1) 폭포수 모델

1. 소프트웨어 요구사항 기술
2. 소프트웨어 설계
3. 소프트웨어 구현 (또는 코딩)
4. 시험과 디버깅
5. 설치
6. 소프트웨어 유지보수

"[시스템의] 세부 사항 중 많은 부분은 우리가 [시스템의] 구현을 진행할 때라야 비로소 우리 눈에 보이기 시작한다. 그리고 그렇게 알게 된 것 중 일부는 우리의 기존 설계를 무효로 만들고, 다시 전 단계로 돌아갈 수밖에 없게 만든다."

- 납기일 전 철야
- 철야에도 불구하고 납기일 지연
- 지연에 따른 비난과 스트레스로 개발자 에너지 소진
- 결국 납품된 솔루션은 고객의 요구를 충족하지 못함

2) 애자일 소프트웨어 개발

애자일 방법론은 소프트웨어 개발 방법에 있어서 아무런 계획이 없는 개발 방법과
계획이 지나치게 많은 개발 방법들 사이에서 타협점을 찾고자 하는 방법론이다.
계획이 없는 방법론의 경우, 앞으로의 일을 예측하기 힘들고 효율적이지 못하다는 점에서 취약점을
가지고 있으며, 계획에 너무 의존하는 경우는 그 형식적인 절차를 따르는데 필요한 시간과
비용을 무시할 수 없으며, 전체적인 개발의 흐름 자체를 느리게 하는 단점을 가지고 있다.

그렇기 때문에 애자일 방법론에서 택한, 그리고 다른 고전적인 방법론,
예를 들면 폭포수 모델 또는 나선 모형과 구별되는 가장 큰 차이점은 less document-oriented,
즉 문서를 통한 개발 방법이 아니라, code-oriented, 실질적인 코딩을 통한 방법론이라는 점이다.

그러므로 애자일 개발 방법론은 계획을 통해서 주도해 나갔던 과거의 방법론과는 다르게
앞을 예측하며 개발을 하지 않고, 일정한 주기를 가지고 끊임없이 프로토 타입을 만들어내며
그때 그때 필요한 요구를 더하고 수정하여 하나의 커다란 소프트웨어를 개발해 나가는
adaptive style 이라고 할 수 있다.

애자일 개발 프로세스란 어느 특정 개발 방법론을 가리키는 말은 아니고
"애자일(Agile=기민한, 좋은것을 빠르고 낭비없게 만드는 것) 개발을 가능하게 해 주는
다양한 방법론 전체를 일컫는 말이다. 예전에는 애자일 개발 프로세스는 "경량(Lightweight)"
프로세스로 불렸다. 익스트림 프로그래밍 (XP:eXtreme Programming)이 애자일 개발 프로세스의
대표적인 방법이라 볼 수 있다

*** 스크럼**

1. 솔루션에 포함할 기능/개선점에 대한 우선 순위를 부여한다.
2. 개발 주기는 30일 정도로 조절하고 개발 주기마다 실제 동작할 수 있는 결과를 제공하라.
3. 개발 주기마다 적용할 기능이나 개선에 대한 목록을 제공하라.
4. 날마다 15분 정도 회의를 가져라.
5. 항상 팀 단위로 생각하라.
6. 원활한 의사소통을 위하여, 구분 없는 열린 공간을 유지하라.

*** 애자일의 실패는?**
http://morethanair.com/archives/817

고객 : A 기능을 이번 Phase에 넣고 싶은데요
나 : 이번 Scope에서 처리하기는 힘듭니다. 다음 Phase에서 진행하시죠.
고객 : 다음 Phase의 CR로 넣는다면 추가 비용이 있습니까?
나 : 네, CR로 들어가면a 비용이 발생합니다.
고객 : 그러면 이번 Phase에서 해주세요.
나 : ??

*** 애자일/스크럼 프로젝트는 왜 실패하는가?**
http://egloos.zum.com/parkpd/v/3724780

2) 데브옵스(DevOps)
*** Devops**
https://zetawiki.com/wiki/%EB%8D%B0%EB%B8%8C%EC%98%B5%EC%8A%A4_DevOps
*** 없어서 못 뽑는다는 데브옵스(DevOps) 개발자, 어떤 일을 할까?**
http://www.thisisgame.com/webzine/news/nboard/4/?n=71605
*** Full-stack Engineer**
https://zetawiki.com/wiki/%ED%92%80%EC%8A%A4%ED%83%9D_%EC%97%94%EC%A7%80%EB%8B%88%EC%96%B4
*** 풀스택 개발자, 그것은 환상**
https://brunch.co.kr/@supims/17

- 소프트웨어의 개발(Development)과 운영(Operations)의 합성어로서, 소프트웨어 개발자와 정보기술 전문가 간의 소통, 협업 및 통합을 강조하는 개발 환경이나 문화를 말한다. 데브옵스는 소프트웨어 개발조직과 운영조직간의 상호 의존적 대응이며 조직이 소프트웨어 제품과 서비스를 빠른 시간에 개발 및 배포하는 것을 목적으로 한다.

*** 목적**
데브옵스의 목적은 전반적인 배포 파이프라인에 걸쳐있다. 여기에는 개선된 배치(deployment) 주기를 포함하며 다음으로 이어질 수 있다: 제품 출시까지 걸리는 기간(time to market) 단축 새로운 판의 더 낮은 실패율 픽스 간 짧아진 리드 타임(상품 생산 시작부터 완성까지 걸리는 시간)
복구 시 더 빠른 평균 시간 (새로운 릴리스의 충돌 및 그 밖에 현재의 시스템를 비활성화하는 상황에서)

단순한 프로세스들은 데브옵스 접근을 사용하여 더 프로그래밍 가능하게되고 유동적으로 되고 있다. 데브옵스는 운영 프로세스의 예측 가능성, 효율성, 보안, 유지보수 가능성을 극대화하는 것이 목적이다. 더 가끔씩 자동화가 이러한 목표를 지원한다.

1.코드 - 코드 개발 및 검토, 버전 관리 도구, 코드 병합
2.빌드 - 지속적 통합(CI) 도구, 빌드 상태
3.테스트 - 테스트 및 결과가 성능을 결정
4.패키지 - 애플리케이션 디플로이 이전 단계
5.릴리스 - 변경사항 관리, 릴리스 승인, 릴리스 자동화
6.구성 - 인프라스트럭처 구성 및 관리, IaC(Infrastructure as Code) 도구
7.모니터링 - 애플리케이션 성능 모니터링, 최종 사용자 경험.