Lean에 대해 알아보자

@pineoc
이윤석

Index

1. Lean?

2. Kanban

3. Appendix A, Lean Software Development

Lean?

Lean?

- 사전적 정의: 야윈, 마른, 절감한
- **Lean Manufacturing(Wiki)**
  - From Toyota Production System(TPS)
  - **Waste** Minimization
  - **Value** Maximization

Lean

maximize `customer value` while minimizing `waste`

Lean, Waste

아래의 세 가지 Waste는 TPS의 Key concept이다.

`TPS = Toyota Production System`

- Muda(Uselessness): 낭비, 비 부가가치 활동
- Mura(Unevenness): 공평, 흐름의 가변성
- Muri(Overburden): 과중한 부담

Lean, Waste

Muda: 낭비, 비 부가가치 활동을 의미함

* Transport: 제품 수송에 드는 비용
* Inventory: 제품 저장에 드는 비용(재고)
* Motion: 제품, 이동
* Waiting: 대기
* Over-production: 과다 생산
* Over-processing: 과다 가공
* Defects: 불량

Eliminate waste

`Waste(Muda)`를 해결하는 방법(Countermeasures)

`Transport` - Value Stream Mapping, Continuous Flow
`Inventory` - JIT, Continuous Flow, Kanban(Pull System)
`Motion` - Value Stream Mapping
`Waiting` - Continuous Flow
`Over-production` - Takt Time, Kanban(Pull System)
`Over-processing` - Kaizen
`Defects` - Standardized Work

Kanban

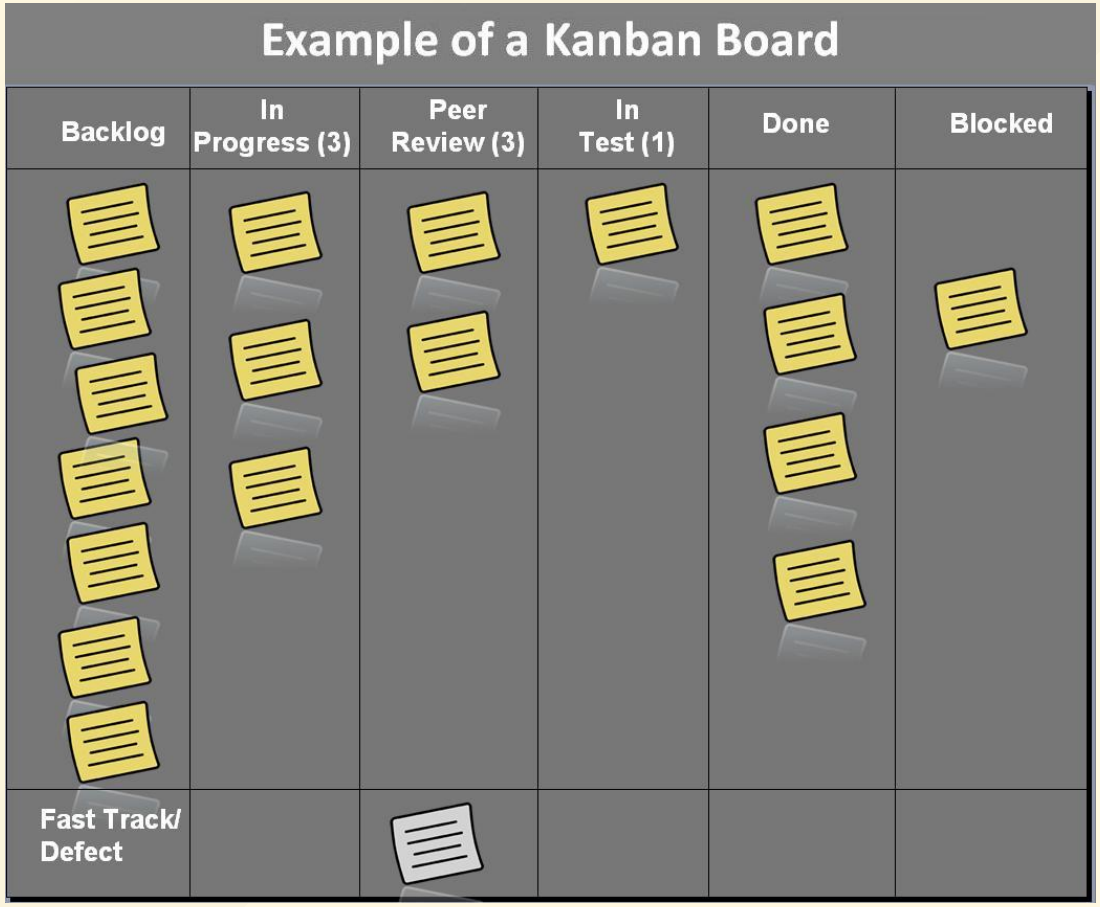
Pull System

Kanban

- 간판(看板), Visual Card
- 프로세스 도구
- 일을 작은 조각으로 나누고, 카드에 각 항목을 기입한 후 벽에 붙인다.
- `WIP` 개수를 제한한다. (WIP = Work In Progress)
- `리드타임(lead time)`을 측정한다.
- Wiki

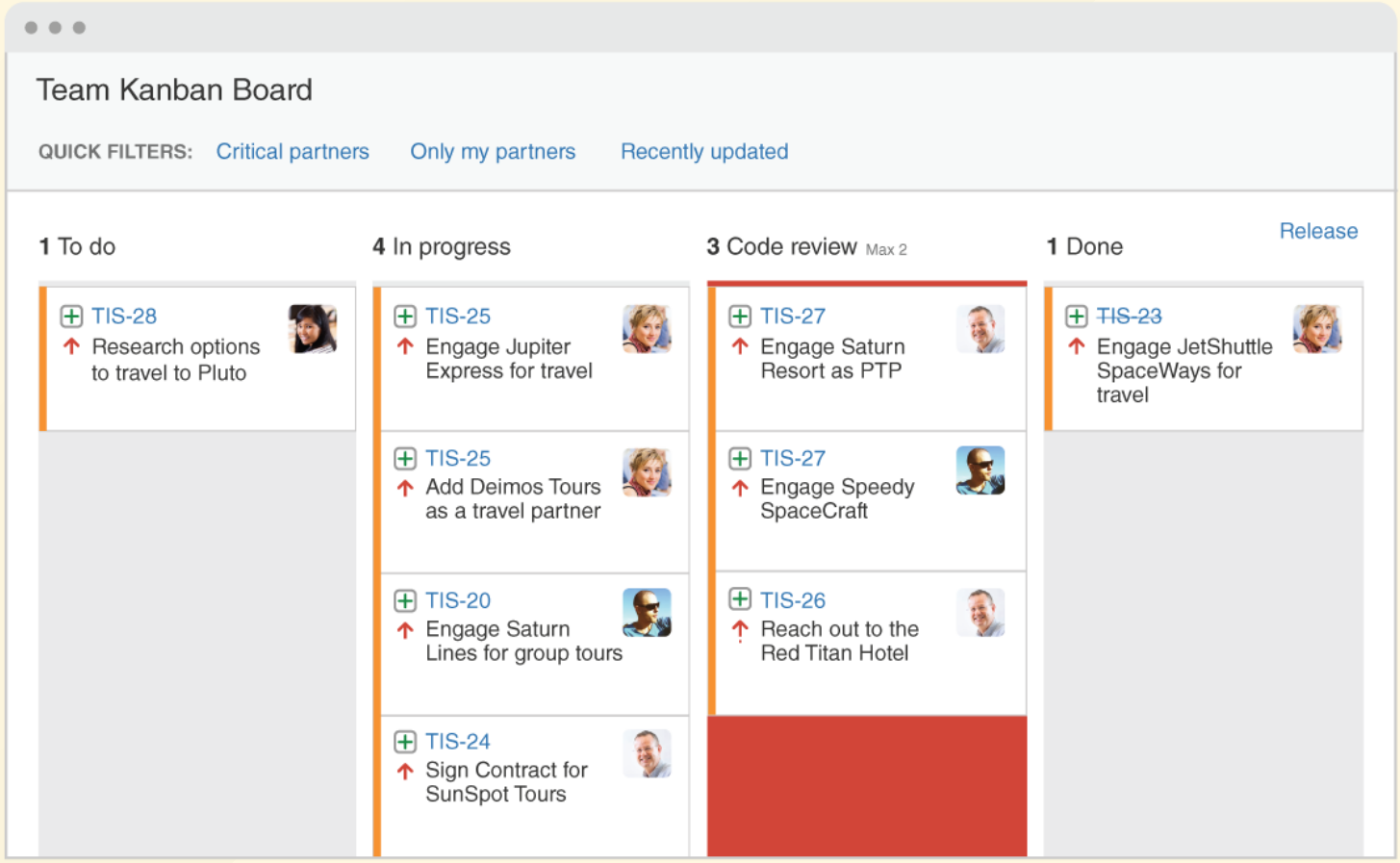
Kanban

Kanban Board Example



Kanban

JIRA Kanban Board



Kanban

General Practices

- Visualization - 시각화
- Limiting `WIP`(Work In Progress) - 진행 업무 제한
- `Flow` management - 흐름 관리
- Making policies explicit - 정책 만들기
- Using feedback loops - 피드백 루프
- Collaborative or Experimental `evolution` - 협업

Kanban

Limiting WIP(Work In Progress)

Kanban - `Limiting WIP`

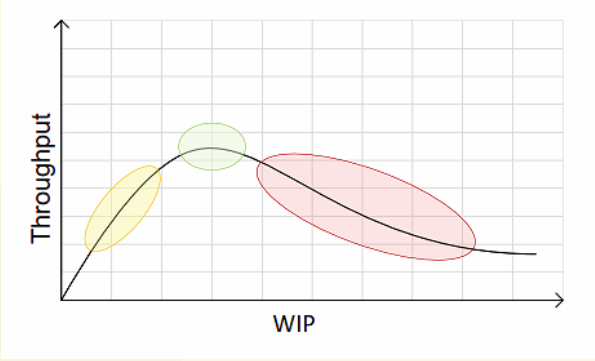
Why?

- 팀은 작게 나눈 일을 집중하면서 `throughput`을 올릴 수 있다.
- blockers, bottlenecks 볼 수 있음
- 과도한 업무로 개발자가 지치지 않도록 함. 생산성 유지

Kanban - `Limiting WIP`

작업 흐름 각 단계마다 동시에 진행 가능한 작업 수 제한

- Pull System 구현 기본 사항
- 자원의 활용도가 높아짐
  - Low WIP --> Developer idle++ --> bad `productivity`
  - High WIP --> Task idle++ --> bad `lead time`



Kanban & Lean

Kanban & Lean

- 기민하게 그리고 적시에 상품을 출시하기 위한 스케줄링 시스템 (with Pull System)
- `Waste` Minimization
  - Multitasking, context switching에서 생기는 낭비 제거
  - Pull 방식으로, 불필요한 일을 하지 않음`(Over-production)`
- `Value` Maximization
  - 사용자가 원하는 기능, 제품을 만들어낸다

Appendix A

Lean Software Development

Lean Software Development

> Lean software development(LSD) is a translation of `lean manufacturing` principles and practices to the `software development domain`.

목표:

`낭비`를 줄이고, 고객에게 더 높은 `가치`를 만든다.

Lean SD, 7 Principles

- Eliminate `Waste`: 낭비 제거
- Amplify `learning`: 학습!
- Decide as `late` as possible: 느린 결정
- Deliver as `fast` as possible: 빠른 전달
- Empower the `team`: 팀 존중
- Build `integrity in`: 내제화
- See the `whole`: 전체보기

Lean SD, Principles - `Eliminate Waste`

- `Value stream mapping`을 통해 `waste`를 찾는다
- waste가 있는 지점을 찾고 제거한다
- 핵심 공정만 남을 때까지 `iteratively`하게

Lean SD, Principles - `Amplify learning`

Software development is a continuous learning process based on `iterations` when writing code.

- `잦은 반복(iteration)`과 고객의 피드백을 통해 경험과 지식을 창출하라
- 잦은 반복
  - 요구사항 -> 설계 -> 개발 -> 테스트
- 방식
  - Pair Programming, Code Review ...

Lean SD, Principles -

`Decide as late as possible`

- 예측이 아닌 `사실을 기반`으로 결정을 할 수 있을 때까지 확정을 늦춰라
- 왜?
  - 결정을 미루라는 말이 아님, 정보를 최대한 많이 가지고 있을 때의 결정이 유리하기 때문
  - 개발 환경 = 예측이 불가능한 상황
  - 결정은 하되 변화를 수용할 수 있어야 함

Lean SD, Principles -

`Deliver as fast as possible`

- 고객의 확실한 요구사항을 파악하기 위해 제품을 가능한 빨리 인도하라
- 품질의 내재화가 필수 조건이다
- 필요조건
  - 낭비의 제거, 품질의 내재화
  - 일의 양 제한(WIP 제한)

Lean SD, Principles - `Empower the team`

- `Respect People`
- 사람을 대체 가능한 인력으로 다루지 말라
- 사람을 신뢰하고, 전문 기술을 가질 수 있도록 인재를 육성하라

Lean SD, Principles - `Build integrity in`

- 결함을 예방하는 `테스트`를 통해 코드의 품질을 내재화하라
- 실행방법: TDD, Continuous Integration, Pair Programming...
- 빅뱅 통합보다 `지속적 통합`을 하라
- 자동화된 단위 테스트와 인수 테스트 작성을 하라

Lean SD, Principles - `See the whole`

- 부분 최적화가 전체를 최적화하지 않을 수 있다
- `문제의 근본 원인`을 찾아 전체를 최적화하라

> “ Think `big`, act `small`, fail `fast`; learn `rapidly` ”

고맙습니다