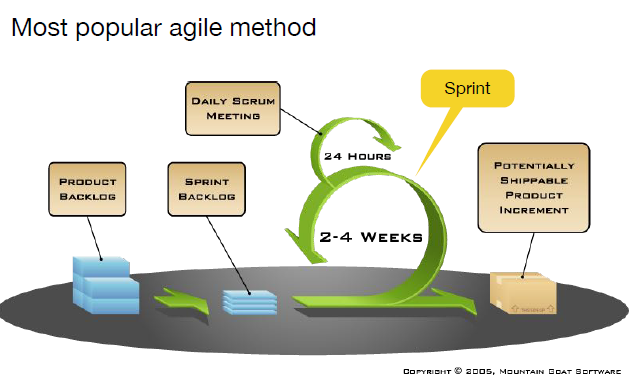
**[ 소프트웨어공학 ]**

**1. Agile 소프트웨어 개발 방법론**

- 애자일은 ‘민첩합’, ‘기민한’이라는 의미로, 고객의 요구사항 변화에 유연하게 대응할 수 있도록 일정한 주기를 반복하면서 개발과정을 진행한다.

- 애자일 모형은 **스프린트(Sprint)** 또는 **이터레이션(Iteration)**이라고 불리는 짧은 개발 주기를 반복하며, 반복되는 주기마다 만들어지는 **결과물(product increment)**에 대한 고객의 평가와 요구를 적극 수용한다.

[**https://book.naver.com/bookdb/book\_detail.nhn?bid=17769459**](https://book.naver.com/bookdb/book_detail.nhn?bid=17769459)



- 실천방법으로는 **Scrum**과 **XP(eXtreme Programming)**이 있다.

**2. 스크럼(Scrum)**

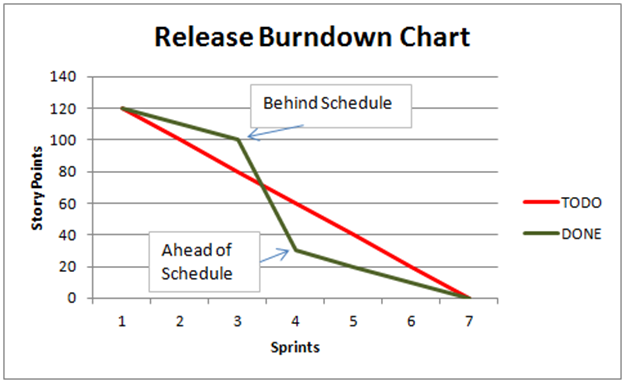
- **스크럼(Scrum)**의 어원은 럭비에서 반칙으로 경기가 중단된 경우 양팀의 선수들이 럭비공을 가운데 두고 상대팀을 밀치기 위해 서로 대치해 있는 대형을 말한다. 스크럼은 초기 위험 관리(early risk management)에 초점을 맞춘 관리 및 제어 프로세스이다. 매일 스크럼 미팅이 진행되며, 스프린트가 끝날 때마다 각 스프린트 검토하고 우리가 달성한 것과 향후 2 ~ 4 주 내에 무엇을 할지를 계획하는 과정으로 진행된다.

: **제품 백로그**(product backlog) – 제품 개발에 필요한 모든 요구사항(user story)을 우선순위에 따라 나열한 목록이다.

: 스프린트 계획 회의(Planning) – 스프린트에 처리할 요구사항을 개발자들이 나눠서 작업할 수 있도록 태스크(task) 단위로 나누고 **스프린트 백로그(sprint backlog)**를 작성한다.

: **스프린트** – 실제 개발 작업을 진행하는 과정, 2-4주정도의 기간, 태스크는 할 일(to-do), 진행중(in-progress), 완료(done)의 상태를 갖는다.

: **일일 스크럼 미팅(Daily Scrum Meeting)** – 모든 팀원이 매일 약속된 시간에 약 15분 정도의 짧은 시간동안 진행 상황을 점검한다. 남은 작업 시간은 **burn-down chart**에 표시한다. **스크럼 마스터(master)**는 발견된 장애 요소를 해결할 수 있도록 지원하는 역할을 한다. Product Owner(요구사항 관리자) -> 요구사항에 구현 진척상황 관리, 추가적이거나 변경되는 요구사항 관리. Risk(문제가 발생할 수 있는 상황), Issue(문제 발생해서 해결해야되는 상황)



: 스프린트 검토 회의(Review) – 부분 또는 전체 완성 제품이 요구사항에 잘 부합되는지 사용자가 포함된 참석자 앞에서 테스팅을 수행한다. 스프린트의 한 주당 한 시간 정도로 진행한다. 제품 책임자(Product Owner)는 개선할 사항에 대한 피드백을 정리한 후 다음 스프린트에 반영할 수 있도록 제품 백로그를 업데이트한다.

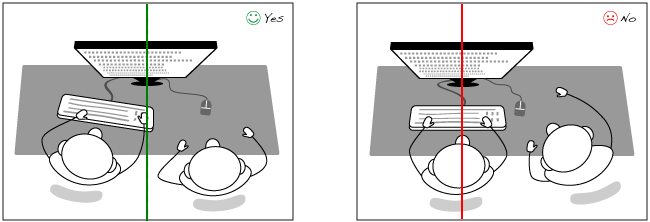
: 스프린트 회고(Retrospective) – 스프린트 주기를 되돌아보며 정해놓은 규칙을 잘 준수했는지, 개선할 점은 없는지 등을 확인하고 기록한다. 스프린트가 끝난 시점에 수행하거나 일정 주기로 수행한다.

**3. XP(eXtreme Programming)**

: **XP(eXtreme Programming)**에서는 반복주기를 이터레이션(iteration)이라고 하면 1~3주정도의 기간으로 진행된다.

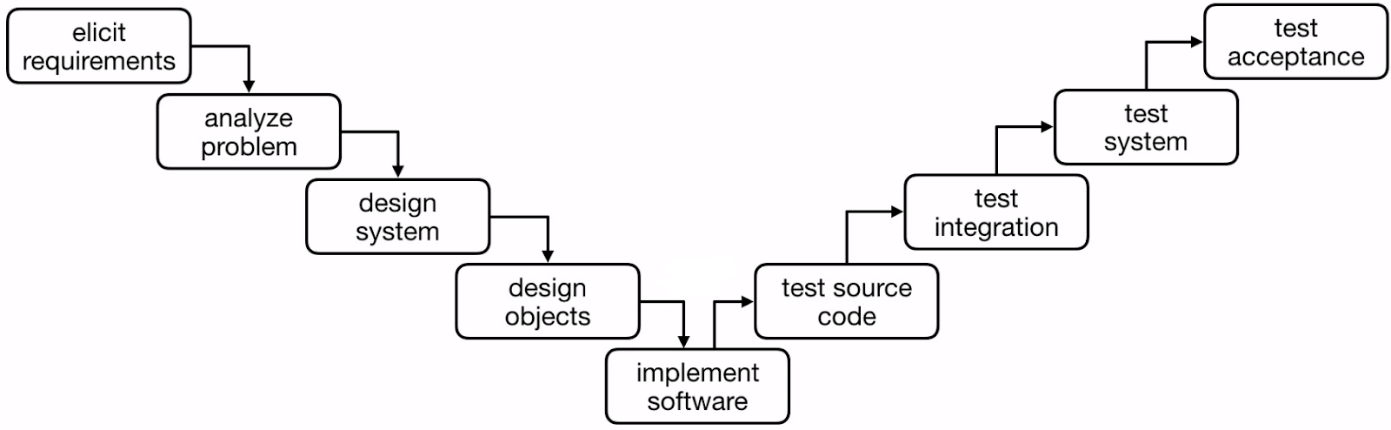
**: Pair-Programming(짝프로그래밍)** – 짝으로 나눠서 교대로 프로그래밍 수행. Driver(코드짜는사람)와 Navigator(옆에서보는사람). 5분간격으로 역할을 교체하면서 진행. 서로 코드 리뷰하는 것, Peer-Review

<https://gmlwjd9405.github.io/2018/07/02/agile-pair-programming.html>



**: TDD(Test Driven Development)** – 실제 구현 코드를 작성하기 전에 **요구사항에 따라 테스트케이스를 먼저 작성**하고 **개발하는 방식**을 말한다. 따라서 요구사항이 무엇인지 좀더 정확히 파악한 후 개발을 진행할 수 있으며, 개발이 진행될 때 **테스트가 지속으로 진행**될 수 있도록 자동화된 테스팅 도구를(JUnit, Cucumber, Jest) 사용한다.

\*\* 테스트 단계에 따른 소프트웨어 생명주기 모델(V-Model)



\*\* 단위 테스트(소스코드) -> 통합 테스트(서비스간 연동 테스트) -> 시스템 테스트 -> 인수 테스트

**: CI/CD(Continuous Integration/Continuous Delivery)** – **나눠서 개발된 모듈들을 지속적으로 통합**하고 주요 **의사결정권자들이 확인할 수 있는 배포환경에 전달**하는 것을 말한다.

