공 학 석 사 학 위 논 문

스크럼 기반의 프로젝트 관리도구 설계 및 구현

2011년 2월

부 경 대 학 교 산 업 대 학 원

컴 퓨 터 공 학 과

김 태 형

공 학 석 사 학 위 논 문

스크럼 기반의 프로젝트 관리도구 설계 및 구현

지도교수 송 하 주

이 論文을 工學碩士 學位論文으로 提出함

2011년 2월

부 경 대 학 교 산 업 대 학 원

컴 퓨 터 공 학 과

김 태 형

이 논문을 김태형의 공학석사

학위논문으로 인준함

2010 년 12 월 15 일

주 심 공학박사 정 목 동 (인)

위 원 공학박사 조 우 현 (인)

위 원 공학박사 송 하 주 (인)

목 차

그림 목차

표 목차

스크럼 기반의 프로젝트 관리도구 설계 및 구현

김 태 형

부 경 대 학 교 산 업 대 학 원 컴 퓨 터 공 학 과

요약

스크럼은 복잡한 프로젝트를 관리하기 위한 아주 단순한 프레임워크로 이미 실무의 여러 조직에서 스크럼을 도입하여 성공적인 사례들이 소개되고 있다. 하지만 단순한 스크럼의 실천 방법에 비하여 도입하기는 쉽지 않은 것으로 알려져 있다. 이에 본 연구에서는 스크럼의 큰 틀을 도구화하여 조직에 도입되었을 때 스크럼 마스터의 역할을 일부 대신해 줄 수 있도록 하며, 제품 백로그 작성 및 스프린트 단계의 구체적 내용들을 툴을 활용하여 진행 및 관리 함으로서 스크럼에 대한 이해가 낮은 경우에도 보다 쉽고 정확하게 도입하여 실천할 수 있을 것이라고 기대하게 되어 이러한 도구를 설계하게 되었다.

Design and Implementation of A Scrum-based Project Management System

Tae Hyeong Kim

Department of Computer Engineering, Graduate School,

Pukyong National University

Abstract

Scrum to manage complex projects is a very simple framework. Successful cases have already been introduced by many organizations in the Scrum. But doing Scrum practice is known to be difficult. In this study, using the tool is used instead of some of the Scrum Master's role. In this way, more easily and precisely would be able to practice to expectations is that's why this tool was designed.

1. 서론

1.1. 연구의 필요성

오늘날 IT, Game, SI분야 뿐만 아니라 전자, 금융, 자동차 등 거의 모든 산업 분야에서 S/W의 비중은 급격히 커지고 있으며, 그 속도도 점점 빨라지고 있다. 이에 S/W개발의 생산성과 품질을 향상시키기 위한 노력으로 70년대부터 S/W공학이 발전하여 왔다. 그럼에도 불구하고 여전히 S/W개발 프로젝트는 매우 힘들게 진행되는 경우가 많으며, 상당수의 프로젝트가 중간에 중단되고 있고, 절반 이상의 프로젝트가 납기와 예산을 초과하고 있으며, PL은 항상 바쁘게 뛰어다니고, 개발자는 야근 특근을 밥 먹듯 하지만, 관리자는 프로젝트가 계획대비 진행률이 어느 정도 인지 파악조차 되지 않아서 불안해 하고 있다. 2000년대에 와서는 애자일 방법론이라는 새로운 패러다임이 등장하는데 이것은 다른 무엇보다도 S/W를 낭비 없이 빠르게 만드는 것이 가장 중요하다고 믿는 사람들이 커뮤니티를 만들어 활동하면서 발전하였다. 애자일 패러다임에서는 S/W개발자들 간의 상호작용, 작동하는 S/W, 고객과의 협력, 변화에의 대응 이 4가지를 주요가치로 삼고 있으며 이러한 애자일 패러다임이 적용된 방법론 중 하나가 바로 스크럼(Scrum)이다.[1]

스크럼은 복잡한 프로젝트를 관리하기 위한 아주 단순한 프레임워크로, 고객이 원하는 하나의 프로젝트를 2~4주 단위의 스프린트로 계획하여 매 스프린트 마다 출시 가능한 S/W결과물을 산출하여 마지막에 높은 품질과 고객만족도의 최종 결과물을 만들어 내게 한다. 이미 실무의 여러 조직에서 스크럼을 도입하여 성공적인 사례들이 소개되고 있다. 이에 현업에서는 스크럼이 통하는 방법론으로 인식되고 있다.[2]

스크럼의 실천 방법 자체는 비교적 단순하지만 도입하기는 쉽지 않은 것으로 알려져 있다. 그 이유는 우선 스크럼을 도입하기 위해서는 전체 조직원들이 스크럼에 대해서 어느 정도 이해하고 있어야 하며, 특별히 스크럼에 대해서 잘 알고 있어 팀에 스크럼이 잘 적용되고 있는지를 항상 관찰하고 이를 어기게 되는 위협요소들을 차단해주는 키잡이 역할을 하는 스크럼 마스터가 있어야 하기

때문인데, 현재 S/W개발 기업은 25명 이하의 소규모 기업이 큰 비중을 차지하고 있으며, 이는 각 기업마다 스크럼마스터 역할을 할만한 인력을 확보하는 것이 부담이 될 수 있고 또한 업무를 진행하면서 조직원들에게 스크럼에 대한 별도의 교육을 하는 것도 쉽지 않은 상황임을 나타내고 있다.[3]

이에 본 연구에서는 스크럼의 큰 틀을 도구화하여 조직에 도입되었을 때 스크럼 마스터의 역할을 일부 대신해 줄 수 있도록 하며, 제품 백로그 작성 및 스프린트 단계의 구체적 내용들을 툴을 활용하여 진행 및 관리 함으로서 스크럼에 대한 이해가 낮은 경우에도 보다 쉽고 정확하게 도입하여 실천할 수 있을 것이라고 기대하게 되어 이러한 도구를 설계하게 되었다.

1.2. 연구의 목적 및 방법

본 연구는 스크럼을 도입하려는 다양한 규모의 프로젝트 및 조직이 보다 쉽게 스크럼을 실천할 수 있도록 돕는 도구를 개발하는 것이 목적이다. 그러기 위해서 기존 실천하고 있는 사례 중에서 모범적인 실천 법을 조사 및 분석하여 그 실천법을 최대한 도구화 하여 스크럼에 대해서 잘 모르는 조직이라 할지라도 도구를 통하여 스크럼에 대한 정보를 얻고 실천 법에 대한 기본 틀을 제공 받을 수 있도록 한다.

1.3. 논문의 구성

본 논문은 1장에서 논문의 목적과 필요성에 대해서 설명하고 2장에서는 스크럼 관련 연구 및 현재 스크럼의 실천 활동을 살펴보고 그 과정에서 나타나는 몇 가지 문제점들을 짚어 본다. 3장에서는 스크럼의 과정을 도구화 하기 위한 요구사항을 분석하여 Use Case Diagram으로 나타내고 있으며 4장에서는 도구의 설계에 대해서 기술한다. 설계는 전체 시스템의 개요를 보이고 데이터 설계 주요 모듈의 흐름 설계 프로토콜 설계 실제 UI의 구성 설계 순으로 소개한다. 5장에서는 도구가 설계에 따라 구현되었을 때 이전과 스크럼의 실천 활동상에 어떤 변화가 일어 나게 되는지를 살펴 보고 추가적으로 전체 조직과 프로젝트 진행에 미치게 될 효과를 평가한다.

마지막으로 6장에서 향후 추가되어야 할 연구과제에 대해서 기술한다. 부록 1에서는 4장에서 설계된 데이터의 실제 xml DTD와 각 element들의 의미를 소개하며, 부록 2에서는 역시 4장에서 소개된 프로토콜의 실제 정의를 상세히 소개한다.

2. 관련연구

2.1. Agile 및 Scrum과 관련 연구

소프트웨어 개발의 프로세스에 대한 연구는 소프트웨어의 위기 이후로 꾸준히 이루어 지고 있으며 Agile은 이 같은 연구 활동 과정에서 2000년 대 전후로 개발자들의 커뮤니티로부터 탄생하여 나타난 패러다임이다. Agile은 XP, TDD 와 같은 좋은 실천 법들을 탄생 시켰으며 요구사항에 보다 유연하게 대처하는 개발 방법론으로 확대되어 왔다. Scrum은 이 같은 Agile의 패러다임에 맞는 단순하면서도 구체적인 실천법들을 정의 해 놓은 개발 방법론이라 할 수 있다. 스크럼에는 제품 백로그, 스프린트 계획회의, 스프린트, 일일 스크럼회의, 소멸 차트, 스프린트 회고와 같은 활동이 정의 되어 있으나 이것들을 실무에서 실천하는 방법은 조직이나 S/W 프로젝트 특성에 따라 유연하게 변해야 한다. 이에 따라 최근 스크럼은 다양한 형태의 프로젝트에서 스크럼 프로세스의 실천 방법과 적용 사례를 연구하고 있다.[4][5] 산업 현장에서도 스크럼의 실천 방법에 대하여 실무자들간에 토론이 이루어지고 있으며 실제 프로젝트에 적용하고 있다.

2.2. 도구를 도입하지 않는 스크럼 실천 방법 및 문제점

현재는 스크럼에 대한 연구가 소규모 조직과 프로젝트를 위주로 진행되고 있는데 이것은 스크럼 자체가 스프린트라는 단기 목표에 중점을 두고 있으며, 조직원들간의 원활한 의사소통을 위해 물리적으로 근접한 위치에 있기를 권하고, 프로젝트 진행 상태를 나타내는 상황판을 벽면이나 화이트보드 및 포스트 잍 등으로 구성하는 실천 방법의 한계라고 볼 수 있다.

아래 그림 1은 현재 스크럼의 실천 요소들을 순서적으로 도식화 한 그림이다.

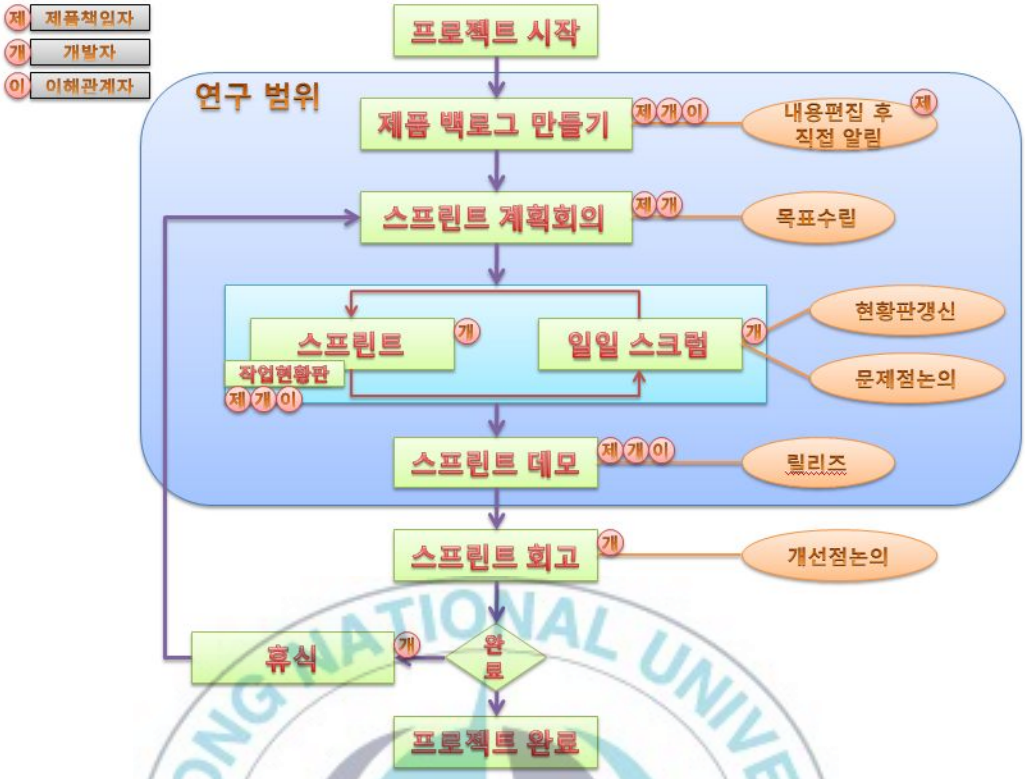

그림 1 현재 스크럼 활동 흐름

비록 지금은 스크럼을 실천하고 있는 많은 실무자들이 이와 같은 실천 방법을 선호하고 있으나, 물리적으로 떨어져 있는 조직과 다양한 프로젝트에 스크럼을 적용하기 위해서는 스크럼의 기본적인 실천 개념을 위배하지 않으면서 실천을 돕는 도구개발에 대한 연구가 필요하다. 최근에 대규모 프로젝트에서의 스크럼 실천 방법에 대한 연구가 이루어지고 있으며 스크럼 마스터 과정에서도 이와 같은 주제의 교육을 실시하고 있다.

스크럼의 흐름을 살펴 보면 먼저 프로젝트가 시작되고 고객의 요구사항인 제품 백로그를 만드는 행위가 일어난다. 제품 백로그는 개발자와 이해관계자, 고객이 모두 참여하여 함께 만들어 간다. 이 활동은 별도의 워크샵 등 다양한 방법으로 진행되며 이 결과로 고객이 만족할 만한 요구사항을 도출하고 이를 문서화 한 결과물을

얻게 된다. 애자일에서는 이 요구사항은 고객에 의해서 상시 변할 수 있는 것으로 고객은 제품 백로그를 변경 한 후 개발자와 이해관계자들에게 이를 알리고 설명할 의무를 갖는다. 이 활동을 가장 명확히 하기 위해서는 모든 개발자와 이해관계자를 한자리에 모이게 한 다음 실시해야 한다. 하지만 현실적으로 이렇게 하기에는 어려움이 있으며, 이로 인하여 요구사항의 내용이 잘못 이해되며 이는 프로젝트 실패의 가장 큰 요인이 된다.

스크럼에서 프로젝트의 진행은 스프린트라는 단위로 이루어 지며, 이 스프린트는 2주 정도의 기간 단위로 고객에게 출시 가능한 결과물을 얻을 수 있을 결과물을 얻어내는 것이 주 활동이고 이러한 스프린트를 반복적으로 수행하여 점진적으로 프로젝트를 완료한다. 이 스프린트에 포함될 제품 백로그를 결정하고 제품 백로그에 담당자를 임명하는 것이 스프린트 계획 회의의 주된 목적이다. 이 활동에는 개발자와 제품 책임자가 참여하고 제품 백로그 아이템의 중요도와 예상 작업 기간 등을 고려하여 스프린트에 포함할 것인지를 결정하고 스프린트 데모 시 어떤 절차로 데모해야 할지에 대한 구체적인 내용을 결정한다.

스프린트 계획이 수립되면 이제 스프린트르 시작하는데 이때 작업 현황판을 두어 매일 아침 일일 스크럼이라는 활동을 통하여 이 현황판을 개발자들이 직접 수정하고 전일 작업에 대한 문제 사항에 대하여 간단히 논의 한다. 현재 이 현황판은 개발실의 벽에 직접 구성하여 표시하고 갱신하고 있는데 이 경우 파손에 의한 손실이 있을 수 있으며 또 이 내용을 기록하여 다음 스프린트 또는 프로젝트 진행 시 참고할 만한 자료로 활용하기 위해서는 별도의 문서화 작업이 반드시 필요로 한다.

스프린트활동이 끝나면 데모를 하며 이때 해당 스프린트에 포함되었던 모든 제품 백로그 아이템에 대한 구현 확인이 이루어 져야 한다. 그렇게 하기 위해서는 반드시 스프린트 계획 시 각 제품 백로그 아이템 마다 기록해 두었던 데모 방법을 확인하고 그 절차데 따라 테스트 해야 한다. 스프린트 데모에는 개발자와 이해관계자, 고객이 모두 참여한다. 그리고 이 결과를 고객에게 릴리즈 한다.

개발자들은 한번의 스프린트가 끝나고 나면 스프린트 회고라는 활동을 통하여 스

프린트 활동을 개선해 나간다. 이것은 스크럼이 정해진 방법이 아닌 조직과 프로젝트 성격에 따라 그 실천법을 다양하게 적용할 수 있다는 특징이 있기 때문에 의미가 큰 활동이라 볼 수 있다.

프로젝트가 완료되지 않았다면 약간의 휴식 기간을 갖은 후 다음 스프린트 계획을 수립하고 이후 활동을 반복적으로 수행하게 된다.

위 그림에서 연구 범위 라고 표시된 파란색 박스 안의 실천 항목들은 본 연구의 목표에서 다루게 될 내용 실천 항목들이다.

3. 요구 분석

3.1. 도구 지원 범위 결정

스크럼 방법론을 애자일 패러다임의 전체적인 관점으로 보았을 때 순수 스크럼 실천법 뿐 아니라 XP, TDD 등의 기법들도 함께 포함하고 있다. 이번 연구에서는 우선 순수 스크럼 실천 법 중에서 제품 백로그 만들기, 스프린트 계획 회의, 스프린트 진행, 스프린트 데모에 관련된 활동을 도구화 하는 데 중점을 둘 것이며 그 이외의 활동은 향후의 연구 과제로 남겨둔다.

3.2. Usecase 분석

- 프로젝트 및 조직 정보 관리

프로젝트 및 조직 정보 관리는 프로젝트 자체에 대한 정보와 프로젝트에 참여하고 있는 개발자 그리고 그 외 고객을 제외한 이해관계자들에 대한 정보를 관리하는 기능들을 포함하고 있다.

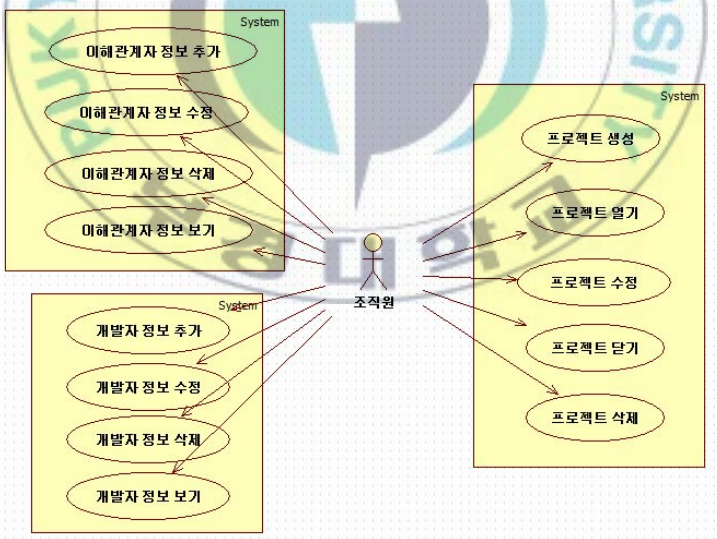

그림 2 프로젝트 및 조직 정보 관리 Usecase Diagram

프로젝트 및 조직 정보 관리에서는 프로젝트 이해관계자들에 대한 정보를 추가, 삭제, 수정, 열람 할 수 있어야 한다. 개발자 정보를 추가, 삭제, 수정, 열람 할 수 있어야 한다. 프로젝트 자체의 정보를 생성, 수정, 삭제 가능해야 한다. 또 현재 작업 중인 프로젝트를 닫을 수 있어야 하며, 이전에 만들어진 프로젝트 정보를 열어 볼 수 있는 기능을 제공해야 한다.

- 제품 책임자 관리

제품 책임자는 프로젝트를 발주한 고객 중 요구사항에 대한 결정권을 직접 가지거나 또는 대행 할 수 있는 사람들을 말한다. 애자일 패러다임 특성상 고객의 요구사항이 수시로 변하는 것에 대하여 유연하게 대처할 수 있어야 하는데 여기에 등록된 제품 책임자는 도구를 통하여 직접 개발사를 방문하지 않고도 요구사항을 수정할 수 있다.

그림 3 제품 책임자 관리 Usecase Diagram

제품 책임자 관리 기능에서는 제품 책임자의 기본정보를 생성하고 실제 제품 책임자 정보를 추가, 수정, 삭제, 열람 할 수 있어야 한다. 제품 책임자의 기본 정보는 제품 책임자들이 소속된 회사로서 프로젝트를 의뢰한 고객 업체에 대한 정보이다.

- 제품 백로그 관리

제품 백로그란 요구사항 리스트를 뜻한다고 볼 수 있다. 각 요구사항은 백로그 아이템이라고 불리며, 하나의 백로그 아이템은 그림 4에서 알 수 있듯이 이름, 중

요도, 추정치, 실제 속도, 상태, 데모 방법, 추가 설명 등의 속성을 가진다.

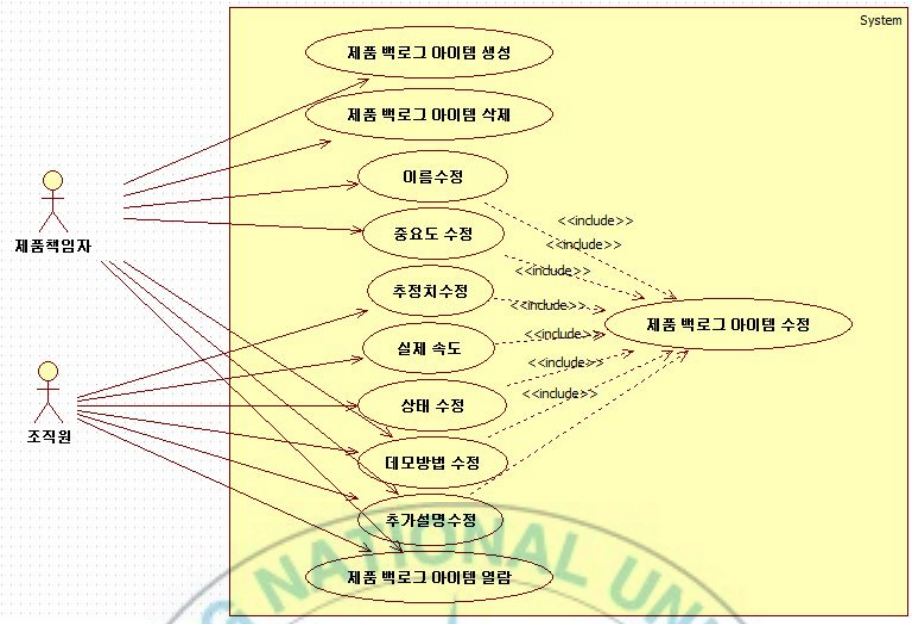

그림 4 제품 백로그 관리 Usecase Diagram

제품 백로그 아이템은 처음에 생성은 제품책임자 즉, 고객에 의해서 만들어 지고 수시로 갱신될 수 있다. 또한 각 항목의 중요도는 제품 책임자가 결정할 수 있는 값이다. 이렇게 생성된 제품 백로그 아이템을 개발하기 위해서는 개발자에 의해서 추정속도가 결정되고, 고객과 개발자가 함께 논의하여 이 항목의 데모 방법을 결정해야 한다. 이후 특정 스프린트에 포함되어 실제 개발이 진행되는 동안 개발자는 진행 상태를 수정할 수 있으며 이 값들은 스프린트 진행 작업 현황판을 통하여 나타난다.

- 스프린트 관리

스크럼에서는 하나의 프로젝트를 분할 정복 한다고 볼 수 있는데 이때 분할된 하나의 단위가 스프린트이다. 스프린트의 특성이라면 반드시 출시 가능한 결과물이 나올 수 있도록 분할되어야 한다는 것이며, 보통 하나의 스프린트는 2~4주 정도의 기간 내에 이루어 진다. 스프린트 관리 기능은 실제 프로젝트가 진행되는 과정을

관리하는 것으로 도구는 정확하고 쉽게 진행 상태를 나타낼 수 있도록 구현되어야 한다.

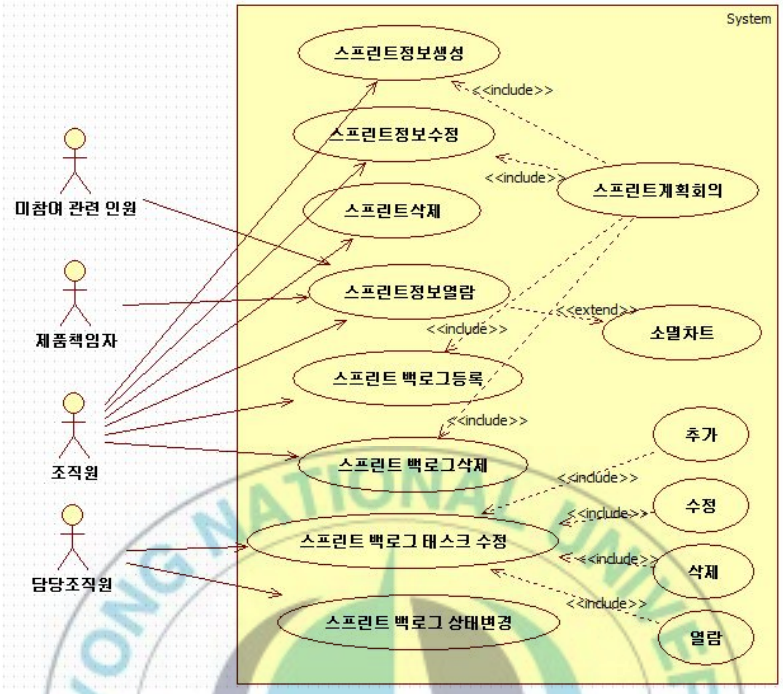

그림 5 스프린트 관리 Usecase Diagram

그림 5는 스프린트 진행중의 도구의 Usecase에 대하여 나타낸 Usecase Diagram 이다. 스프린트 계획 수립 활동에서 스프린트 정보를 생성하고 값을 설정한다. 값을 설정한 후 개발자 들에 의해서 실제 스프린트를 진행하게 된다. 스프린트 백로그란 제품 백로그들 중에서 해당 스프린트에 포함된 항목들을 말한다. 개발자는 이 스프린트 백로그 항목의 개발을 진행하게 되며, 이 과정에서 다시 하나의 항목은 여러 개의 태스크로 나누어 작업할 수 있다. 또 스프린트의 진행 정보는 소멸차트라는 시각적인 형태의 UI로 표현할 수 있어야 하며 이 내용은 프로젝트에 관련된 모든 사람이 같은 시점에서 동일한 내용을 조회 할 수 있어야 한다.

4. 도구 설계

4.1. 전체 시스템 개요

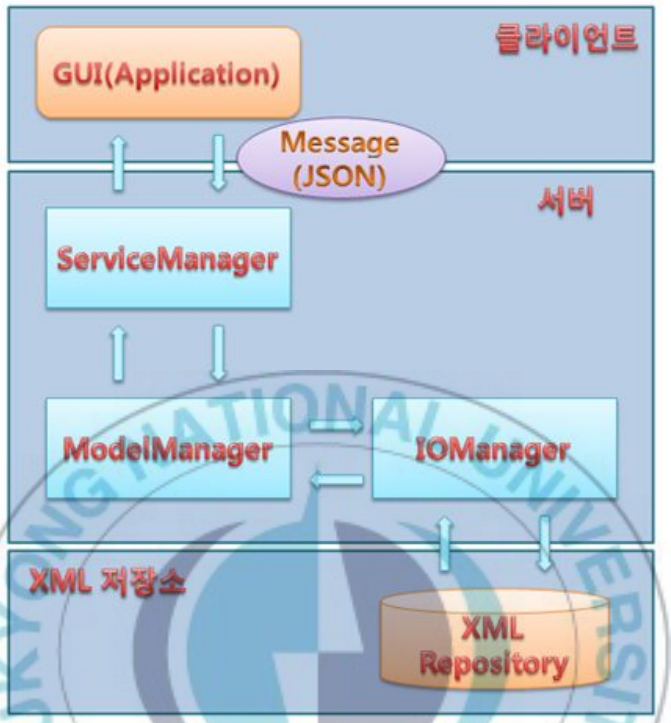

그림 6 전체 시스템 개요도

그림 6은 전체적인 도구의 시스템 구조 개요도 이다. UI로부터 전달되는 사용자 요청 액션을 ServiceManager가 받아 ModelManager와 연동되어 만약 저장소로의 접근이 필요로 한지를 판단하고 모델 정보를 바로 얻어 내거나 또는 저장소로부터 정보를 가져와서 모델 정보를 구성한 후 내용을 얻어 UI로 보내어 사용자의 요청에 응답하게 된다. Client의 구현은 일반 Application이든, Web Application이든 Smart Phone등 Mobile Application이든 관련 없이 독립적으로 구현될 수 있도록 한다. 이를 위해서 ServiceManager, ModelManager, IOManager에서는 모델 객체가 아닌 별도로 정의한 (부록 2참조) 프로토콜을 이용하여 서비스를 주고받도록 설계되

었다. 또한 Storage와의 IOManager와의 메시지도 이 프로토콜을 이용함으로써 Storage도 모델에 독립적으로 구현될 수 있으며 IOManager에 의해서 Storage로의 접속자체를 추상화 하여 위치적으로 도 떨어져 있거나 동일한 시스템에 있거나 관계없이 동작하도록 한다.

4.2. 데이터 설계

4.2.1. 개체 관계도

그림 7 데이터 개체 관계도

그림 7은 본 도구에서 필요한 데이터 분석으로 나타난 간략화 된 개체관계도(ERD)이다. 그림을 보면 열한개의 개체로 이루어져 있음을 알 수 있다. 태스크를 제외한 모든 개체의 속성은 그림에서는 생략하였다. 태스크는 하나의 속성 정보를 표현하였는데 이 개체는 특수한 개체로 하나의 백로그 아이템은 담당 개발자에 의해서 다시 여러 개의 태스크로 나뉠 수 있는데 이를 표현하기 위한 개체이다. 아래에 개체들의 관계를 간략히 설명한다.

- 하나의 프로젝트는 하나의 고객사를 갖는다.
- 하나의 프로젝트는 하나의 프로젝트 조직을 갖는다.
- 하나의 프로젝트는 다수의 스프린트를 갖는다.
- 고객사는 다수의 제품책임자를 갖는다.
- 프로젝트 조직은 다수의 이해관계자와 개발자를 갖는다.
- 제품책임자는 다수의 백로그 아이템에 책임이 있으며 하나의 백로그 아이템은 다수의 담당자를 가질 수 있다.
- 스프린트는 다수의 백로그 아이템과 이를 담당할 개발자를 갖는다.
- 개발자는 자신에게 할당된 백로그 아이템을 세부 작업인 태스크로 나누고 작업을 진행 할 수 있다.

본 연구에서는 도구를 통해 프로젝트를 시작하고 진행하는 동안 생성되는 모든 정보를 xml로 표현하도록 하였으며 위 개체를 5개의 xml로 표현하도록 최종 설계하였다.

개체	관련 Xml
프로젝트 및 조직 정보	project.xml
고객사 및 제품 책임자 정보	product_owner.xml
백로그 아이템 정보	product_backlog.xml
스프린트 정보	sprint.xml, member_sprint_backlog.xml

표 1 데이터 표현을 위한 xml구성

Xml은 그림 7의 데이터 분석내용에 맞추어 구성한 것인데 그림 7의 개체 관계도에 나타난 개체와 관계는 RDB의 관점으로 그 관계를 나타낸 것으로 xml파일의 개수가 개체의 개수로 도출되지 않고 일부를 통합 하여 구성하였다. 고객사는 제품책임자 정보로 통합되어 표현되도록 설계되었고, 프로젝트와 프로젝트 조직 이해관계자 개발자 정보는 프로젝트 정보라는 하나의 xml로 통합 관리하는 것으로 설계하였다. 제품 백로그 아이템과 스프린트는 그림 7의 모습과 유사하게 xml로 설계하였다. 제품 백로그 아이템은 요구사항이므로 프로젝트성공을 위한 가장 중요한 요소이므로 독립적으로 관리되어야 할 필요가 있으며, 스프린트의 경우 담당 개발자

는 도구를 이용하여 자신이 담당하게 된 스프린트 백로그 정보를 갱신해야 하는데 이때 다른 개발자와의 간섭을 피하기 위하여 각 개발자 단위로 스프린트 백로그 정보를 별도의 xml로 표현하도록 설계하였다.

Xml의 상세 설계 정보는 부록 1에 실제 개발에 사용될 DTD를 소개하고 각 Element와 Attribute에 대한 설명이 있으므로 참고 바란다.

4.3. 프로그램 클래스 설계

4.3.1. 데이터 표현을 위한 클래스 구조

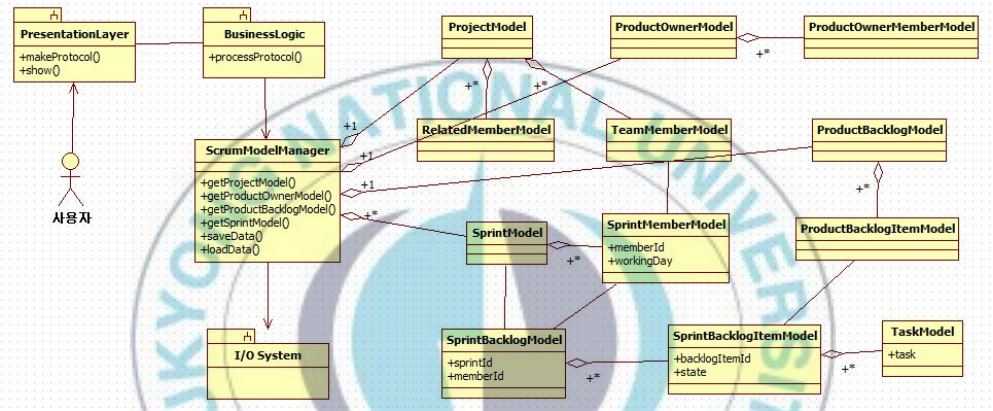

그림 8 데이터 표현을 위한 클래스 구조

그림 8은 데이터 표현 정보를 중심으로 보았을 때 도구의 모듈 구조도 이다. ProjectModel, ProductOwnerModel, SprintModel, ProductBacklogModel 이렇게 4개의 객체가 표 1의 개체에 대응되는 객체들이며 이들은 xml의 구조에 따라 복합적인 객체로 설계되었다. ScurmModelManager는 이 모델 객체들에 대한 접근 메소드를 제공하며 IOSystem과 연동하여 모델의 값을 저장소에 저장할 수 있는 프로토콜로 전환되는 정보를 담고 있게 된다. 이때 프로토콜을 생성하는 처리는 ProtocolManager 객체가 담당한다.

ScurmModelManager의 위쪽에 BusinessLogic System이 있는데 이것이 그림 6의 ServiceManager의 역할을 주로 담당하는 객체이다. 이렇게 ScrumModelManager에 의해서 사용자 요청으로부터 모델이 독립적이게 되고, IOSystem을 두어 저장소와의 독립성을 유지하도록 설계하였다.

4.3.2. 프로토콜 처리를 위한 클래스 구조

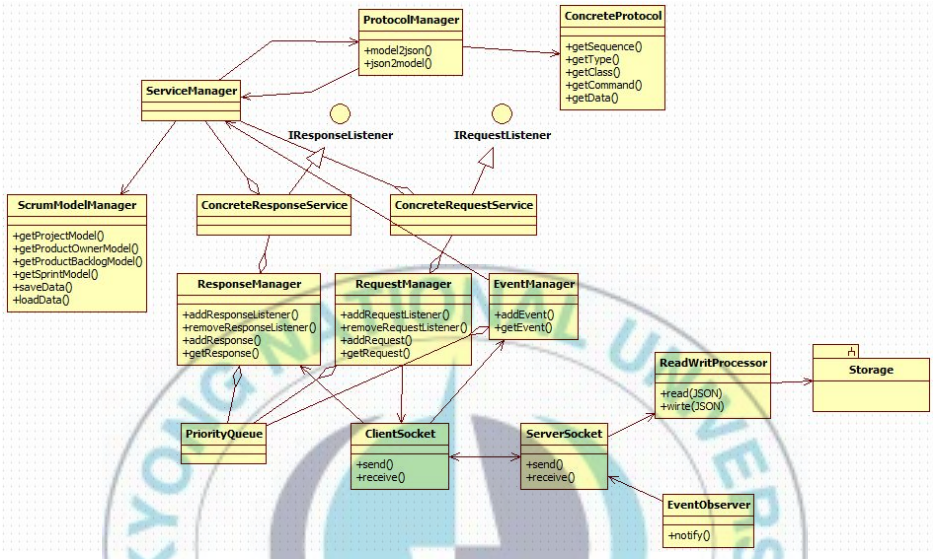

그림 9 프로토콜 처리를 위한 클래스 구조

그림 9는 프로토콜 처리 과정에 참여하는 클래스를 주된 관점으로 한 객체 구조도 이다. 그림 9의 왼쪽 상단에 ServiceManager객체가 있고 이 객체는 사용자의 요청액션 서비스를 받아 ProtocolManager를 이용하여 프로토콜을 구성한다. 이 때 ScrumModelManager로부터 요청을 만드는데 필요한 정보를 얻을 수 있다. 그리고 요청을 요청큐에 담는다. RequestManager는 요청큐를 일정 규칙으로 검사하여 요청이 있을 시 자신에게 등록된 IRequestListener 타입의 객체들 중에서 해당 요청을 처리할 수 있는 리스너에게 처리를 위임한다. 실제로 IRequestListener 인터페이스를 구현한 여러 개의 리스너 객체들이 RequestManager에 등록되어 있다. 이 리스너객체를 요청에 따른 처리를 수행하고 필요 시 이 요청을 서버로 보내어 처리하도

록 한다. 이렇게 처리된 결과는 응답 큐에 담기고 ResponseManager는 응답큐를 검사하여 응답이 있을 시 이를 처리 할 수 있는 IResponseListener 객체에게 처리를 위임한다. 그림 9에서는 요청 큐와 응답 큐를 별도로 표시하지 않고 왼쪽 아래에 PriorityQueue로 표시하여 나타내고 있다. 위와 같이 설계하여 어떤 요청이 처리 중일 때에도 도구의 UI동작이 블럭킹되지 않도록 하였다. 또한 IRequestListener와 IResponseListener를 두어 이 두 타입의 쌍을 추가함으로써 서비스의 추가를 쉽게 할 수 있도록 설계하였다. 이는 앞서 스크럼에 대해서 소개할 때 스크럼의 특성상 그 실천법이 조직이나 프로젝트의 특성에 따라 변할 수 있도록 해야 하는 것에 대비한 설계이다.

한가지 더 보아야 할 부분은 서버측에 있는 EventObserver인데 이는 사용자의 요청액션에 의하지 않고 데이터가 변경되거나 어떤 변화가 도구에 반영되어야 할 경우에 발생하는 것으로 예를 들면 스프린트가 완료되어 데모를 해야 할 경우 그 시간이 되기 얼마 전에 알려주어야 할 필요가 생긴다. 이러한 유형의 처리를 위한 것으로 서버(또는 같을 로컬 일수도 있음)로부터 Event가 전달되고 이는 이벤트 큐에 담기게 되고 EventManager에 의해서 요청이나 응답과 유사한 형태로 처리되도록 설계하였다.

4.3.3. 사용자 액션 처리 클래스 구조

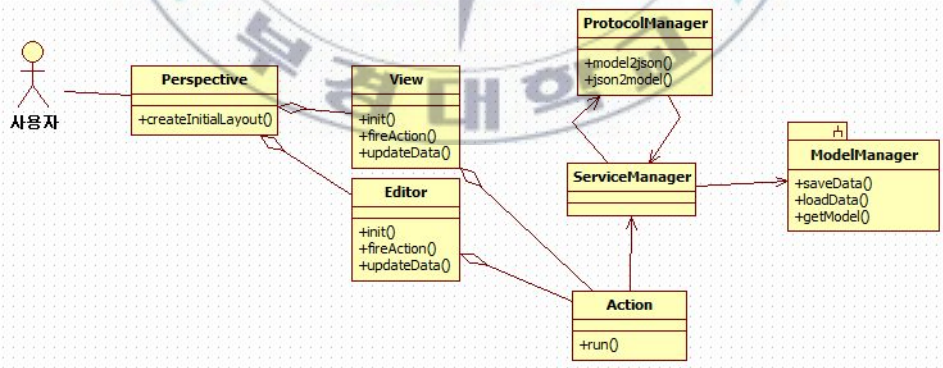

그림 10 사용자 액션 처리 클래스 구조

그림 10은 사용자의 UI액션 관점으로 본 도구의 객체 구조이다. 사용자는 자신이 작업하는 퍼스펙티브 환경을 유지하고 있게 된다. 퍼스펙티브에는 프로젝트 정보 퍼스펙티브, 제품 책임자 퍼스펙티브, 제품 백로그 퍼스펙티브, 스프린트 계획 수립 퍼스펙티브, 스프린트 진행 퍼스펙티브, 스프린트 데모 퍼스펙티브를 가지도록 설계하였으며, 그림 10에서는 이것들을 대표하여 Perspective라를 개체 하나로 표현하였다. 여기에서 각 컨텍스트에 맞는 UI를 구성하여 해당 컨텍스트에서 사용자가 관심 있어 할 만한 정보들을 나타낸다. 이때 UI는View와 Editor로 구성될 것이며 이를 통하여 정보를 보이게 된다. 즉, 사용자는 자신이 작업 시점에서 관심 있어하는 View와 Editor를 통하여 액션을 발생 시키게 되고 이는 ServiceManager에 전달되며 ProtocolManager에 의해서 프로토콜로 만들어 진 후 처리되도록 설계하였다.

4.4. 프로토콜 설계

프로토콜은 사용자의 요청을 전달하기 위해 사용되는데 요청이나 응답을 객체화하지 않고 프로토콜로 정의하여 이를 저장하는 데이터 베이스와 처리하는 모듈간의 결합도 줄여주어 본 프로젝트에서는 XML을 이용하여 저장하고 있으나 RDB나 기타 다른 데이터 표현기법을 이용할 수 있도록 하며, 또 정보를 사용자에게 보여주는 UI Layer에서의 모듈 또한 프로토콜만 처리할 수 있도록 구현하면 되므로 실제 UI 구현과는 독립적이게 된다. 그 결과 Web이나 Smart Phone등에서 클라이언트 프로그램만 구현하면 모든 정보를 표현할 수 있게 되는 것이다.

4.4.1. 프로토콜의 구조

그림 11 프로토콜 구조

모든 프로토콜의 구조는 위 그림과 같이 단순하게 설계되어 있다. 프로토콜의 정보 형식은 JSON형식[6]을 따르는 것으로 하고 있으며, 많은 정보를 필요로 하는 프

로토콜의 경우 상세 정보에 객체 타입의 배열로 표현하도록 설계하였다.

- 번호(Seq)

프로토콜 번호로서 요청과 응답은 동일한 번호를 갖도록 한다. 이렇게 하여 각 응답이 왔을 때 어떤 요처에 대한 응답인지 정확히 알 수 있다.

- 타입(Type)

이 프로토콜이 요청인지 응답인지를 구분하기 위한 값으로 요청 시 “Request”, 응답 시 “Response” 값을 갖는다.

- 클래스(Class)

이 프로토콜이 어떤 기능에 관련된 것 인지를 구분하기 위한 것으로 Project/Product-Owner/Product-Backlog/Sprint 이렇게 4개중 1개의 값을 갖는다.

- 명령(Command)

각 클래스 별로 어떤 행위를 해야 하는 지에 대한 상세 명령 정보를 가진다. 예를 들어 개발자의 정보를 가져오고자 할 경우 클래스 값은 Project가 될 것이며 명령은 getMember 값을 갖는다.

- 상세 정보(Data)

상세 정보란 명령을 수행하기 위해서 필요한 추가적인 정보를 말한다. 위에서 예로든 개발자의 정보를 가져오는 경우 해당 개발자의 id값을 전달해 주어야 한다.

4.5. UI 설계

도구의 UI 퍼스펙티브 단위로 구성할 것이라고 앞서 사용자 액션 관점의 모듈 설계애서언급한 바 있다. 이에 사용자 UI는 퍼스펙티브의 단위로 실제 그 구성을 스케치하여 설계한 내용을 나타낸다.

- 스크럼 각 활동 별 퍼스펙티브 UI

그림 12 퍼스펙티브 UI

위 그림은 (가), (나), (다), (라) 순으로 제품 책임자 정보 퍼스펙티브, 제품 백로그 퍼스펙티브 , 스프린트 데모 퍼스펙티브, 스프린트 계획 회의 퍼스펙티브의 UI 스케치를 나타내고 있다. UI의 구성을 이와 같이 함으로써 사용자는 자기가 현재 하고자하는 활동에 대한 정보에만 집중 할 수 있게 된다.

제품 책임자 정보 퍼스펙티브는 고객업체의 정보와 해당 업체에서 요구사항에 대한 책임이 있는 제품 책임자들의 정보를 나타내기 위한 UI이다.

제품 백로그 퍼스펙티브에서는 요구사항 리스트를 나타내는 UI이다. 이는 제품책임자가 수시로 활용하게 될 퍼스펙티브 이기도 하다.

스프린트 정보 퍼스펙티브는 스프린트 계획 수립 시 보게 될 UI로서 스프린트의 목표 및 시작일 데모일 등 기본적인 스프린트 정보를 설정하고 스프린트에 포함시킬 제품 백로그를 선택하고 담당 개발자를 지정할 수 있도록 하고 있다. 또 제품 백로그의 생성시 결정되지 않았던 중요도, 추정치 등의 추가적인 정보도 이때 결정된다.

스프린트 데모 퍼스펙티브에서는 스프린트가 완료된 후 제품 책임자 및 이해관계자들을 한자리에 모아 개발자들은 자신들의 작업 결과를 데모할 때 보이는 UI이다. 이때 데모의 방식은 스프린트 계획 시 미리 정해두었던 절차에 따라 진행되어야 하며 이 목록을 자동적으로 구성하여 나타내고 각 절차가 잘 완료되는지를 체크리스트로 나타낼 수 있도록 UI를 구성한다.

- 작업 현황 판 퍼스펙티브(스프린트 진행 퍼스펙티브)

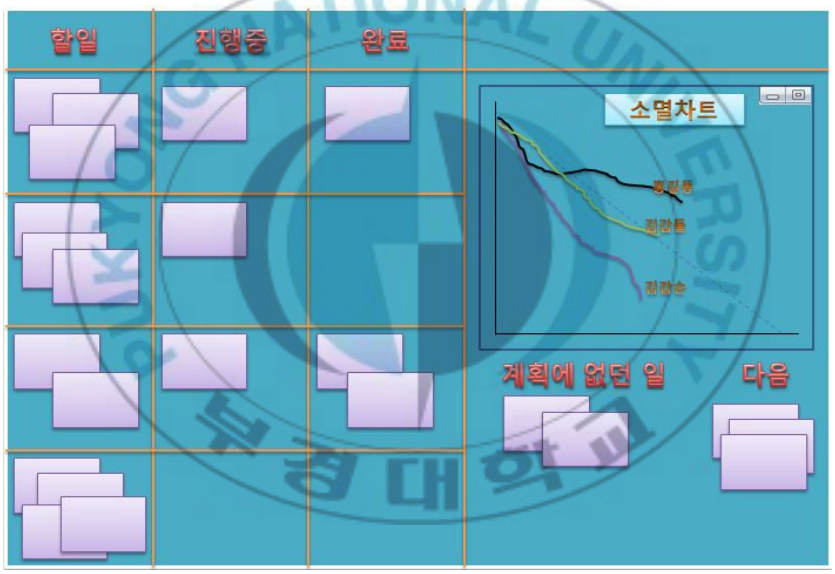

그림 13 작업 현황판 UI

스프린트가 진행되는 동안 모든 프로젝트 참여자들이 수시로 확인하게 될 퍼스펙티브이며, 개발자는 자신에게 할당된 백로그 아이템의 진행 상태를 최소 하루에 1번은 갱신해야 한다. 우측 상단의 소멸차트를 통하여 각 개발자 별 업무 진행 상황

을 알 수 있으며, 이를 통하여 프로젝트의 지연에 대한 위험을 미리 파악하고 팀장은 이를 해결할 수 있는 방안을 마련할 수 있다. 위 그림 15를 예로 든다면 김갑순이라는 개발자의 업무 진행상황이 느린 것을 알 수 있다. 이 경우 김갑순 개발자는 자신의 문제점에 대한 논의를 위해 일일 스크럼을 요청할 수 있으며, 또 팀장은 상대적으로 업무 진행이 빠른 홍길도 개발자와의 업무 조율을 통하여 스프린트가 지연되는 것을 방지 할 수 있다. 그리고 정확한 문제의 원인을 분석하여 다음 스프린트에는 더 정확한 계획을 수립할 수 있도록 한다.

5. 도구 평가

5.1. 변화된 스크럼 실천 활동 흐름

본 연구의 결과물인 도구를 이용했을 때의 스크럼 실천 흐름의 변화를 나타낸다.

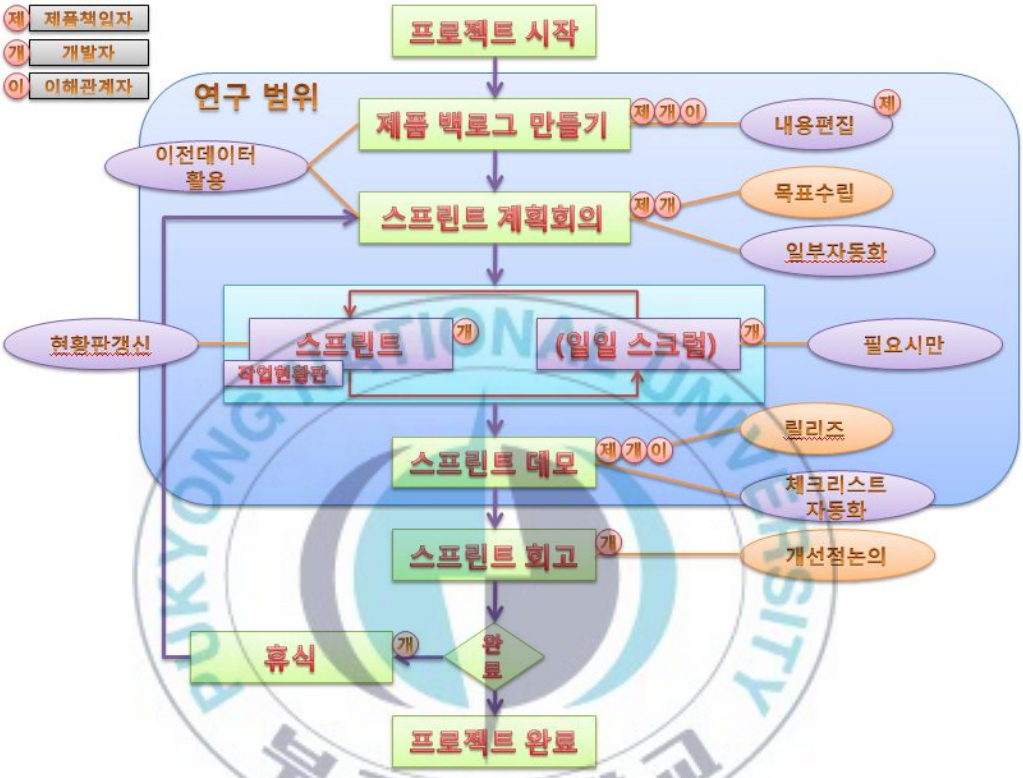

그림 14도구 도입후의 스크럼 활동 흐름

- 제품 백로그의 내용이 영구적으로 보관되어 이후에 유용한 정보로 활용될 수 있다.
- 고객이 제품 백로그의 내용을 변경할 경우 즉시 모든 프로젝트 관련 인원이 이를 알 수 있으며, 스프린트에서 진행중인 백로그 아이템에 대한 변경을 막을

수 있다.

- 스프린트에 포함될 백로그 아이템의 선택을 자동화 할 수 있으며 모호한 의사 결정을 위한 수단을 제공할 수 있다.
- 스프린트 진행 중 언제든지 작업 현황판의 갱신이 가능하며, 이 내용은 즉시 모든 프로젝트 관련 인원이 알 수 있다.
- 작업 현황판의 상황에 따라 문제점 유무가 파악되므로 일일 스크럼은 매일 하지 않아도 된다.
- 스프린트 데모 시 필요한 모든 제품 백로그 아이템을 빠짐없이 나타내고 각 아이템의 데모 방법을 자동으로 표시 할 수 있다.

5.2. 기대 효과

본 연구에서 제안한 스크럼 기반의 프로젝트 관리도구를 사용하였을 때 아래와 같은 이점이 있을 것으로 기대된다.

첫째, 별도의 교육이나 전문가의 도움을 최소화 하면서 스크럼 방법론을 쉽게 도입할 수 있도록 해준다. 둘째, 프로젝트의 진행 중에 개발팀원의 변화가 있더라도 그 사람의 역할 및 업무 진행 상태를 도구를 통해 알 수 있으므로 인수인계로 인한 시간 비용을 줄여준다. 셋째, 서로 떨어져 있는 모든 이해관계자들이 자신의 집무실에서 프로젝트의 진행상황을 알 수 있게 되고 이는 프로젝트가 실패할 확률을 줄이는 데 도움이 된다. 이러한 점을 잘 활용하면 불필요한 야근이나 프로젝트 막바지의 과도한 업무를 미리 예방할 수 있다. 넷째, 과거에 진행된 스프린트 정보가 축적되면 유사한 요구사항에 대한 위험을 미리 파악하거나 보다 더 정확한 추정치를 얻는데 활용할 수 있다. 이것은 개발계획 시 발생하는 비용 오차를 줄여 준다. 다섯째, 물리적으로 떨어져서 개발을 진행할 경우 매일매일 이루어 져야 하는 일일 스크럼을 실천하는데 어려움이 발생하는데, 도구를 통하여 진행 중인 백로그 아이템에 대한 상황전파가 빠르게 이루어져 직접 모여서 일일 스크럼을 진행한 것과 동일한 효과를 얻을 수 있다.

6. 결론

스크럼 기법은 그 자체만으로도 다양한 장점이 있지만 지원 도구를 이용하면 이러한 장점을 더욱 부각시킬 수 있게 될 것이며, 위에서 언급한 내용 이외에도 스크럼을 도입하면서 진행되는 여러 활동들을 더 편하게 실천할 수 있도록 해준다.

또한 도구는 스크럼이 조직 내에 자연스럽게 자리잡을 수 있는데 도움이 될 것이다. 끝으로 현재 도구는 스크럼의 핵심적인 기능을 중심으로 설계되었는데 향후 XP나 TDD등의 기법들도 활용할 수 있는 기능을 도입하는 것에 대해서 연구를 수행할 예정이다.

7. 참고 문헌

[1]헨릭 크니버그 지음, 심우곤, 엄위상, 한주영 옮김, ”스크럼과 XP”, 인사이트

[2]켄 슈와버, 마이크 비들 지음, 박일, 김기웅 옮김, ”Agile Software Development with Scrum”, 인사이트

[3]이세영, 용환승, “소규모 프로젝트를 위한 애자일 프레임워크 설계 및 평가”, 한국정보과학회논문지 제15권 11호 2009.11

[4]안중근, 인호, 이동현, 김능회, “해외 외주 프로젝트 이행을 위한 요구사항 시뮬레이션 도구 기반의 Agile Scrum Process 연구”, 한국정보과학회 2009 가을 학술발표 논문집 제 36권 제2호(B) 2009.11

[5]김태윤, 김남규, 손용락, “중소기업의 효율적 데이터 품질관리를 위한 스크럼 기반의 표준관리 방안”, 한국데이타베이스학회 제 17권 제1호 2010., pp83~105

[6]고석하, 홍정유 공저, “소프트웨어 프로젝트 관리”, 생능출판사

[7]론 제프리즈, 앤 앤더슨, 체트 핸드릭슨 공저, 박현철 등역, “Extreme Programming Installed” 인사이트

[8]켄 슈와버 저, 황상철 역, “엔터프라이즈 스크럼”, 에이콘출판사

[9] http://www.json.org/ JSON 웹사이트

[10] http://www.crisp.se/scrum/checklist 스크럼 실천 체크리스트 정보 웹사이트

[11] http://csmkorea.wordpress.com/ 스크럼 마스터 인증과정 웹사이트

8. 부록 1 DTD

데이터 분석 시 나타난 객 개체를 실제 구현에서 표현하기 위한 XML의 DTD에 대한 설명.

- 프로젝트(Project) 객체 상세 정보

프로젝트 정보 DTD

```
<!ELEMENT project (name, project-name, start-date, end-date,
sprint-period, description, member+, observer*)>
<!ELEMENT name (#PCDATA) required>
<!ELEMENT project-name (#PCDATA) required>
<!ELEMENT start-date (#PCDATA)>
<!ELEMENT end-date (#PCDATA)>
<!ELEMENT sprint-period (#PCDATA)>
<!ELEMENT description (#PCDATA)>
<!ELEMENT member (id, name, e-mail, phone, mobile, position,
description)>
<!ELEMENT id (#PCDATA)>
<!ELEMENT name (#PCDATA)>
<!ELEMENT e-mail (#PCDATA)>
<!ELEMENT phone (#PCDATA)>
<!ELEMENT mobile (#PCDATA)>
<!ELEMENT position (#PCDATA)>
<!ELEMENT description (#PCDATA)>
<!ELEMENT observer (id, name, e-mail, description)>
<!ELEMENT id (#PCDATA)>
<!ELEMENT name (#PCDATA)>
<!ELEMENT e-mail (#PCDATA)>
<!ELEMENT description (#PCDATA)>
```

프로젝트 정보 DTD 설명	
엘리먼트 또는 속성	설명
Project	루트 엘리먼트.
Name	내부적으로 관리되는 프로젝트 이름.
Project-name	실제 프로젝트 이름(고객이 원하는 이름으로 고객의 요청에

	따라 수시로 변할 수 있기 때문에 별도로 관리한다.)
start-date	프로젝트의 공식 시작일
end-date	정해진 프로젝트의 완료일
Sprint-period	스프린트 단위이다. 이 값은 스프린트 정보에서 시작일을 결정하면 자동적으로 적용되어 데모일이 설정되어야 한다. 만약 단위내에 프로젝트 종료일이 포함된다면 단위는 무시되고 프로젝트 종료일을 스프린트 종료일로 잡아야 한다.
description	프로젝트에 대한 부가적인 설명
member	프로젝트에 참여하는 팀원. 모든 팀원을 등록해야 하며 각 스크럼 팀은 여기에서 실제 스프린트에 참여하는 인원을 선택하여 등록하게 된다.
id	팀원을 구분하기 위한 ID 팀에서 정한 임의의 규칙대로 로 부여하면 된다. 이름이 같은 팀원이 있을 수 있기 때문에 고안된 요소이다. (원하는 경우 자동생성 가능하도록 한다.)
name	팀원의 이름
e-mail	팀원의 메일 주소. 회사에서 사용하는 공식 이메일 주소를 지정하는 것이 좋다.
phone	팀원의 전화번호로 사내 내선번호를 의미한다.
mobile	팀원의 핸드폰 번호.
position	팀원의 역할. 팀장, 스크럼 마스터, 개발자, 테스터, 다자이너 등 팀내에서 해당 맴버가 하는 역할을 지정한다. (기본적인 목록은 제공할 수 있으나 강요하지 않으며 직접 쓸수 있도록 한다.)
description	맴버에 대한 부가적인 설명
observer	프로젝트에 직접 참여하지는 않지만 프로젝트 진행을 보고 받기를 원하는 인원에 대한 정보이다.
id	이해관계자 정보를 식별하기 값
name	이해 관계자의 이름

e-mail	이해 관계자의 이메일 주소
description	이해관계자의 필요한 추가 정보 예를 들면 직급 따위를 기록한다.

- 제품 책임자(Product Owner) 개체 상세 정보

제품책임자 정보 DTD

```
<!ELEMENT product-owner-info (company, member+)>
<!ELEMENT company (name, phone, address)>
<!ELEMENT name (#PCDATA)>
<!ELEMENT phone (#PCDATA)>
<!ELEMENT address (#PCDATA)>
<!ELEMENT product-owner (id, name, e-mail, phone, mobile)>
<!ELEMENT id (#PCDATA)>
<!ELEMENT name (#PCDATA)>
<!ELEMENT e-mail (#PCDATA)>
<!ELEMENT phone (#PCDATA)>
<!ELEMENT mobile (#PCDATA)>
```

제품 책임자 정보 DTD 설명	
엘리먼트 또는 속성	설명
Product-owner-info	루트 엘리먼트.
Company	제품 책임자의 소속 회사 정보
name	제품책임자의 소속회사 이름
phone	제품책임자의 소속회사 전화번호
address	제품책임자의 소속회사 주소
Product-owner	제품 책임자 정보
id	제품 책임자를 구분하기 위한 ID 팀에서 정한 임의의 규칙대로 로 부여하면 된다. 이름이 같은 제품 책임자가 있을 수 있기 때문에 고안된 요소이다. (원하는 경우 자동생성 가능하도록 한다.)

name	제품 책임자의 이름
e-mail	제품 책임자의 메일 주소. 회사에서 사용하는 공식 이메일 주소를 지정하는 것이 좋다.
phone	제품 책임자의 전화번호로 사내 내선번호를 의미한다.
mobile	제품 책임자의 핸드폰 번호.

● 제품 백로그(Product Backlog) 개체 상세 정보

제품 백로그 정보 DTD

```
<!ELEMENT product-backlog (backlog-item)*>
<!ELEMENT backlog-item (id, name, significance, final-
estimation, demo-date, state, description)>
<!ELEMENT id (#PCDATA)>
<!ELEMENT name (#PCDATA)>
<!ELEMENT significance (#PCDATA)>
<!ELEMENT final-estimation (#PCDATA)>
<!ELEMENT real-cost (#PCDATA)>
<!ELEMENT demo-type (#PCDATA)>
<!ELEMENT state (#PCDATA)>
<!ELEMENT description (#PCDATA)>
```

제품 백로그 정보 DTD 설명

엘리먼트 또는 속성	설명
Product-backlog	루트 엘리먼트.
Backlog-item	백로그 아이템의 루트 엘리먼트
id	제품 백로그 항목을 구분하기 위한 ID. 팀에서 정한 임의의 규칙대로 갑을 할당하면 된다. (자동 생성 기능을 지원해야 한다.) 이 항목은 단순히 각 항목을 구분하기 위한 것일 뿐 우선순위나 순서와는 아무런 관련이 없다.
name	제품 백로그 항목에 대한 간략한 이룸
significance	중요도. 제품 책임자가 결정하는 값으로 값이 클수록 더 높은 중요도를 가지게 된다. 실제 제품 백로그 아이템은 이 값을

	기준으로 정렬되며 스프린트 계획 시 가장 높은 우선순위의 항목들을 우선적으로 선택하여야 한다.
Final-estimation	최종 추정치. 스프린트 계획 회의시 팀원들에 의해서 결정된다. 이 값은 해당 백로그 아이템을 처리하기 위해서 필요한 MD(Man-Day)값이다. (과거의 데이터를 참고/ 플래닝 포커 등 활요할 기능을 제공해야 한다.)
Real-cost	실제 비용. 해당 백로그 항목이 처리되는데 걸린 실제 비용을 MD단위로 기록한다. 이것을 처리가 완료된 후 해당 일을 담당했던 팀원에 의해서 기록되는 값이다.
Demo-type	데모 방법. 해당 백로그 아이템에 대한 데모방법을 기술한다. 예를 들면 입금에 대한 Demo-type의 내용은 (로그인, 입금 페이지열기, 100원 입금, 잔액조회 페이지로 이동, 잔액이 100원 증가했는지 확인)과 같을 수 있다.
State	상태. 해당 제품 백로그의 진행 상태를 나타낸다. 이 값은 대기, 준비, 진행, 완료 상태를 가지도록 한다. 신규: 생성 후 한번도 스프린트에 포함되지 않은 상태(처음 생성시 항상 이 값을 가짐) 대기: 이번 스프린트에 포함된 상태 진행: 스프린트에서 누군가 작업을 진행 중인 상태 완료: 백로그 항목의 개발이 완료된 상태. 스프린트가 끝나고 완료 또는 대기로 다시 변경된다.
description	백로그 항목에 대한 추가 정보를 기록한다.

- 스프린트(Sprint) 개체 상세 정보

스프린트 정보 DTD

```
<!ELEMENT sprint (id, goal, start-date, demo-date, power, real-power, concentration, state, daily-scrum, sprint-member+)>
<!ELEMENT id (#PCDATA) required>
<!ELEMENT goal (#PCDATA) required>
<!ELEMENT start-date (#PCDATA) required>
<!ELEMENT demo-date (#PCDATA) required>
<!ELEMENT power (#PCDATA) required>
<!ELEMENT real-power (#PCDATA)>
<!ELEMENT concentration (#PCDATA) required>
<!ELEMENT state (#PCDATA) required>
<!ELEMENT daily-scrum (time, place)>
```

```
<!ELEMENT time (#PCDATA) required>
<!ELEMENT place (#PCDATA) required>
<!ELEMENT sprint-member (member-id, working-day)>
<!ELEMENT member-id (#PCDATA) required>
<!ELEMENT working-day (#PCDATA) required>
```

스프린트 정보 DTD 설명	
엘리먼트 또는 속성	설명
Sprint	루트 엘리먼트.
Id	스프린트를 구분하기 위한 ID. 팀에서 정한 임의의 규칙에 의해서 부여할 수 있다. (자동 부여 기능을 지원해야 한다.)
Goal	스프린트의 목표, 반드시 존재해야 한다.
Start-date	스프린트의 시작일. 스프린트 계획회의가 끝나면 스프린트가 시작되는 것이다.
Demo-date	데모 날짜. 스프린트가 종료되는 날짜로 프로젝트 정보의 단위와 시작시점을 기준으로 자동 설정된다. (토/일은 휴일)
Power	스프린트 추정속도. 추정속도 = 가능한 맨-데이 * 집중도
Real-power	스프린트 실제속도 스프린트가 끝난 후 실제 속도로서 이것은 각 스토리의 최종 추정치에 의해 계산된다.
Concentration	스프린트 집중도. 최근 스프린트의 진행 상태로 구할 수 있다. 집중도 = 과거의 실제속도/가능한 맨-데이 (과거 데이터의 평균을 구할 수도 있다.)
state	스크럼이 완료된 상태인지 알기위함 (진행/완료 두 값중 한가지를 가짐) 스프린트의 상태는 준비가 없기 때문에 진행중과 완료 뿐이다.)
Daily-scrum	일일 스크럼 정보

Time	일일 스크럼 시간
place	일일 스크럼 장소
Sprint-member	스프린트에 참여하는 스크럼 팀 정보
Member-id	팀원의 ID
Working-day	해당 멤버의 근무일 (모든 멤버의 근무일이 스프린트의 "가능한 멘-데이"가 된다.)

● 스프린트 백로그(Sprint Backlog) 개체 상세 정보

스프린트 정보 DTD

```
<!ELEMENT sprint-backlog (sprint-id, member-id, backlog-item*)>
<!ELEMENT sprint-id (#PCDATA)>
<!ELEMENT member-id (#PCDATA)>
<!ELEMENT backlog-item (id, state, tasks)>
<!ELEMENT id (#PCDATA)>
<!ELEMENT state (#PCDATA)>
<!ELEMENT tasks (task*)>
<!ELEMENT task (#PCDATA)>
```

스프린트 정보 DTD 설명	
엘리먼트 또는 속성	설명
Sprint-backlog	루트 엘리먼트.
Sprint-id	관련 스프린트 ID
Member-id	담당 팀원 ID
Backlog-item	이 팀원이 담당한 백로그 아이템 정보
id	제품 백로그에서 이번 스프린트에서 진행하기로 결정된 백로그 항목의 ID값을 가진다. (이 항목은 스프린트 가용량에 따라 자동으로 할당 될 수가 있다.)
state	상태. 해당 제품 백로그의 진행 상태를 나타낸다. 이 값은 대기, 준비, 진행, 완료 상태를 가지도록 한다.

	대기: 스프린트 백로그에 그냥 존재하는 상태 진행: 현재 진행중인 상태 완료: 백로그 항목의 개발이 완료된 상태. 스프린트가 제품 백로그의 state를 갱신한다.
tasks	태스크의 루트 엘리먼트
task	개발자가 이 아이템에 대하여 작업을 세분화하기 위해서 기록한 내용.

9. 부록 2 Protocol

- 프로젝트 정보

프로젝트 정보 요청	
Seq	순서대로 붙여지는 번호
Type	Request
Class	Project
Command	getProjectInfo
Data	{name:프로젝트식별자}

프로젝트 정보 응답	
Seq	요청과 같은 번호
Type	Response
Class	Project
Command	Successful/Failed
Data	{name:프로젝트식별자, project-name:프로젝트이름, start-date:시작날짜, end-date:종료날짜, sprint-period:스프린트기간, description:부가설명, member-count:개발자인원수, member:[{id:아이디, name: 이름, e-mail:이메일주소, phone:내선번호, mobile:이동전화번호, position:역할, description:부가설명}, {…}, {…}, …], observer-count:이해관계자 인원수, observer[{id:이해관계자아이디, name:이해관계자이름, e-mail:이해관계자이메일주소, description:이해관계자부가설명}, {…}, {…}, …]}

- 제품 책임자 정보

제품 책임자 정보 요청	
Seq	순서대로 붙여지는 번호
Type	Request

Class	Product-Owner
Command	getProductOwnerInfo
Data	{name:프로젝트식별자}

제품 책임자 정보 응답	
Seq	요청과 같은 번호
Type	Response
Class	Product-Owner
Command	Successful/Failed
Data	{company-name:고객회사명, company-phone:고객회사 전환번호, compaly-address:고객회사 주소, product-owner-count:제품책임자 인원수, product-owner:[{id:제품책임자ID, name:제품책임자이름, e-mail:제품책임자이메일주소, phone:제품책임자사내연락처, mobile:제품책임자이동전화}, {…}, {…}, …]}

- 제품 백로그 정보

제품 백로그 정보 요청	
Seq	순서대로 붙여지는 번호
Type	Request
Class	Product-Backlog
Command	getProductBacklog
Data	{name:프로젝트식별자}

제품 백로그 정보 응답	
Seq	요청과 같은 번호
Type	Response

Class	Product-Backlog
Command	Successful/Failed
Data	{product-backlog-count:제품 백로그 아이템 개수, product-backlog-item:[{id:백로그아이템ID, name:이름, significance:중요도, final-estimation:최종추정지, real-cost:실제비용, demo-type:[{"데모방법","확인"},{…},{…}, …], state:현재상태, description:추가설명}, {…}, {…}, …]}

● 스프린트 현황

스프린트 현황 정보 요청	
Seq	순서대로 붙여지는 번호
Type	Request
Class	Sprint
Command	getSprintStatusInfo
Data	{project-name:프로젝트식별자, sprint-id:스프린트ID}

스프린트 현황 정보 응답	
Seq	요청과 같은 번호
Type	Response
Class	Sprint
Command	Successful/Failed
Data	{project-name:프로젝트식별자, sprint-id:스프린트ID, goal:스프린트목표, start-date:시작일, demo-date:데모일, power:추정속도, real-power:실제속도, concentration:집중도, state:스프린트상태, daily-scrum:["시간", "장소"], sprint-member-count:스크럼팀 인원수, sprint-member[{member-id:개발자 ID, working-day:근무일수}, {…}, {…}, …], sprint-backlog-count:스프린트백로그아이템개수, sprint-backlog:[{member-id:담당개발자Id, state:상태, task-count:태스크개수, task:["태스크1", "태스크2", …]}, {…}, {…}, …]}

● 스프린트 데모

스프린트 데모 요청	
Seq	순서대로 붙여지는 번호
Type	Request
Class	Sprint
Command	getSprintDemoInfo
Data	{project-name:프로젝트식별자, sprint-id:생성될스프린트ID}

스프린트 데모 응답	
Seq	요청과 같은 번호
Type	Response
Class	Sprint
Command	Successful/Failed
Data	{project-name:프로젝트식별자, sprint-id:스프린트ID, goal:스프린트목표, start-date:시작일, demo-date:데모일, sprint-backlog-count:제품 백로그 아이템 개수, sprint-backlog-item:[{id:백로그아이템ID, member-id:담당개발자ID, name:이름, significance:중요도, final-estimation:최종추정지, real-cost:실제비용, demo-type:[{"데모방법","확인"},{…},{…}, …]}

● 스프린트 데모 완료(체크 상태를 저장함)

스프린트 데모 완료 요청	
Seq	순서대로 붙여지는 번호
Type	Request
Class	Sprint
Command	finishSprintDemo

Data	{project-name:프로젝트식별자, sprint-id:스프린트ID, goal:스프린트목표, start-date:시작일, demo-date:데모일, sprint-backlog-count:제품 백로그 아이템 개수, sprint-backlog-item:[{id:백로그아이템ID, member-id:담당개발자ID, name:이름, significance:중요도, final-estimation:최종추정지, real-cost:실제비용, demo-type:[{"데모방법","확인"},{…},{…}, …]}

스프린트 데모 완료 응답	
Seq	요청과 같은 번호
Type	Response
Class	Sprint
Command	Successful/Failed
Data	

- 프로젝트 생성

프로젝트 생성 요청	
Seq	순서대로 붙여지는 번호
Type	Request
Class	Project
Command	newProjectInfo
Data	{name:프로젝트식별자, project-name:프로젝트이름, start-date:시작날짜, end-date:종료날짜, sprint-period:스프린트기간, description:부가설명, member-count:개발자인원수, member:[{id:아이디, name: 이름, e-mail:이메일주소, phone:내선번호, mobile:이동전화번호, position:역할, description:부가설명}, {…}, {…}, …], observer-count:이해관계자 인원수, observer[{id:이해관계자아이디, name:이해관계자이름, e-mail:이해관계자이메일주소, description:이해관계자부가설명}, {…}, {…}, …]}

프로젝트 생성 응답	
Seq	요청과 같은 번호
Type	Response
Class	Project
Command	Successful/Failed
Data	

● 프로젝트 열기

프로젝트 열기 요청	
Seq	순서대로 붙여지는 번호
Type	Request
Class	Project
Command	openProjectInfo
Data	{name:프로젝트식별자}

프로젝트 열기 응답	
Seq	요청과 같은 번호
Type	Response
Class	Project
Command	Successful/Failed
Data	{name:프로젝트식별자, project-name:프로젝트이름, start-date:시작날짜, end-date:종료날짜, sprint-period:스프린트기간, description:부가설명, member-count:개발자인원수, member:[{id:아이디, name: 이름, e-mail:이메일주소, phone:내선번호, mobile:이동전화번호, position:역할, description:부가설명}, {…}, {…}, …], observer-count:이해관계자 인원수, observer[{id:이해관계자아이디, name:이해관계자이름, e-mail:이해관계자이메일주소,

	description:이해관계자부가설명}, {…}, {…}, …]}

● 프로젝트 저장

프로젝트 저장 요청	
Seq	순서대로 붙여지는 번호
Type	Request
Class	Project
Command	saveProjectInfo
Data	{name:프로젝트식별자, project-name:프로젝트이름, start-date:시작날짜, end-date:종료날짜, sprint-period:스프린트기간, description:부가설명, member-count:개발자인원수, member:[{id:아이디, name: 이름, e-mail:이메일주소, phone:내선번호, mobile:이동전화번호, position:역할, description:부가설명}, {…}, {…}, …], observer-count:이해관계자 인원수, observer[{id:이해관계자아이디, name:이해관계자이름, e-mail:이해관계자이메일주소, description:이해관계자부가설명}, {…}, {…}, …]}

프로젝트 저장 응답	
Seq	요청과 같은 번호
Type	Response
Class	Project
Command	Successful/Failed
Data	

● 프로젝트 닫기

프로젝트 닫기 요청	
Seq	순서대로 붙여지는 번호

Type	Request
Class	Project
Command	closeProjectInfo
Data	{name:프로젝트식별자}

프로젝트 닫기 응답	
Seq	요청과 같은 번호
Type	Response
Class	Project
Command	Successful/Failed
Data	

- 프로젝트 삭제

프로젝트 삭제 요청	
Seq	순서대로 붙여지는 번호
Type	Request
Class	Project
Command	removeProjectInfo
Data	{name:프로젝트식별자}

프로젝트 삭제 응답	
Seq	요청과 같은 번호
Type	Response
Class	Project

Command	Successful/Failed
Data	

● 개발자 추가

개발자 추가 요청	
Seq	순서대로 붙여지는 번호
Type	Request
Class	Project
Command	addProjectMemberInfo
Data	{name:프로젝트식별자, member-info:{id:개발자ID, name:개발자이름, e-mail:개발자 메일주소, phone:개발자내선번호, mobile:개발자이동전화번호, position:역할, description:추가설명}}

개발자 추가 응답	
Seq	요청과 같은 번호
Type	Response
Class	Project
Command	Successful/Failed
Data	

● 개발자 삭제

개발자 삭제 요청	
Seq	순서대로 붙여지는 번호
Type	Request
Class	Project

Command	removeProjectMemberInfo
Data	{name:프로젝트식별자, member- id:개발자ID}

개발자 삭제 응답	
Seq	요청과 같은 번호
Type	Response
Class	Project
Command	Successful/Failed
Data	

- 이해관계자 추가

이해관계자 추가 요청	
Seq	순서대로 붙여지는 번호
Type	Request
Class	Project
Command	addProjectObserverInfo
Data	{name:프로젝트식별자, member- id:개발자ID}

이해관계자 추가 응답	
Seq	요청과 같은 번호
Type	Response
Class	Project
Command	Successful/Failed
Data	

● 이해관계자 삭제

이해관계자 삭제 요청	
Seq	순서대로 붙여지는 번호
Type	Request
Class	Project
Command	removeProjectObserverInfo
Data	{name:프로젝트식별자, observer- id:이해관계자ID}

이해관계자 삭제 응답	
Seq	요청과 같은 번호
Type	Response
Class	Project
Command	Successful/Failed
Data	

● 제품 책임자 추가

제품 책임자 추가 요청	
Seq	순서대로 붙여지는 번호
Type	Request
Class	Product-Owner
Command	addProductOwnerInfo
Data	{name:프로젝트식별자, product-owner-info:{id:제품책임자ID, name:이름, e-mail:이메일주소, phone:회사직통번호, mobile:이동전화번호}}

제품 책임자 추가 응답	
Seq	요청과 같은 번호
Type	Response
Class	Product-Owner
Command	Successful/Failed
Data	

● 제품 책임자 삭제

제품 책임자 삭제 요청	
Seq	순서대로 붙여지는 번호
Type	Request
Class	Product-Owner
Command	removeProductOwnerInfo
Data	{name:프로젝트식별자, product-owner-id:제품책임자ID}

제품 책임자 삭제 응답	
Seq	요청과 같은 번호
Type	Response
Class	Product-Owner
Command	Successful/Failed
Data	

● 제품 책임자 정보 저장

제품 책임자 정보 저장 요청	
Seq	순서대로 붙여지는 번호

Type	Request
Class	Product-Owner
Command	saveProductOwnerInfo
Data	{project-name:프로젝트 식별자, company-name:고객회사명, company-phone:고객회사 전환번호, compaly-address:고객회사 주소, product-owner-count:제품책임자 인원수, product-owner:[{id:제품책임자ID, name:제품책임자이름, e-mail:제품책임자이메일주소, phone:제품책임자사내연락처, mobile:제품책임자이동전화}, {…}, {…}, …]}

제품 책임자 정보 저장 응답	
Seq	요청과 같은 번호
Type	Response
Class	Product-Owner
Command	Successful/Failed
Data	

● 제품 백로그 아이템 추가

제품 백로그 아이템 추가 요청	
Seq	순서대로 붙여지는 번호
Type	Request
Class	Product-Backlog
Command	addProductBacklogInfo
Data	{name:프로젝트식별자, product-backlog-info:{id:제품백로그ID, name:이름, significance:중요도, final-estimation:최종추정치, demo-type:{[“데모방법1”, “확인”], [“”,””], [“”,””], …}, state:상태, description:추가설명}}

제품 백로그 아이템 추가 응답	
Seq	요청과 같은 번호
Type	Response
Class	Product-Backlog
Command	Successful/Failed
Data	

● 제품 백로그 아이템 삭제

제품 백로그 아이템 삭제 요청	
Seq	순서대로 붙여지는 번호
Type	Request
Class	Product-Backlog
Command	removeProductBacklogInfo
Data	{name:프로젝트식별자, product-backlog-id:제품백로그ID}

제품 백로그 아이템 삭제 응답	
Seq	요청과 같은 번호
Type	Response
Class	Product-Backlog
Command	Successful/Failed
Data	

● 제품 백로그 정보 저장

제품 백로그 아이템 저장 요청	
Seq	순서대로 붙여지는 번호

Type	Request
Class	Product-Backlog
Command	saveProductBacklogInfo
Data	{project-name:프로젝트식별자, product-backlog-count:제품 백로그 아이템 개수, product-backlog-item:[{id:백로그아이템ID, name:이름, significance:중요도, final-estimation:최종추정지, real-cost:실제비용, demo-type:[{"데모방법","확인"},{…},{…}, …], state:현재상태, description:추가설명}, {…}, {…}, …]}

제품 백로그 아이템 저장 응답	
Seq	요청과 같은 번호
Type	Response
Class	Product-Backlog
Command	Successful/Failed
Data	

● 스프린트 계획 회의

이 액션에서는 기존 ID를 이용하여 정보를 가져오거나 또는 새롭게 생성한다. (해당 프로토콜 참조)

● 스프린트 정보 생성

스프린트 정보 생성 요청	
Seq	순서대로 붙여지는 번호
Type	Request
Class	Sprint
Command	newSprintInfo
Data	{project-name:프로젝트식별자, sprint-id:생성될스프린트ID}

스프린트 정보 생성 응답	
Seq	요청과 같은 번호
Type	Response
Class	Sprint
Command	Successful/Failed
Data	

● 스프린트 정보 삭제

스프린트 정보 삭제 요청	
Seq	순서대로 붙여지는 번호
Type	Request
Class	Sprint
Command	removeSprintInfo
Data	{project-name:프로젝트식별자, sprint-id:제거될스프린트ID}

스프린트 정보 삭제 응답	
Seq	요청과 같은 번호
Type	Response
Class	Sprint
Command	Successful/Failed
Data	

● 스프린트 정보 저장

스프린트 정보 저장 요청	
Seq	순서대로 붙여지는 번호

Type	Request
Class	Sprint
Command	saveSprintInfo
Data	{project-name:프로젝트식별자, sprint-id:스프린트ID, goal:스프린트목표, start-date:시작일, demo-date:데모일, power:추정속도, real-power:실제속도, concentration:집중도, state:스프린트상태, daily-scrum:["시간", "장소"], sprint-member-count:스크럼팀 인원수, sprint-member[{member-id:개발자 ID, working-day:근무일수}, {…}, {…}, …]}

스프린트 정보 저장 응답	
Seq	요청과 같은 번호
Type	Response
Class	Sprint
Command	Successful/Failed
Data	

● 스프린트 백로그 아이템 추가

스프린트 백로그 아이템 추가 요청	
Seq	순서대로 붙여지는 번호
Type	Request
Class	Sprint
Command	addSprintBacklogItem
Data	{project-name:프로젝트식별자, sprint-id:스프린트ID, member-id:개발자ID, product-backlog-id:제품백로그ID}

스프린트 백로그 아이템 추가 응답

Seq	요청과 같은 번호
Type	Response
Class	Sprint
Command	Successful/Failed
Data	

● 스프린트 백로그 아이템 삭제

스프린트 백로그 아이템 삭제 요청	
Seq	순서대로 붙여지는 번호
Type	Request
Class	Sprint
Command	removeSprintBacklogItem
Data	{project-name:프로젝트식별자, sprint-id:스프린트ID, product-backlog-id:제품백로그ID}

스프린트 백로그 아이템 삭제 응답	
Seq	요청과 같은 번호
Type	Response
Class	Sprint
Command	Successful/Failed
Data	

● 스프린트 백로그 아이템 정보 저장

스프린트 백로그 아이템 저장 요청	
Seq	순서대로 붙여지는 번호

Type	Request
Class	Sprint
Command	saveSprintBacklogItem
Data	{project-name:프로젝트식별자, sprint-id:스프린트ID, member-id:개발자ID, product-backlog-id:제품백로그ID, sprint-backlog-info:{state:상태, task-count:태스크개수, task:["태스크1", "태스크2", …]}}

스프린트 백로그 아이템 저장 응답	
Seq	요청과 같은 번호
Type	Response
Class	Sprint
Command	Successful/Failed
Data	

- 스크럼 팀원 추가

스크럼 팀원 추가 요청	
Seq	순서대로 붙여지는 번호
Type	Request
Class	Sprint
Command	addSprintMember
Data	{project-name:프로젝트식별자, sprint-id:스프린트ID, member-id:개발자ID}

스크럼 팀원 추가 응답	
Seq	요청과 같은 번호

Type	Response
Class	Sprint
Command	Successful/Failed
Data	

● 스크럼 팀원 삭제

스크럼 팀원 삭제 요청	
Seq	순서대로 붙여지는 번호
Type	Request
Class	Sprint
Command	removeSprintMember
Data	{project-name:프로젝트식별자, sprint-id:스프린트ID, member-id:개발자ID}

스크럼 팀원 삭제 응답	
Seq	요청과 같은 번호
Type	Response
Class	Sprint
Command	Successful/Failed
Data	

감사의 글...

2005년도에 입학하여 회사 생활과 맞물려 중간에 학업을 중단하여 6년이 지난 지금에야 졸업을 앞두게 되었습니다. 회사 일을 핑계로 학업에 더 충실하지 못하였던 점이 무척이나 아쉬움으로 남지만 그래도 늦게나마 이렇게 졸업할 수 있게 된 것을 감사하게 생각합니다.

6년이라는 긴 시간 동안 저를 지도해 주신 송하주 교수님께 깊은 감사의 말씀을 드립니다. 그리고 논문을 내는데 많은 도움을 준 랩실 동생들 태용, 승혁, 영준, 성진, 주형, 재형 모두에게 고맙다는 말을 전합니다.

또 매 학기 열심히 지도해주신 교수님들과 바쁘다는 핑계로 학사일정에 소홀했을 때 꼼꼼히 챙겨주신 조교 선생님들 에게도 감사의 말씀을 드립니다.

학업을 중단했을 때 다시 시작할 수 있도록 도와준 동생 웅이 학기가 진행되는 동안 여러가지 챙겨주고 함께 공부했던 동생들 기정이, 기봉이 여러 가지 진행 일정을 매번 알려준 진희씨에게도 고맙다는 말을 전합니다.

끝으로 제가 일과 학업을 모두 할 수 있도록 건강을 챙겨주신 아버지와 마음속에서 저를 지켜봐 주신 어머니께 감사의 말씀을 드립니다.

졸업이란 기다려 지면서도 막상 다가오면 많은 아쉬움을 남기는 것 같습니다. 그 아쉬운 마음은 앞으로 살아가는데 더 열심히 노력하는 동기가 되리라 생각하며 제가 몸담았던 IDBLAB의 사람들을 잊지 않겠습니다.

2011년 1월

김태형