의 확장 시나리오(growth scenario) 추적을 통해 변경될 컴포넌트 또는 커넥터, 인터페이스 측면에서의 변경 영향(예측된 변경의 규모/어려움)을 설명	
• 아키텍처 요구사항을 만족시키는 데 따르는 아키텍처 이슈/위험요소 설명	
• 용어설명	

라. 유틸리티 트리

표 21. 우선순위화된 유틸리티 트리 목록

품질 속성	속성 정제	기호	시나리오	중요도	구현성	합계
유지보수성 (Maintainability)	변경의 지역화	M1	• 하나의 서브시스템에 대한 변경이 다른 시스템에 대한 변경을 요구하지 않는다.			
	롤 백 시스템	M2	• 독립적으로 롤 백 서브시스템 배치			
	시험 기간 감소	M3	• 복귀(regression) 시험 기간을 5일에서 1일로 감소			
	업그레이드 강화	M4	• 운영체제 데이터베이스, COTS 제품의 관리 업그레이드 시간을 감소			

품질 속성	속성 정제	기호	시나리오	중요도	구현성	합계
운영성 (Operability)	우선순위 재투표	O10	<ul style="list-style-type: none"> 시스템은 사용자 클래서, 데이터 타입, 미디어 타입, 수신자 또는 사용자에 의해 20분안에 1000개의 우선순위를 재결정 할 수 있어야 한다. 			
	동시 처리 강화	O14	<ul style="list-style-type: none"> 시스템은 V0게이트웨이나 MTM 게이트웨이를 통해 운영의 간섭 없이 1000개의 동시 요구를 서비스 할 수 있어야 한다. 			
신뢰성 (Reliability)	자원 제한	R1	<ul style="list-style-type: none"> 실패하거나 10분 이상 지연되는 데이터 입력과 출력에 의해 유지되는 자원이 없다. 			
	자원 확보	R2	<ul style="list-style-type: none"> 요구의 한 부분에 대한 데이터에러는 다른 부분의 수행을 방해 해서는 안된다. 			
	데이터 보관	R6	<ul style="list-style-type: none"> 시스템 오버로드나 고장의 결과로 써 손실되는 요구가 없어야 한다. 			
확장성 (Scalability)	지원 용량 확대	S2	<ul style="list-style-type: none"> 시스템은 50개의 사이트를 지원 할 수 있어야 한다. 			
	많은 데이터 소스 지원	S3	<ul style="list-style-type: none"> 시스템은 100개의 데이터 소스로부터의 수집을 지원할 수 있어야 한다. 			
	전기 분포 지원	S4	<ul style="list-style-type: none"> 시스템은 2000개의 사이트에 전기 분포를 지원할 수 있어야 한다. 			
성능 (Performance)	응답시간 향상	Pq	<ul style="list-style-type: none"> Landsat L-7 탐색을 위한 탐색 응답시간의 5배 향상 			

마. 아키텍처 접근법 분석서

표 22. 아키텍처 접근법 분석서

아키텍처 접근법 분석서						
시나리오 번호 (Scenario #)	번호	시나리오 (Scenario)	유ти리티 트리에 나온 시나리오 문장			
품질속성 (Attributes)	시나리오와 관련된 품질속성					
환경 (Environment)	시나리오가 실행되는 조건과 시스템이 동작하는 환경에 대한 가정					
자극 (Stimuli)	시나리오가 시작되는 자극에 대한 정확한 설명(예: 험수호출, 시스템 장애, ...)					
대응 (Response)	자극에 대한 시스템의 정확한 반응(예: 응답시간, 수정 난이도, ...)					
아키텍처 판단 (Architectural Decision)	민감점 (Sensitivity)	절충점 (Trade-off)	위험 (Risk)	무위험 (Nonrisk)		
시나리오로 표현한 품질속성에 해당하는 시스템 반응에 영향을 미치는 아키텍처 판단들	민감점 번호	절충점 번호	위험 번호	무위험 번호		
...		
...		
근거 (Reasoning)	앞에 나온 목록에 등장하는 아키텍처 판단들이 시나리오가 표현하는 품질속성을 달성하는 데 공헌한다는 근거					
아키텍처 다이어그램 (Architectural Diagram)	앞에 나온 근거를 뒷받침할 수 있는 다이어그램					

바. 결과 정리

표 23. 결과 보고서 주요 항목

결과보고 아웃라인	설 명
설계 검증 단계 요약	9개 활동에 대한 요약
비즈니스/아키텍처 드라이버	비즈니스/아키텍처 드라이버 목표 소개 활동 산출물 요약
아키텍처 접근법	개정된 아키텍처 접근법 보고서 요약
유ти리티 트리	개정된 유ти리티 트리 요약
시나리오	브레인스토밍과 우선순위가 결정된 시나리오 목록
시나리오 토론 내용	개정된 유ти리티 트리에서 관련된 분석 내용 요약
위험요소, 비위험요소, 민감요소, 트레이드오프	검증 단계 수행 중에 발견된 모든 위험요소, 비위험요소, 민감점, 절충점의 내용 정리
기타 이슈	검증 단계 수행 중에 다루지 못한 기타 이슈들의 정리
결론 정리	검증팀에 의해 식별된 위험요소 요약 및 위험요소가 어떻게 품질속성의 달성을 위협이 되는지 설명

SW개발 품질관리 매뉴얼 (SW Visualization)

공동집필

이상은, 정보통신산업진흥원 부설 SW공학센터
백성현, 정보통신산업진흥원 부설 SW공학센터
강승준, 정보통신산업진흥원 부설 SW공학센터
송원민, 정보통신산업진흥원 부설 SW공학센터
이성은, 정보통신산업진흥원 부설 SW공학센터
장원석, 정보통신산업진흥원 부설 SW공학센터
정도균, 정보통신산업진흥원 부설 SW공학센터
조지만, 정보통신산업진흥원 부설 SW공학센터

주관/후원

nipa 정보통신산업진흥원

발행일

2013년 12월 20일