ALM / Task management process

자바스터디 조대협
(http://bcho.tistory.com)

Table ofcontents

Overview

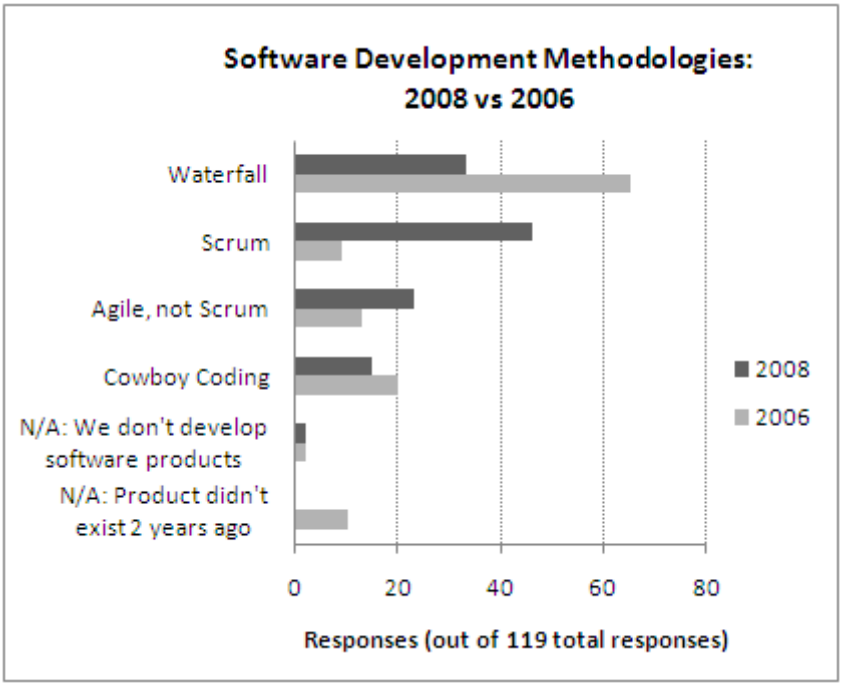
프로젝트에서 중요한 포인트중의 하나는 팀의 운영과 관리이다. 프로젝트에 Unified Process 나 Waterfall model과 같은 기존의 방법론을 사용하더라도, 그 방법론에는 자세한 task 관리 프로세스에 대해서는 거의 정의가 되어 있지 않다.

반대로 요즘 유행하는 Agile 방법론의 경우, task 관리에 대한 전략과 수행 방법을 기술하고 있지만, 실제 프로젝트를 관리하는 관점에서는 전체 스케쥴에 대한 예측과 관리가 어렵기 때문에(불확실성의 문제), SI 프로젝트등에서는 쉽게 적용할 수 없는 문제를 가지고 있다.
또한 실제 프로젝트에서는 이미 고객이나 주관사의 방법론을 표준으로 사용하고 있기 때문에 Agile 방법론을 적용하는 것이 쉽지 않다.

이 문서의 목적은 위 두 가지 접근 방법의 문제점을 상호 보완하여, 전체 프로세스는 기존의 전통 방법론을 따르고, 전체 프로세스의 각 단계를 Agile 방법론으로 보완하여, 실질적인 task 관리 방법론을 구축하고자 하는데 목적이 있다.

Agile 방법론은 요즘 들어 가장 인기 있고 가벼운 방법론 중에 하나이다. 이 방법론의 특징은 구성원간의 협업 (Collaboration)을 중요시 하는 사상을 가지고 있다. 본 문서에는 이 사상을 기반으로 하여, 일일 Task 관리를 하는 기법을 중심으로 설명하고자 한다.

Agile 방법론에는 XP,AUP,Scrum,Lean 등 여러가지 구체적인 방법론 들이 있는데, 아래 도표를 보면 알 수 있듯이, 근래에 들어서는 Scrum이 가장 인기있고 많이 사용되는 방법론이다.

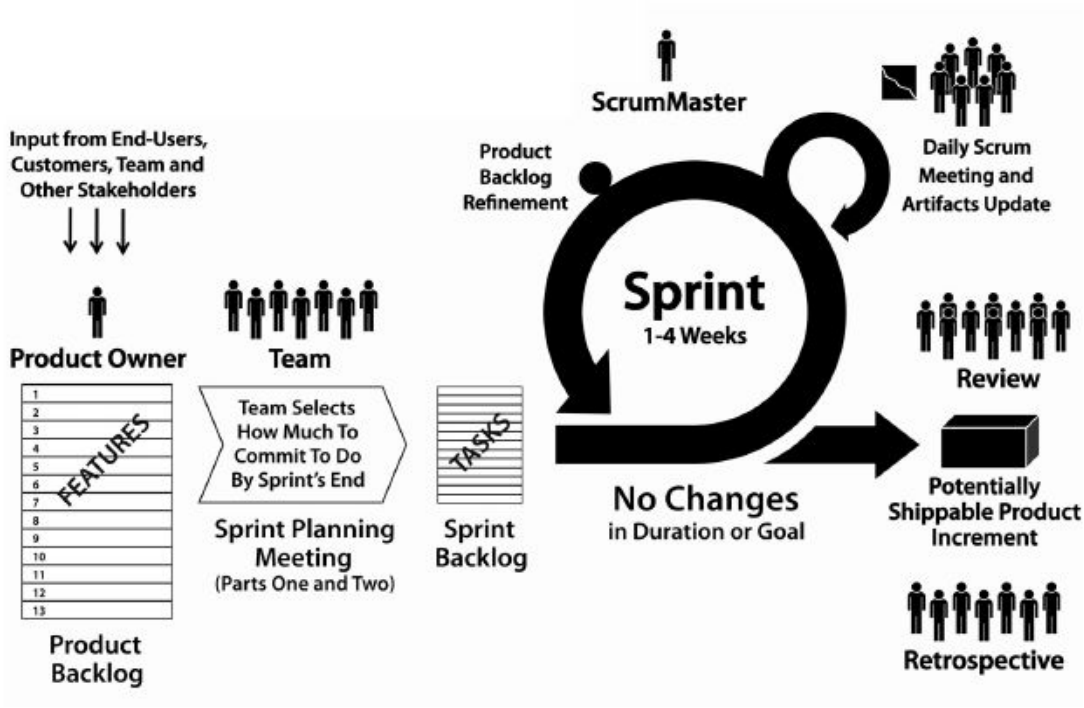


출처 http://crankypm.com/2008/10/poll-results-software-development-methodologies-agile-vs-waterfall/

Scrum

Summary

그럼 먼저 Scrum 방법론에 대해서 살펴보도록 하자.



Scrum은 기본적으로 Iterative and incremental 개발 방법에 기초를 한다.
Scrum은 몇 개의 Iteration으로 구성되는데, 이 Iteration을 **Sprint**라고 부르며 각 Sprint는 1~4주 정도의 기간을 갖는다. 이 Sprint는 일종의 Timebox의 개념을 가지며 한번 정해진 Sprint의 기간은 변경될 수 없다는 것을 전제로 한다.

다음으로 Scrum에는 고객의 요건으로부터 작성된 **Product Backlog**라는 것을 가지고 있다. 쉽게 이야기 하면 "대략적인 할일 목록"이다. 구현 단계에 Scrum을 적용할 경우에는 SRS (Software Requirement Specification)이나 TRS (Technical Requirement Specification)들의 목록이 Product Backlog의 항목으로 적절하다.
이 Product Backlog의 항목을 팀 미팅을 통해서 이번 Sprint에 구현할 항목을 선택한다. 항목의 선택은 Product Backlog 항목의 우선순위(Priority)를 통해서 선택이 되고, 이 선택된 항목들을 구체적으로 구현하기 위해서 TASK로 쪼게 진다. 이 task 목록은 이번 Sprint에서 구체적으로 해야 할 일을 정의하고 되고 이 목록 리스트를 **Spring Backlog** 라고 부른다.
Sprint Backlog의 task 들은 스케쥴이 지정되어 팀의 담당자에게 Assign된다.
팀은 이 Sprint Backlog를 가지고 정해진 기간동안의 Sprint를 구현한다.

Sprint중에 매일 아침 팀원들은 10~20분 정도의 **Daily Scrum Meeting** 이라는 회의 시간을 갖는데, 이 시간 동안, 팀원들은 어제 한일과 오늘 한일에 대해서 간략하게 보고하고, 기술적인 질의 사항에 대한 문의나 (팀원들의 도움을 받을 수 있는), RISK 사항등을 고용한다. 이 Daily Scrum Meeting을 통해서 팀은 현재 Sprint의 진행 상황을 공유할 수 있다.

Sprint가 끝난 후에 팀은 프로젝트의 stakeholder (고객)과 간단한 Review 미팅을 갖는데, 주로 이번 Sprint 에서 구현한 내용을 간단한 형태의 DEMO로 소개를 하고 Feed back을 받아서 Product Backlog를 업데이트 하고 다음 Sprint를 준비한다.

Review가 끝난 후에는 팀원끼리 Sprint에 대한 회고 (**Restrospective**)의 시간을 갖는데,
이 시간에는 Sprint에서 Scrum 방법론을 적용했을 때의 잘되었던 점과 잘못되었던 점을 토론하여 Scrum 운영 프로세스에 반영하여 팀의 Scrum 운영 방식을 성숙화 시킨다.

Role in Scrum

Scrum에서는 이 프로세스를 적용하기 위해서 몇 가지 역할을 정의하고 있다.

Product Owner

Product Owner는 요구 사항을 정의하고, Product Backlog를 업데이트 하는 역할을 맡고 있다. 가장 중요한 역할은 Product Backlog내의 Item 에 대한 우선 순위를 정하고, 정해진 구현 기간 내에 프로젝트 팀의 ROI (Return Of Investment)를 극대화 하는 책임을 가지고 있다. Product Owner는 직접 Sprint에는 참여하지 않고, Sprint 전에 Backlog의 업데이트후, Spring review에 구현 내용이 요구 사항에 맞춰서 적절하게 구현되었는지 확인하는 작업에만 참여한다.

Team

Team은 Product Backlog의 내용에 따라 Sprint에서 구현하는 역할을 맡는다.

Scrum Master

Scrum Master는 Project Manager도 아니고, 3자의 입장에서 Team과 Product Owner가 Scrum 방법론을 제대로 수행할 수 있도록 돕는 역할을 수행한다. 예를 들어 Product Owner가 Sprint중에 요건을 추가하려고 할 때 이를 제지하거나, Team이 Scrum을 수행하는 데 어려움이 있을 때 이를 돕는 역할을 한다. 종종 Scrum Master의 역할과 Product Owner의 역할을 한 사람이 겸임하는 경우가 있는데, Scrum Master의 역할중의 하나가 Product Owner가 Sprint중에 영향을 줄 수 있는 요건 변경을 막아야 하는 역할이기 때문에, 겸임할 수 없다.

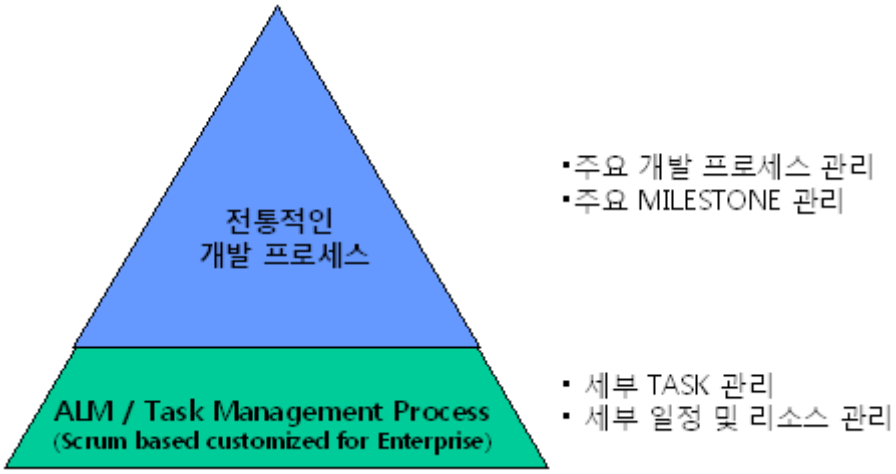
지금까지 Scrum 방법론에 대해서 간략하게 설명하였다. 그러나 이 Scrum 방법론은 태생이 제품 개발을 위한 프로세스로 개발되었다. (Product Owner와 같은 용어에서 알 수 있듯이..) 그래서 Requirement나 기능의 추가/삭제가 비교적 자유로운 In house 개발이나 패키지 솔루션 개발에서는 사용될 수 있을지 몰라도, 구현해야 하는 Requirement의 범위가 고정적인 Enterprise 프로젝트에는 적용하기가 어렵다. 그래서 다음은 경험을 바탕으로 하여

Customize 한 Enterprise 프로젝트에서의 Scrum 프로세스에 대해서 설명하고자 한다.

ALM / Task Management Process

Process Design Principals (프로세스 디자인 방향)

ALM / Task Management Process 의 기본 디자인 방향은 기존의 전통적인 개발 방법론을 대체하는 것이 아니라, 기존의 방법론은 전체 개발 프로세스를 커버하고, ALM / Task Management Process 는 Scrum 방법론에 기반 하여 각 단계에 대한 TASK 관리를 위해서 사용된다.



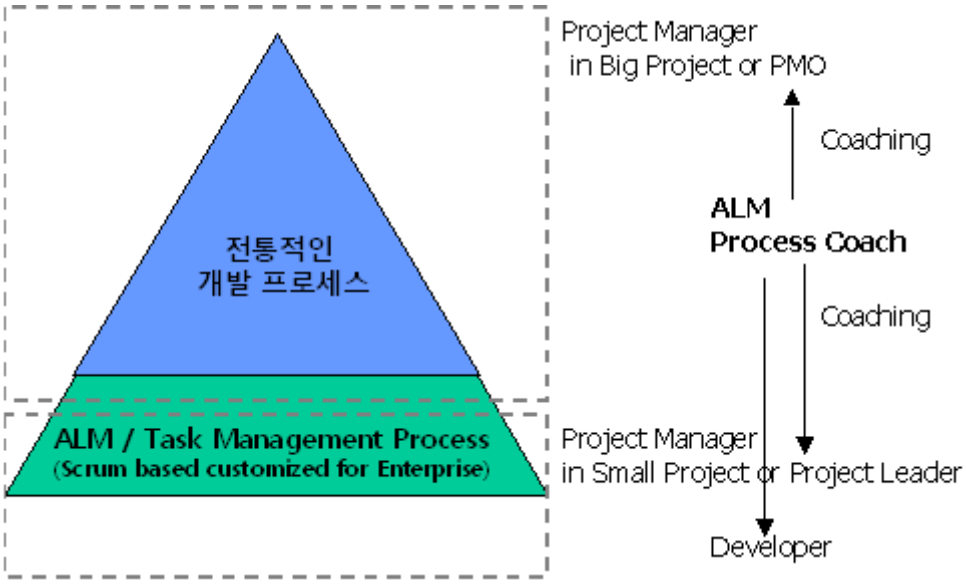
전체 프로젝트 관리로는 사용되지 않고, 각 수행 그룹에서 사용된다. (예를 들어 전체 프로젝트가, 고객 지원 업무, 빌링, 포탈 등으로 구성되어 있을 때 전체 프로젝트 관리는 전통적인 개발 프로세스에 기반하여 프로젝트가 관리되고, 빌링팀, 포탈팀 등 각 개발 단위팀에서는 ALM / Task Management Process 방법론을 이용하여 팀을 관리한다.)

Team Structure for ALM / Task Management Process

ALM/Task Management을 실제 팀에서 수행하기 위해서 팀 구조와 역할 정립이 필요하다.

- Project Manager : 전체 프로젝트를 관리 하는 역할을 한다. 큰 프로젝트의 경우 PMO나 여러 개발 단위의 PM을 정의하며 주로 전통적인 개발 프로세스를 기준으로 프로젝트를 관리한다. 단 ALM / Task Management Process의 개념을 이해하고, ALM / Task Management Process와 기존의 개발 프로세스를 연계 시키는 역할을 맡는다.
- Project Leader : 작은 조직이나 프로젝트에서는 PL이 없이 PM이 PL역할을 함께 하는 경우가 많으며, 실제 개발팀을 가지고 Implementation을 리드하는 사람이다. 실제로 ALM / Task Management process를 이용하여 팀의 Task를 관리한다.

- ALM Process Coach : 특이한 Role일지도 모르겠는데, Scrum의 Scrum Master와 같은 역할로 생각하면 된다. Process가 팀에 완전히 정착되기 까지는 팀에 대한 교육과 프로세스에 대한 지속적인 Mentoring이 필요하다. ALM Process Coach는 팀의 프로세스를 set up하고, 팀이 프로세스를 준수하도록 도우며, 개선하는 역할을 진행한다.



실제로 모 은행 프로젝트에서 ALM / Task Management Process를 적용해본 적이 있는데, PM이 두개의 팀을 운영하고 있었다. 그중에 A팀에 본인이 투여되었고, ALM / Task Management Process와 System을 구축하여 A팀에서 적용하여 효과를 보자 PM이 다른팀(B팀)에도 본 프로세스를 적용해 주기를 요청하였다. 그래서 System을 공유해줬으나, B팀에는 별도의 코칭을 제공하지 않았다. 프로세스를 제대로 준수하는지, 상태 업데이트는 제대로 하는지에 대한 코칭이 없었는데, 1개월후 A팀은 Task Management Process에 어느정도 적응을 하여, 효율적으로 만족 스럽게 사용하고 있었으나, B팀의 경우 이 프로세스와 시스템 사용이 또 다른 Burden이 되서 실업무 따로, 시스템 운영 따로 되는 결과를 만들어 냈다. 그만큼 ALM / Task Management Process를 적용하기 위해서는 성숙단계까지 지속적인 관찰과 코칭이 필요하다.

Step 1. Prepare Product Backlog

Product Back log는 실제로 구현되어야 하는 기능 목록을 나열한다.
목록은 고개의 요구 사항으로부터 도출 되는데, 일반적으로 SRS (Software Requirement Specification)이나 TRS (Technical Requirement Specification)로부터 도출이된다. 구체화 되고 정확하고 상세한 목록을 도출 해야 한다. 비즈니스 요건 같은 경우를 목록으로 도출하였을

경우에는 종료 조건이 모호해질 수 있고 그런 경우 종료에 대한 기준이 모호해 질 수 있다. Product Back Log의 항목은 다음과 같다.

NO	Action Item	Description	Estimated Resource (Man/Day)	WIKI	Priority	Status
Opened						
2	ALINT deployment architecture guide	Blar...	10	http://wiki/req/ALINT	Medium	In Progress
3	Weblogic deployment architecture guide	Blar...	15	http://wiki/req/req/WLS	Low	Opened
4	Configuration guide for RAC	Blar...	5	http://wiki/req/RAC	High	Opened
Closed						
1	ESB proxy Tuning	Blar...	20	http://wiki/req/Proxy Tuning	High	Closed

1) NO

ITEM에 대한 ID

2) Item

실제 구현 요건

3) Description

요건에 대한 간략한 설명

4) Estimated Resource

얼마나 많은 Resource가 소요되는 가에 대한 예측치를 기록한다. 이 값에 따라 추후 PLANNING에 반영한다. 이 값은 초기 예측 값으로 매번 Sprint가 종료될 때 마다 새롭게 업데이트 한다. 업데이트 기준은 기존에 수행한 Sprint의 Item의 실제 수행 시간을 기준으로 앞으로 수행할 Item의 상대적인 개발 난이도등을 측정하는 방법으로 예측할 수 있다.

5) WIKI URL

Back Log의 Item의 이름과 Description만 가지고는 정확한 요건을 알 수 없다. 그래서, 해당 Item에 대해서 정확한 요건 (Requirement description 또는 Use case)을 알기 위해서는 별도의 문서를 참조해야 한다. 문서는 MS-WORD와 같은 형태로 공유 파일 폴더에 들어가 있거나 Subversion과 같은 SCM에 저장이 되어 있거나 Wiki에 작성되어 있을 수 있다. 여기에는 Item과 요구 사항에 대한 추적성을 유지하기 위해서 해당 문서에 대한 위치 (파일 위치나, WIKI URL)을 기술한다.

6) Priority

Item에 대한 우선 순위를 지정한다. 이 우선 순위에 따라서 Sprint를 스케쥴링 하기 때문에 Product Back Log에서 매우 중요한 부분이다.

Priority의 설정은 비즈니스에 대한 영향이 큰 기능, 필수 기능, 그리고 난이도가 높은 기능, 비기능적 요건 등에 대해서 우선 순위를 높게 설정하는 것을 권장한다. 특히 비기능적 요건은 성능이나 안정성등에 관련되는 경우가 많고 이러한 요건들은 아키텍쳐에 많은 영향을 주고 아키텍쳐는 후반으로 갈수록 변경이 어렵기 때문에 초반에 구현 및 검증 작업을 수행해야 한다.

7) Status

Status는 해당 Item의 진행 상황을 의미한다. 이미 아직 진행전이지, 진행중인지 완료가 되었는지를 표현한다. 필드의 값은 프로젝트 상황이나 프로세스에 맞춰서 변화한다.

(QA 검증중, 운영 서버에 반영됨 등)

8) Estimated Value of item (Optional)

이 항목은 Item의 비즈니스적인 가치에 대해서 점수를 메기는 방법으로 SI개발보다는 제품 개발이나 In House 개발등에 유용한 항목이다. 이 값의 조정을 통해서 Product Owner는 정해진 기간내에 최대의 ROI (Return Of Investment)를 낼 수 있도록 Back Log의 우선 순위를 지정할 수 있다.

Step 2. Release Planning

해야할 일의 목록 (Product Back Log)가 정의 되었으면, 언제 어떤 일을 해야할 지를 정의해야 한다. 일반적으로 말하는 스케쥴링인데. Release Planning에서는 프로젝트의 큰 Mile stone을 정하는 작업을 한다. Release 시기 마다 작동 가능한 Product을 Release한다. (모든 기능이 완료되지 않았다 필 수 기능이 완성 되었으면 Release한다.) Release된 시점에서 QA 팀에게 Release version을 넘겨서 Testing을 수행하고 구현된 부분에 대한 품질을 보장 받는다. 이러한 활동은 모든 개발이 끝난 시점에서 Big bang 방식으로 통합하고 테스트 하는 기존에 방식에 비해서 위험을 조기에 발견할 수 있고, 그 위험을 해결하는 비용을 줄일 수 있다. (복잡한 문제일 수 록 나중에 발견되면, 더 고치기가 어렵다.)

그리고 이 Release version을 고객에게 DEMO를 통해서 요구사항과 부합하는 지 확인하고, 잘못된 부분에 대해서 수정할 수 있는 기회를 갖을 수 있다.

이 단계에서 해야하는 구체적인 작업은 다음과 같다.

1) 주요 릴리즈 일정 지정
2) 릴리즈 일정별 Product backlog 내의 Item 지정

이 단계에서 완성된 Release Plan은 Project 관리 입장 (Project Management Officer)에서 전체 스케쥴 WBS (Work breakdown structure)와 맵핑이 되서 관리팀 관점에서 스케쥴 관리를 용이하게할 수 있다.

Step 3. Sprint Planning

Release Planning이 끝난 후에는 각 Release를 달성하기 위해서 Release Planning을 Sprint로 나눈다.

전통적인 Scrum 방법론에서는 다음과 같은 절차로 Sprint를 계획한다.

- 팀원의 가용 시간
- Product Back Log Item을 구체적인 Task로 분할
- 각 Task 에 대해서 수행 시간을 예측

그러나 이 전통적인 접근 방법은 몇가지 문제를 가지고 있다. 먼저 Task 에 대한 수행 시간 예측후에 가용 인력에 Assign하는 방식인데, 이는 각 팀원의 능력이 동일함을 전제로 하고 있다.

그래서 실제 프로젝트에서는 다음가 같은 방법을 권장한다.

- Product Back Log Item을 Task 로 분할
- 각 Task를 팀장이 적절한 사람에게 배분
- 배분된 사람이 Task의 수행 시간을 예측 하도록 함

이 작업을 좀더 상세하게 설명해보면

Sprint는 보통 1주~4주 정도로 정의된다. Sprint의 기간을 정의할 때 기준중의 하나는 고객의 요구 사항 변경이나 작업에 대한 변경도가 얼마나 많은가? 예측된 작업 일정이 충분한가? 와 같이 불확실성이 높을 수 록 Sprint 주기는 짧게 잡는 것이 좋고, 불확실성이 적고 스케쥴이 안정적으로 정의될 수 있는 경우에는 길게 잡는 것이 좋다.

필자의 경우에는 보통 2주 정도를 Sprint주기로 관리를 한다.

Sprint를 정의할 때 먼저 Sprint 기간과, 가용 Resource를 바탕으로, Release Plan에 의해 정의된 Product Back log Item들의 예측 기간(Estimated Resource : Man/Day)를 기준으로 Sprint에 Product Back log Item 들을 할당한다.

해당 Sprint의 기간과 수행할 Back Log Item을 정의하는 일은 PM/PL이 수행하는 것을 권장한다.

Scrum 방식이 유연 해도, 스케쥴 자체는 프로젝트 진행에 있어서 가장 중요한 RISK FACTOR 중의 하나이며, 팀원간의 협의를 해서 진행한다 하더라도 고객 입장에서는 꼭 끝 맞춰야 하는 부분이기 때문에, 우선순위 지정과 스케쥴 조정은 그에 대한 책임을 지고 있는 프로젝트 관리자가 하는 것이 RISK 관리 측면에서 합리적이다.

선별된 Product Back Log Item은 실제로 수행되기 위해서 구체적인 TASK로 나뉘어진다.

TASK는 어떤 사람이 무슨 일은 한다는 구체적인 정의로, 명확한 종결 조건을 가지고 있어야 한다. 예를 들어 "Logging 기능의 설계" 는 언뜻 보면 적절한 TASK로 생각될 수 있지만 설계에 대한 종료 조건이 명확하지 않다. 즉 "Logging 기능 설계 문서를 WIKI에 업데이트" 또는 "Logging 설계 REVIEW 회의"와 같이 산출물이나 회의와 같이 어느정도 정해진 종결 조건을 정의하게 되면 TASK를 관리하기가 용이하다. ("Logging 기능의 설계" Task만 있다면 문제지 위에 예를 든 종료 조건이 다른 Task로 정의되어 있으면 괜찮다)

실제 TASK관리를 해보면, "구현 TASK"가 끝났다고는 하는 경우 개발자 본인의 역량에 따라서 Coding만 되고 실제로 기능이 작동하지 않거나, 의도하지 않은 설계대로 구현되어 있는 경우가 생각보다 많기 때문에, 이런 경우 프로젝트를 관리하는 입장에서 "구현 보강" 이라는 새로운 TASK를 만들고 새롭게 RESOURCE와 시간을 할당해야 하는 부담을 가지게 되기 때문에, TASK에 대한 종료 조건을 명확히 하는 것은 매우 중요하다.

1:1:1 법칙

Task를 정의하는데 가이드를 제시하면, Product Back Log Item은 분석/설계, 구현, 테스트 이 3가지로 크게 분리될 수 있다. 어떤 구현 테스트를 하기 위해서 요건에 대한 분석 및 설계가 필요하다. 비록 미리 다 설계가 되어 있는 부분이라도, 실제 구현에 있어서는 국지적인 설계 변경이 필요하거나, Prototyping (설계 검증을 위한) 들이 필요하기 때문에 분석/설계에 대한 시간은 소요된다.

다음으로 테스트는 구현된 내용을 바탕으로 검증을 해야 하는데, 특히 비기능적인 요건은 시스템 테스트를 하지 않더라도 최소한 자기 PC에서 성능 테스트 (이를 Micro benchmark test라 한다.)를 하고 단위 테스트를 수행해야 하고, 구현이 아닐 경우라도 설계나 요건 분석 등의 문서상 산출물은 REVIEW 시간을 필요로 하기 때문에, 일반적으로 분석/설계, 구현,테스트에 대한 시간 및 Resource 할당 비율은 1:1:1이 된다.

Task 수행의 위의 3단계가 종료될 때 마다 주요 Task에 대해서는 어떤 형식으로든지, 리뷰 회의를 갖는 것을 권장한다. (Peer Review, Team Walkthrough etc)

20% 버퍼의 법칙

이렇게 Task list를 도출하고 나면 각 Task에 수행 시간과 담당자를 지정해야 한다.

이 Task list 도출과 이 과정은 팀원들과 함께 수행되어야 하는데, 팀원들의 능력과 근무 가능 시간이 각각 다르기 때문에 팀원의 의사가 매우 중요하다. 팀원과 미팅을 통해서 해당 Task를 수행 가능한 사람에게 Assign 하고, Assign을 받은 사람과 수행 시간을 논의하여 결정한다. 되도록이면 Assign 받은 사람의 수행 시간에 대한 결정과 의견을 존중하는 것을 권장한다. 실제 Task에 대한 시간을 Estimation해보면, 경험상으로 충분한 시간으로 Task 수행 시간을 Assign 하게 하고 Buffer율을 20%를 두도록 해도, 실제 수행해보면 그보다 20~50%의 시간이 모자라는 경우가 태반이고, 이는 초반 프로젝트 스케쥴 관리에 Risk factor가 된다. 보통 1~2개월간의 Sprint 기간에는 이런 스케쥴 예측에 대한 오차 범위가 크게 나타나는데 이를 반영해서 일정을 잡고, 2개월 후에는 팀의 스케쥴 측정에 대한 경험이 쌓여서 점점 더 정확하고 세밀한 스케쥴 관리가 가능하게 된다.

이렇게 만든 Sprint의 Task 목록, 예측 시간, 담당자의 목록을 **Sprint Back Log** 라고 하고 다음과 같은 형태로 작성이 가능하다.

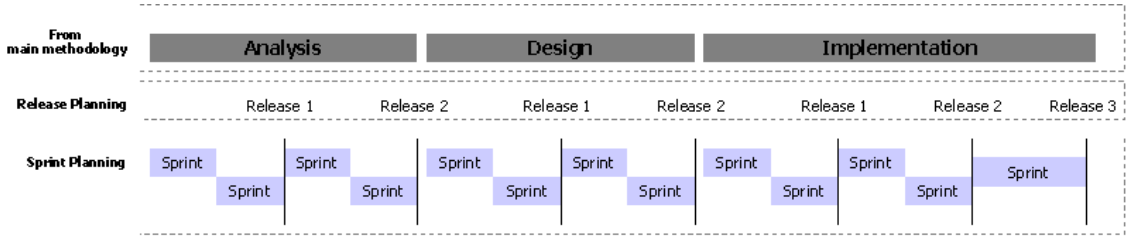
						Feb											
NO	Action Item	Task	Estimated Resource	Owner	Status	1	2	3	4	5	6	7	8	9	10	11	12
1	AI 1. ESB Proxy tuning	Build or review testing plan	3	mhjeong	Closed	3	2	1	0								
2	AI 1. ESB Proxy tuning	Execute load testing	4	bcho	Closed	3	3	2	1	0							
3	AI 1. ESB Proxy tuning	Gather performance data and find bottleneck point	9	bcho	In Progress	9	9	9	9	9	8	7	6	5			
4	AI 1. ESB Proxy tuning	List up bottleneck point and prioritizeit	8	fenton	In Progress	8	8	8	8	7	6	5	3	2			
5	AI 1. ESB Proxy tuning	Develop alternative solution	3	fenton	Closed	3	2	1	0								
6	AI 1. ESB Proxy tuning	Prototype alternative solution	4	andy	Closed	4	3	2	1	0							
7	AI 1. ESB Proxy tuning	Micro benchmark prototype solution	5	andy	In Progress	5	5	5	5	5	4	3	2	1			
					Remaining Time	35	32	28	24	21	18	15	11	8			

Sprint Back Log : Sprint Planning 단계에 작성됨

위와 같이 엑셀을 사용해서 Sprint 스케쥴을 관리할 경우에는 Task의 스케쥴을 우측에 날짜

박스를 만들어놓고, 수행 기간을 칠해 놓는 방식으로 그래프 형태로 스케쥴을 관리할 수 있다.

Release Planning과 Sprint Planning을 정리해보도록 하자.
기본 원리는 큰 스케쥴을 작은 단위의 스케쥴로 나누어서 관리하는 방법으로, 기존의 전통적인 방법론이 가지고 있는 스케쥴 단계를 Release Plan으로 나누어서 관리한다.
Release Plan은 PMO에서 관리될 수 있는 단위의 프로젝트 주요 Mile stone이 되며, 각 Release Plan은 실제 개발팀이 수행할 하루 단위의 개인 스케쥴로 정의되는 Sprint Plan으로 나뉘어서 관리 된다.

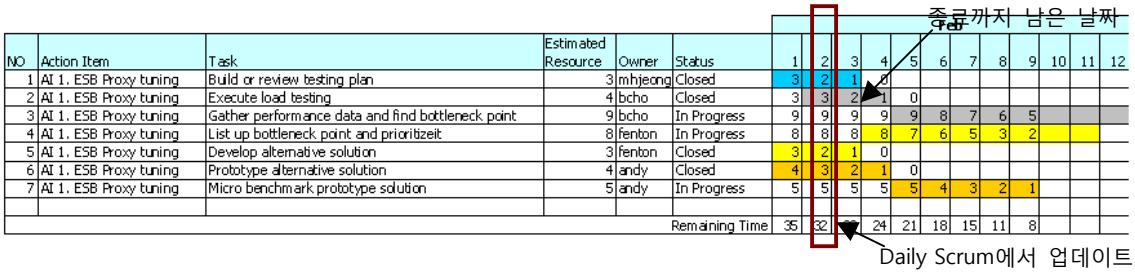


Step 4. Sprint Tracking

Sprint 계획이 끝났으면, 계획에 따라서 프로젝트를 수행한다.
Sprint의 Task 진행 상황을 추적하기 위해서 몇 가지 기법을 지원하는데 다음과 같다.

Daily Scrum

Daily Scrum은 일일 오전 업무 공유(보고가 아닌) 회의이다., Scrum에서 가장 유용하고 중요한 기법 중의 하나이다.
Scrum 팀은 매일 오전에 같은 자리에 모여서 **어제 자신이 한일과 오늘 자신이 해야할 일을 짧게 발표한다. 전체 회의 시간은 30분을 넘지 않도록 한다. 이와 함께 자신이 진행하고 있는 Task를 Close하는데 필요한 시간을 같이 이야기 한다.**
이 과정에서 팀원은 다른 팀원의 Task 진행 상황을 Share할 수 있고, 만약 일의 진행에서 문제가 있는 부분이나 도움을 받고 싶은 부분은 이 회의 과정에서 이슈로 제기하여 다른 사람의 도움을 받거나 PM이 일정을 조정할 수 있도록 한다.
PM 관점에서는 Daily Scrum을 통해서 지정된 Task의 진행 상황을 매일 추적할 수 있다.
PM은 회의에서 공유된 일정을 바탕으로 Sprint Back Log 를 업데이트 하는데, 중요한 점은 각 Task의 종료시까지 남은 시간을 반드시 기록한다. 이 데이터는 처음 계획 대비 현재 상황이 어떻게 되고 있는지를 판단할 수 있게 해주는 중요한 지표가 된다..



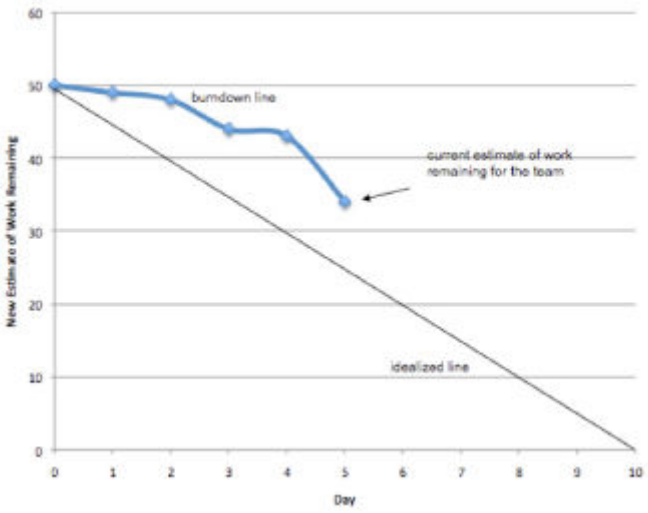
NO	Action Item	Task	Estimated Resource	Owner	Status	1	2	3	4	5	6	7	8	9	10	11	12
1	AI 1. ESB Proxy tuning	Build or review testing plan	3	mhjeong	Closed	3	2	1	0								
2	AI 1. ESB Proxy tuning	Execute load testing	4	bcho	Closed	3	3	2	1	0							
3	AI 1. ESB Proxy tuning	Gather performance data and find bottleneck point	9	bcho	In Progress	9	9	9	9	9	8	7	6	5			
4	AI 1. ESB Proxy tuning	List up bottleneck point and prioritizeit	8	fenton	In Progress	8	8	8	8	7	6	5	3	2			
5	AI 1. ESB Proxy tuning	Develop alternative solution	3	fenton	Closed	3	2	1	0								
6	AI 1. ESB Proxy tuning	Prototype alternative solution	4	andy	Closed	4	3	2	1	0							
7	AI 1. ESB Proxy tuning	Micro benchmark prototype solution	5	andy	In Progress	5	5	5	5	5	4	3	2	1			
					Remaining Time	35	32	27	24	21	18	15	11	8			

이상적인 프로젝트 진행이라면 위의 Sprint Planning (칠해진 부분)이 끝나는 다음날 남은 작업 시간이 0 이 되어야 한다. 만약 연장 작업이 계속 되더라도 칠해진 블록의 범위는 변경하지 않고, 남은 날짜만 계속 업데이트 해간다. 그러면 추후에 결과가 계획 대비해서 얼마나 초과되어서 끝났는지를 확인할 수 있다.

진행중에 고객 요건 변경이나 구현의 난이도에 의해서 스케쥴을 바꿔야 할 경우가 있는데, 전통적인 Scrum 방법론에서는 Sprint가 한번 계획된후에 Sprint 중간에는 바꾸지 못하도록 가이드 하고 있다. 변경이 생겼을 경우 Sprint를 중지하고 다시 Sprint 계획을 하도록 하는데, 국내 SI프로젝트와 같은 상황에서는 이를 매우 통제하기가 어렵기 때문에, 개인적으로는 융통성 있게 Task 를 추가하거나 일정을 변경하는 것을 반영하는 것을 권장한다. 단. Sprint 전체 스케쥴에 최소한의 impact를 줄 수 있는 범위내에서 변경을 하되, 새롭게 발생한 Task 가 있을 때 그만큼 다른 Task에 대한 일정을 미루는 것을 전재로 해야 한다.

Burn down chart

Sprint Product Back Log를 위와 같은 방법으로 Update하면, 매일 남은 작업 일 수를 계산할 수 있는데, 이를 그래프로 표현한 것을 Burn down chart라고 한다.



이상적인 Burn down chart는

(remaining time)=(total work)/(current date)

의 형태를 띄어야 한다. 그리고 실제 프로젝트의 remaining time역시 이 이상적인 곡선에 근접해야 하는데, Burn down chart를 매일 업데이트 함으로써, 이상인 그래프에서 실제 프로젝트의 remaining time 그래프가 얼마나 벗어 나는가를 측정함으로써 프로젝트의 일정상 위기 요소를 파악하고 대비할 수 있다.

이 Burn down chart는 프로젝트 룸의 칠판이나 또는 시스템의 Dash board나, Daily Scrum 을 통해서 공유되는 것이 좋은데, 프로젝트 팀원 역시 사람이고, 사람은 Visual 한 개념을

통해서 현재 상태를 파악하고 프로젝트의 위험도를 좀더 체감할 수 있기 때문에, Risk 요소에 대해서 심각도를 서로 공유하는데 잘 사용될 수 있고 이는 실제 팀의 사기에 영향을 줄 수 있다.

Task Status

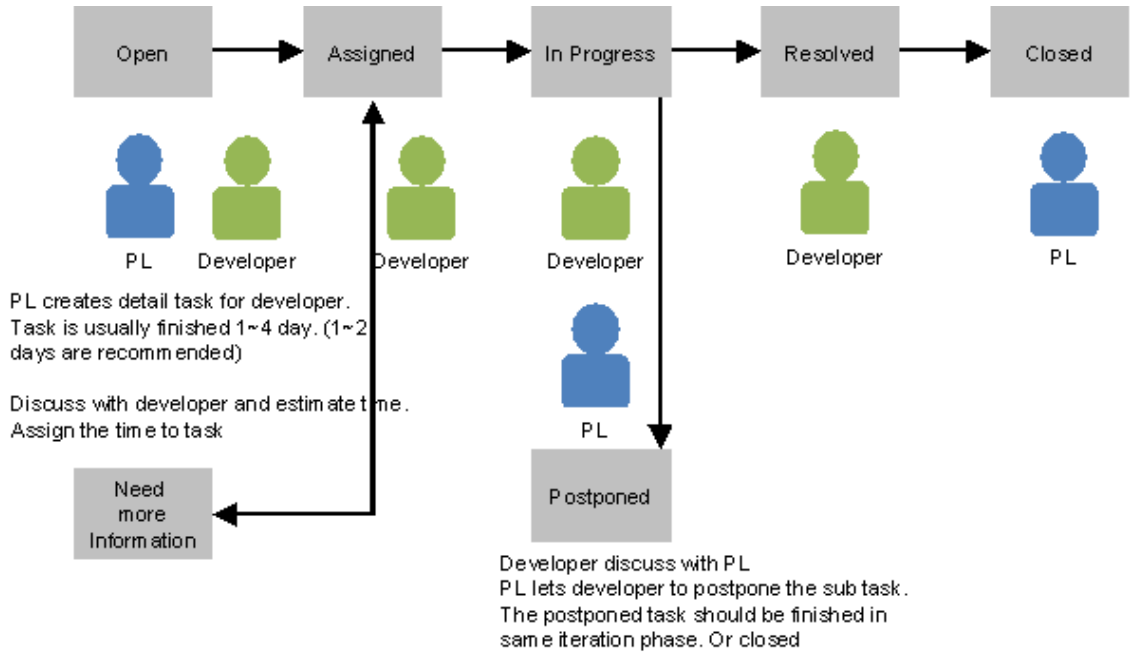
다음으로 각 Task에 대해서 상태를 관리해야 한다. 위의 예시에서 보여준 Sprint Product Back Log의 Status 부분을 말하는데, 이 부분은 사실 매일 Daily Scrum 미팅을 하고 엑셀로 프로젝트를 관리하는 경우에는 크게 중요하지 않지만, 좀더 전문화되고 정교한 Task 관리 절차를 만들거나 시스템으로 구축하고자 할때는 제일 필수적인 부분이다.

시스템으로 구축할 경우에는 해당 TASK를 Daily Scrum 때 뿐만 아니라, 항상 상태를 업데이트 해서 팀원이나 운영 조직간에 자주 의사 소통을 할 수 있고, 저장된 상태나 값을 기반으로 여러가지 지표를 뽑아 낼 수 있다.

Task Status는 Task의 상태 흐름에 따라 정의 되는데 이를 Task Workflow라고 한다. Task Workflow는 최대한 단순해야 하며, 관리 지향적이기 보다는 실무 중심적이어야 한다.

또한 Workflow가 적용되는 업무의 특성별로 잘 정의되어야 한다. (개발의 Task workflow와 Bug의 Task workflow는 분명히 달라야 한다.)

다음은 Task workflow의 예이다.



- Open (생성됨) 새롭게 생성되고 정의된 TASK로, 담당자가 지정되지 않은 TASK이다.
- Assigned (할당됨) TASK가 담당자와 협의를 거쳐 지정되게 되고, 수행 시간이 ASSIGN되고 스케줄링 된 상태이다. 이때 반드시 담당자가 TASK의 내용을 이해하고

자신에게 ASSIGN되었다는 것을 인지해야 한다.

- Need more information (추가 정보 필요) 만약에 할당 받은 Task의 내용이 명확하지 않거나 추가 정보가 필요할 경우 Need more information으로 상태를 바꾸고 PM/PL에게 Task의 재정의를 요청하는 상태이다.
- In Progress 지정된 (진행중) TASK를 개발자가 작업을 시작했을 때 "진행중" 인 상태를 나타낸다.
- Postponed (연기됨) 진행중인 작업이 어떤 이유(Product Bug, 환경 준비 미비)로 인해서 더 이상 진행할 수 없을 때, PM과 상의하여 해당 Task를 "지연" 상태로 변경한다.
- Resolved (해결됨) 개발자가 할당된 TASK를 완료하고 종료조건을 충족하였을 때, "해결됨" 상태로 바꾸고 PM에게 작업이 완료되었음을 알린다.
- Closed (종료됨) PM은 개발자가 완료한 작업을 TASK의 지시사항대로 제대로 종료되었는지 확인하고, 종료조건을 충족하여 제대로 완료되었을 경우에는 "종료" 상태로 바꾸거나, 만약 제대로 종료가 되지 않았으면 상태를 Assigned로 바꿔서 다시 개발자에게 할당한다.

Step 5. Ending Sprint

정해진 기간 동안 Sprint가 종료되면, Sprint를 정리 한다.
Sprint의 종료 조건은 팀에 따라 정할 수 있는데 크게 아래 조건들을 많이 사용한다.

- 정해진 시간
- Task 종료
- 예산 종료시 까지

어떤 조건을 사용하던지, 가장 중요한 것은 명시적인 종료 조건을 정의 해야 한다는 것이다.

좋은 종료 조건의 예

1) 구현의 경우 : 단위 테스트 100% 성공, Code Coverage Rate 60% 달성
2) 설계 단계 : 메시지 채널 설계 문서 고객 확인 받음
3) 설계 단계 : Prototype 작성 및 A,B,C,D 기능 동작 확인
4) 기간 BASE 경우 : 3/19일까지 코드 분석된 내용 리뷰 (이 경우, 컨설팅이나 Code Inspection, 탐색적 테스팅 같이 특정 기간과 예산을 가지고 진행하는 경우에는 기간 단위의 종료 조건을 사용하는 경우도 있다.)

Review Sprint

Sprint가 종료된 후에는 Sprint에서 구현된 산출물을 Review하는 단계가 필요하다. 요건에 따라 적절하게 구현이 되었는지 품질은 만족해야 했는지 등의 검증이 필요하다.
단순히 Sprint에서 무엇 무엇을 했고, 잘됐다 안됐다가 아니라, 실제 구현 코드, 산출 문서, 테스트 결과등의 구체적인 자산을 가지고 Review를 수행해야 한다.

이 과정에서는 고객을 참여 시켜서, 자산에 대해서 간략한 데모를 고객에게 수행한다. 이 데모는 고객 보고를 위해서 별도의 PPT나 데모 준비를 하는 것이 아닌 Informal한 리뷰이다. Formal한 리뷰는 Release 시기별로 진행하도록 하고, Sprint Review 준비를 위해서 별도의 시간이나 Resource를 낭비하지 않도록 한다.

Test

특히 Sprint가 Implementation인 경우, Implementation이 제대로 완료되었는지 아닌지를 확인할 수 있는 방법은 Testing 밖에 없다.

Sprint Planning에서 Implementation의 경우 Test Task를 반드시 포함시켜야 하고, Review과정에서는 이 Test 결과를 Review하도록 한다. Test는 기능적 테스트 뿐만이 아니라 , 안정성이나 성능과 같은 비기능적 요건에 대한 Test도 반드시 포함되어야 한다.

Sprint별 테스트를 통해서 잠재적인 문제를 빨리 찾아낼 수 있고 필요에 따라서 디자인이나 아키텍쳐에 대한 변경을 가할 수 있다.

Scrum을 사용하는 실제 구현 팀에 대해서 테스팅 구현은 다음과 같은 구현을 권장한다.

- 단위 테스트 (Unit Test) : 개발자가 개발 컴포넌트 단위로 테스트 수행
- 회귀 테스트 (Regression Test) : 테스트 했던 내용을 다음 테스트에도 포함 시켜 새로운 코드 추가나 변경이 기존 기능에 영향을 주지 않는 지 매번 검증
- 테스트 자동화 (Test Automation) : 테스트를 자동화 하여 회귀 테스트를 지원하고, 테스팅의 효율성을 극대화 함
- 점진적 통합 (Contiguous Integration) : 일일 빌드등을 통해서 BIG BANG방식의 코드 통합이 아닌 점진적 통합 전략을 사용함

[!] 위의 내용은 본 문서에서 다루고자 하는 범위가 아니기 때문에 좀더 자세한 부분은 ALM/Test Automation 과 ALM/Build Automation 을 참고하기 바란다.

Code Review

Code Review는 완성된 산출물을 같이 Review하여 Defect를 찾아내고, 좀더 좋은 개선 방안에서 의견을 수렴하는 자리다. 일반 Review와 달리, 특정 산출물을 가지고 진행한다는 것이 다른데, 주로 Design이나 Implemented code와 같은 구현에 밀접한 산출물을 가지고 진행한다.

Code Review는 원래 Code Inspection이라는 개념에서부터 시작했다. Code Inspection 전문적인 지식을 가지고 있는 팀이 정형화된 프로세스에 따라서 Code를 Review하고 Defect를 발견하는 과정을 정의 한다. Code Inspection은 Assembly 로 코딩하던 시절에 시작되었고, 요즘과 같이 많은 코드와 변경 사항이 있는 시스템에서는 적용하기가 적절하지 않다. 그래

서 좀더 비형식적인 Code Review 형태가 개발되었는데, Team Review, Walkthrough, Peer Review등이 그것에 속한다.

[!] Code Inspection은 특정 룰을 바탕으로 하기 때문에 자동화된 툴을 통해서도 어느 정도 효과를 볼 수 있다. Code Inspection을 위한 팀과 프로세스를 구성하는 것보다 Test 과정에 Inspection 도구를 사용하면 최소한의 비용으로 효과를 볼 수 있으며, Open source 기반의 Inspection 도구로는 PMD, FindBugs를 권장한다.

어떤 형태로건, 비형식적인 Code Review는 준비 시간이 짧고, 그에 비해서 Defect나 시스템 개선에 대한 효과가 상대적으로 크기 때문에, Sprint 말에는 Code Review를 수행하는 것을 권장하며, Sprint 중간에라도, 구현 Task 가 끝났을때는 비정기적으로라도 Code Review를 수행하는 것을 권장한다.

[!] 위의 내용은 본 문서에서 다루고자 하는 범위가 아니기 때문에 좀더 자세한 부분은 ALM/Collaboration 을 참고하기 바란다.

Code Review를 진행함에 있어서 중요한 점은 Code Review도 Resource와 시간이 투여되는 작업인 만큼 하나의 Task로 정의되어 관리 되어야 한다.
Code Review시에는 Code Review 결과를 Wiki등에 Meeting Minutes (회의록) 형태로 기록해야 하며, **Review 과정에서 나온 내용들은 Action Item으로 정리되어 Sprint Back Log내에 Task로 생성되고 관리되어야 한다**.

Step 6. Update product backlog

Review가 끝났으면 Review 과정에서 나온 추가 요건이나 변경 상황을 반영하여 Product Back Log를 업데이트 한다. 팀이 모여서 우선 순위를 재 조정하고, 요구 사항을 좀더 구체화 하며, 예상 소요 시간을 업데이트 한다.
사람들은 프로젝트를 수행하면서 배우게 되고, 실력적으로 진화하게 된다. 그래서 요구 사항이나 예측치는 그때 상황에 따라 계속 업데이트 되어야 한다. Product Back Log 는 고정된 것이 아니라 항상 변경되고 실상황을 반영해야 한다.

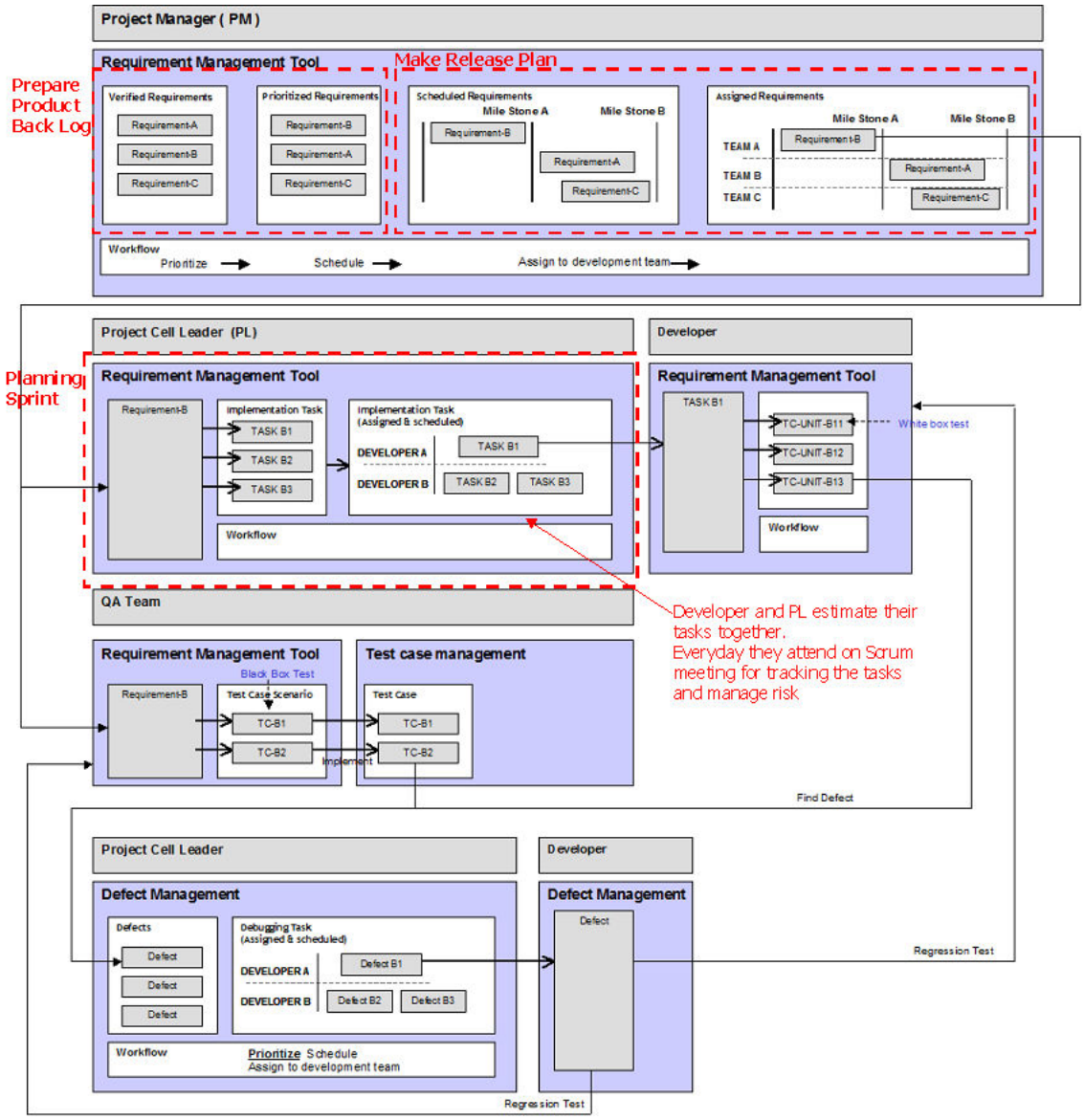
Step 7. Retrospective

Sprint가 종료되고 모든 작업이 끝나면, 팀에서 운영중인 방법론 자체에 대한 Review가 필요하다. 팀에서 운영중인 Task Management Process는 처음에 셋업 되면 많은 시행 착오를 겪게 된다. 수행 시간 예측이나, Sprint 주기 설정, Task 관리 프로세스 등등 많은 과제들이 있

는데 Sprint의 경험을 기반으로 회고를 수행함으로써, 프로세스를 발전시킬 수 있는 방향을 찾을 수 있다.
가장 간단한 수행 방법은 PM이 이메일이나 종이를 이용하여 이번 Sprint에 "무엇이 잘되었는가?" 와 "무엇이 잘못되었는가"를 수집한다음에 SWOT 분석 (http://en.wikipedia.org/wiki/SWOT_analysis) 등을 통해서 개선 방안을 찾아볼 수 있다.

Process Summary

지금 까지 설명한 프로세스를 개념을 한 그림으로 정리해보면 다음과 같다.



Implementation

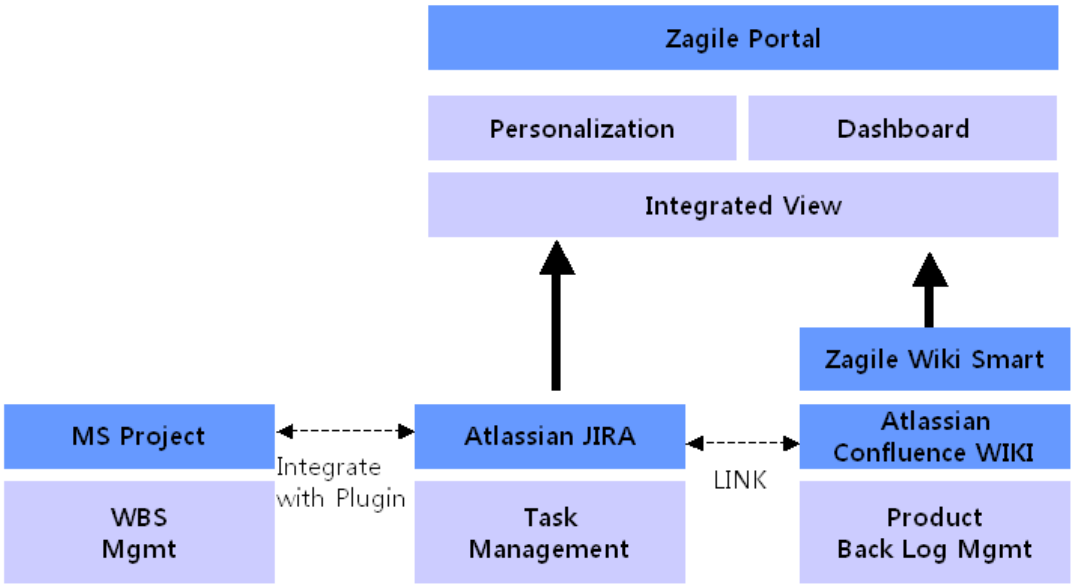
실제로 Task Management Process를 구현하는 몇 가지 방법에 대해서 설명한다. 가장 중요한 점은, 어떤 툴을 사용하느냐가 아니라 어떤 프로세스를 사용하느냐이다.
종종 ALM / Task Management Process를 현업에 적용해보면 문제가 프로세스에 대한 인식이나 성숙된 적용이 없이**, 툴을 도입해서 툴에 이끌려가는 경우가 많다**. 그래서 프로세스 자체가 또 다른짐이 되고 실제 업무 따로 Task 관리가 따로 이루어 지는 경우를 많이 봤다.
어떤 툴을 사용한다고 해도, 조직과 업무에 맞는 현실적이고 실행 가능한 프로세스를 사용하는 것이 가장 중요하며, 이를 위해서는 1~2개월 정도의 시행 착오 기간이 생길 수 있고 가급적이면 이 기간 중에는 프로세스를 관리하고, 이해 당사자들이 제대로 프로세스를 준수하는 지를 챙길 수 있는 Coach를 두는 것을 권장한다.

Jump start with Excel

소규모의 팀에서 이 Task Management Process를 사용하기 위해서는 툴과 도구를 설치하는 시간에 대한 소모를 줄이기 위해서 Microsoft Excel만을 사용해도 충분하다. 보통 단일팀에서 10명 이하라면 약간 불편하긴 하지만 빠르게 방법론을 적용할 수 있고, 만약에 시스템을 적용하더라도, 가장 중요한 것은 프로세스이기 때문에 구축 전에 미리 Excel을 이용하여 프로세스를 적용해보고 단계적으로 발전 시켜가는 것을 권장한다.
Excel Template은 http://bcho.tistory.com에서 다운 받을 수 있다.

Implement Task management process with Atlassian Product

회사내에 자체적으로 ALM을 구축하고자 할때는 Atlassian Product를 추천한다.
Atlassian은 JIRA라는 Issue tracking 시스템을 가지고 있는데, 태생 자체가 Task 와 같은 프로젝트 관리 관점 보다는 **Bug Tracking 관점의 접근이기 때문에 툴과 Process에 대한 Customizing이 필요하다** 그러나 상대적으로 가격이 매우 저렴하고, (Professional Version이 250만원선) 전세계적으로 매우 널리 사용되기 때문에, 관련 자료나 Plug in 들이 많다.
또한 CI나 테스팅 툴, SCM 도구들과 잘 통합이 되는 장점도 가지고 있다.
아래는 ALM/Task Management process 구축을 위한 Atlassian 제품을 이용한 권장 구축 아키텍쳐 이다.



1) Task Management : Sprint에서 관리되는 각종 Task를 관리한다. Atlassian의 JIRA(이슈 트랙킹) 시스템으로 구축하되, Product BackLog Item과 Sprint Back Log Item이 서로 Parent/Child 관계로 맵핑이 되어야 하기 때문에 Issue에 대한 Parent/Child 관계는 JIRA Professional Version 이상에서만 지원한다.
2) Product Back Log Mgmt : Product Back Log에는 Product Back Log Item들과 그에 대한 자세한 설명 (예를 들어 Use case)이 들어간다. Product Back Log Item은 JIRA 내에 Parent Task로 등록해서 관리할 수 있기 때문에 이에 대한 연관 관계를 JIRA내에 확장 필드로 등록하여 Wiki와 JIRA 사이에 상호 추적이 가능하도록 구축한다.
   그 위에 Zagile의 Wiki Smart라는 플러그인을 사용할 수 있는데, Wiki Smart를 사용하면, Product Back Log Item 입력등을 위키 형식이 아니라 좀 더 정형화된 폼을 이용해서 사용할 수 있다.
3) WBS Mgmt : 전체 프로젝트의 WBS를 관리할 때 통상적으로 사용하는 도구가 MS Project이다. http://www.the-connector.com/ 에서 제공하는 플러그인은 MS-Project의 Task와, JIRA의 Task를 연계 시켜준다. 이 도구를 이용하여, MS-Project를 이용한 WBS 관리와 JIRA를 통한 실제 Task 관리를 연동 할 수 있다.
4) Integrated view : JIRA와 Wiki는 각각의 View를 가지고 있기 때문에 한눈에 원하는 화면을 보기도 어려울 수 있고 이로 인해서 툴의 사용 효율성이 떨어질 수 있다. zAgile의 Portal 도구는 Atlassian의 툴을 하나의 View로 통합할 수 있는 기능을 제공한다.
5) Personalized view : 각 이해 당사자 Customer/PM/PL/개발자 별로 봐야 하는 화면이 틀리다. Customer나 PM은 프로젝트의 진척도나 위험도등의 간단한 Dash board를

보고 싶어하며, 개발자는 자신에게 할당된 작업상황에 대한 모니터링을 원한다. zAgile Portal은 개인화 기능을 제공함으로써 역할별로 원하는 화면을 만들 수 있도록 한다.

Implement Task management process with VersionOne

다음으로 Version One (http://www.versionone.com)이라는 전문화된 Task Management 도구가 있다. AUP나 Scrum과 같은 Agile 방법론을 기반으로 하고 있기 때문에 본 Task Management Process에도 적용이 잘 된다.

또한 패키지 판매와 함께, Hosting service도 제공하기 때문에 프로젝트 수행후 빠지는 SI 성 프로젝트에서 임시로 사용하기에는 그만이다. (1 Project의 경우 제한된 인원 내에서 무료로 사용할 수 있으니,작은 조직에서 툴을 Evaluate 할 경우 추천한다.)

단 Issue tracking에 주 목적을 두고 있기 때문에 ALM의 Build Automation이나 Test Automation등의 통합에 문제가 있을 수 있다.

Conclusion

Agile 방법론이 아무리 실용성을 강조하더라도, Risk관리 측면이나 일정 비용을 가지고 프로젝트를 진행하는 현실에서는 Agile의 목표가 불 명확한 방법론은 적용하기가 어렵다. 그래서 전체 프로젝트에 대한 큰 Mile stone은 기존 방법론을 적용하고, 그 아래에서 Agile 방법론을 팀의 Task 관리 기법으로 사용하도록 디자인 된 것이 ALM / Task Management Process이다.

ALM/Task Management Process를 실제로 구현하는데 가장 중요한 것은 어떤 전문화된 시스템이나 도구가 아니라, 팀이 최대한의 효율을 낼 수 있는 프로세스이다. 도구는 프로세스를 도와주기 위해서 존재하는 것이지, 도구에 프로세스나 팀이 이끌려 가서는 안된다.

Reference

Methodology

1) SWOT Analysis : http://en.wikipedia.org/wiki/SWOT_analysis
2) Scrum in 5 minutes : http://www.softhouse.se/Uploades/Scrum_eng_webb.pdf
3) Scrum primer : http://scrumtraininginstitute.com/home/stream_download/scrumprimer

Tools

1) Atlassian : http://www.atlassian.com
2) JIRA : http://www.atlassian.com/jira
3) Confluence Wiki : http://www.atalssian.com/confluence
4) JIRA & MS Project Connector : http://www.the-connector.com/
5) zAgile : http://www.zagile.com
6) VersionOne : http://www.versionone.com