Software Engineering

Lecture 02: 소프트웨어 공학과 개발 프로세스 (Part 2)

Professor: Jung Uk Kim

ju.kim@khu.ac.kr

Computer Science and Engineering, Kyung Hee University

Visual AI Lab.

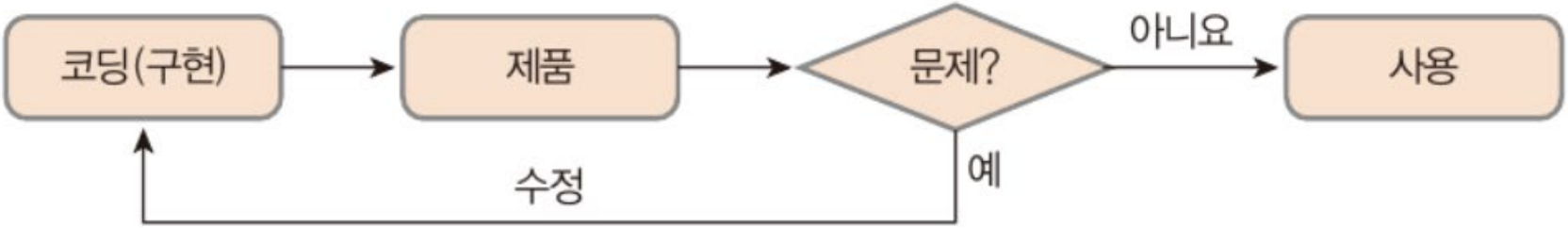
KYUNG HEE UNIVERSITY

소프트웨어 개발 생명주기 모델

- **주먹 구구식 모델**
- **선형 순차적 모델**
  - 폭포수 모델
  - V 모델
- **진화적 프로세스 모델**
  - 프로토타입 모델
  - 나선형 모델
- **단계적 개발 모델**
  - 점증적 개발 모델
- **애자일 프로세스 모델**

주먹 구구식 모델

- **주먹 구구식 모델**
  - 공식적인 가이드라인이나 프로세스가 없는 개발 방식
  - 즉흥적 소프트웨어 개발 또는 코딩과 수정 모델
    - **Build and fix** 또는 **Code and fix**
  - 코드를 작성해 제품을 만든 후에 요구 분석, 설계, 유지보수에 대해 생각
    - 우선 **제품을 완성한 뒤 코드에 문제가 있으면 수정, 문제가 없으면 사용**



[Remind]

- 반려동물 집 짓기
  - 혼자서 할 수 있고 만드는 과정도 단순

주먹 구구식 모델

- **주먹 구구식 모델 (단점)**
  - 정해진 개발순서나 **각 단계별로 문서화된 산출물이 없음**
    - 관리 및 유지보수가 어려움
  - **문제 범위를 정하지 않음**
    - 프로젝트 전체 범위를 알 수 없고 좋은 아키텍쳐를 만들 수 없음
  - **대규모 프로젝트에서는 불가능**한 방식
  - 코딩을 먼저 하므로 계속 수정할 가능성이 높은데, 여러 번 수정하다 보면 가독성이 높은 구조가 나빠져 수정이 어려워질 수 있음

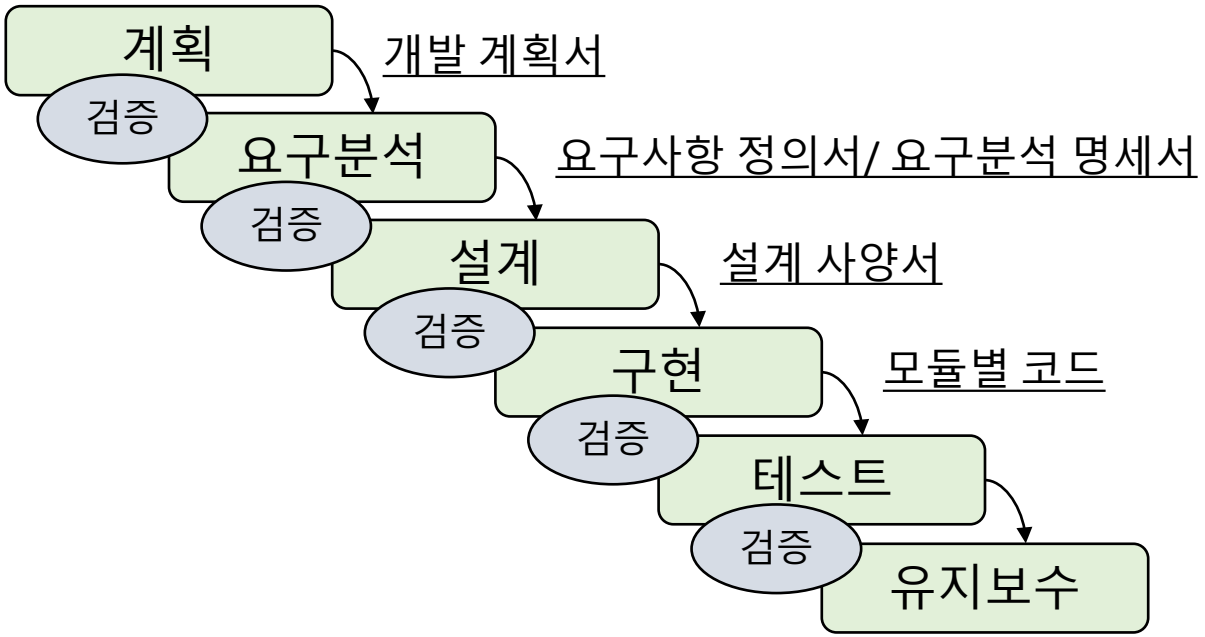
```
int main()
{
	int i = 0;
	int tmp;
	int tmp1 = 1;
	int tmp2 = 1;

	printf("make Fibonacci Sequence ₩n₩n");
	printf("%d  %d  ", tmp1, tmp2);

	while (i != 10)
	{
		tmp = tmp2;
		tmp2 += tmp1;
		tmp1 = tmp;
		printf("%d  ", tmp2);
		i++;
	}
}
```

선형 순차적 모델

- **선형 순차적 모델 (Linear Sequential Model)**
  - 일반적으로 **폭포수 모델**로 알려져 있음 (**고전적 생명 주기**)
    - 폭포에서 물이 떨어지듯이 **다음 단계로 넘어가는 모델**
    - 요구사항이 확정되어 변동이 없는 **소규모의 프로젝트에 적합**
  - 계획, 요구분석, 설계, 구현, 테스트, 유지보수 단계들이 **하향식**으로 진행
    - 단계의 종료마다 **확실하게 작업을 종료**(이 때 **문서화가 강조**)하고 그 **결과를 확인**한 뒤 **다음 단계로 내려 감**

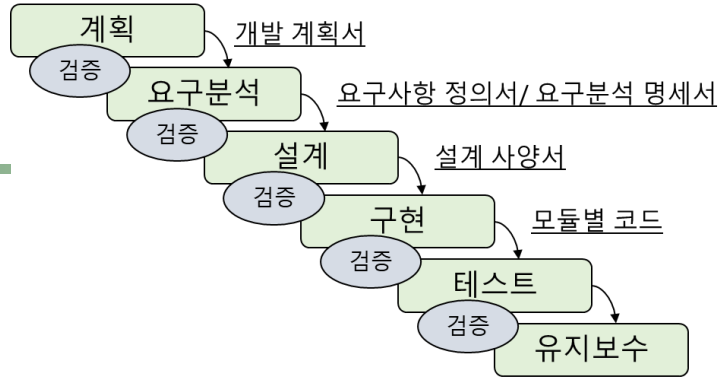


선형 순차적 모델

- **폭포수 모델 (장점)**
  - 절차가 단순하여 이해하기 쉬움
  - **단계별 진척상황에 대한 관리가 용이**함
  - 각 단계별 산출물을 체계적으로 **문서화**할 수 있음

- **주먹 구구식 모델 vs. 폭포수 모델**
  - **단계의 유무**
    - 주먹 구구식 모델: **단계 X** (Code and fix)
    - 폭포수 모델: **단계 O** (계획, 요구분석, 설계, 구현, 테스트, 유지보수)
  - **문서화된 산출물**
    - 주먹 구구식 모델: **문서화된 산출물 X**
    - 폭포수 모델: **각 단계별 문서화된 산출물 O**

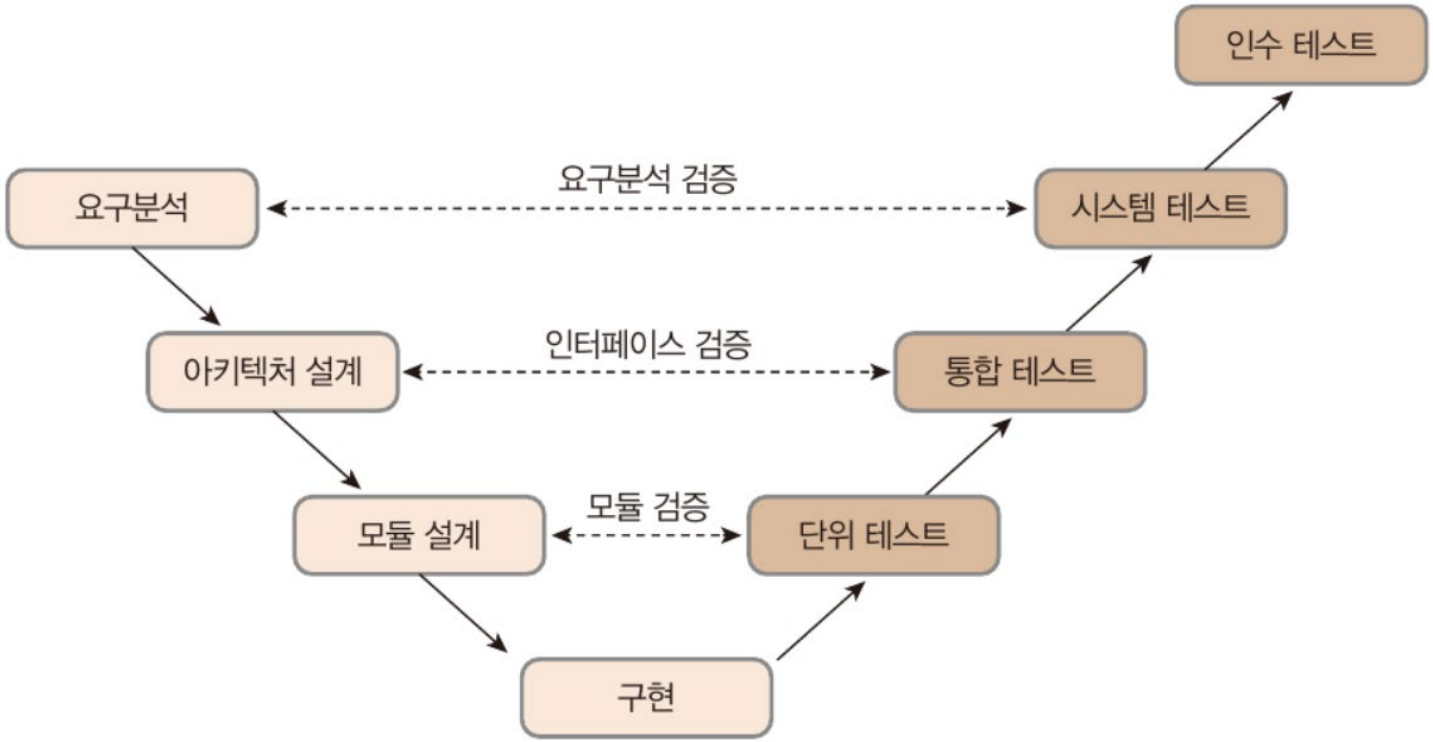
선형 순차적 모델



- **폭포수 모델 (단점)**
  - 각 단계는 **이전 단계가 완료되어야 수행** 됨
    - 예) 설계단계: 요구사항 정의서가 나와야 시작
    - 각 단계에서 사용자의 요구사항을 반영할 수 없음 (요구분석 단계는 제외)
  - 각 단계 결과물이 **완벽한 수준**이 되어야 다음 단계에 오류를 넘겨주지 않음
  - 사용자가 중간에 **가시적인 결과를 볼 수 없음**
    - 사용자의 수정 요구사항을 반영하기 위해 시간과 비용이 많이 듬
  - 최근의 소프트웨어 개발은 **요구사항이 끊임없이 변화**함
    - 최근에는 폭포수 모델은 거의 사용되지 않음

선형 순차적 모델 - V 모델

- **V 모델**
  - **테스트 단계를 추가**한 폭포수 모델의 변형
  - 각 **개발 단계를 검증**하는데 초점 (vs. 폭포수 모델은 **산출물 중심**)
    - 오류를 줄일 수 있음



선형 순차적 모델 - V 모델

- **V 모델 (장점)**
  - 각 **개발 단계를 검증**에 초점을 맞춰 **오류를 줄일 수 있음**
  - **신뢰성이 높이 요구되는 분야**에 적용 가능

- **V 모델 (단점)**
  - **폭포수 모델**과 유사하게 **각 단계가 반복되지 않아** 변경을 다루기 쉽지 않음

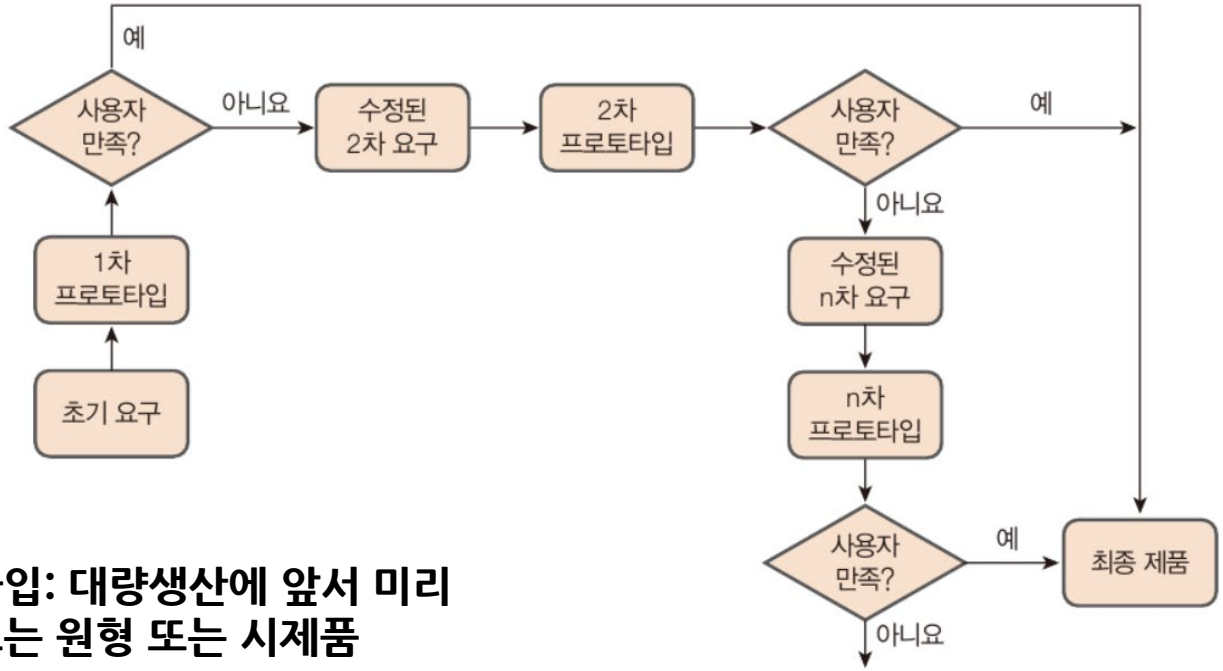
진화적 프로세스 모델

- **진화적 프로세스 모델**
  - 폭포수 모델의 **단점 해결**을 위함 (등장배경)
    - 폭포수 모델의 단점 - 단계를 거슬러 올라가 작업하는 것이 쉽지 않음
    - 개발 과정 중 **새로운 요구가 발생해도 이에 민첩하게 대응**할 수 있는 방법
      - 현실은 새로운 요구가 수시로 발생

→ **총 쏘는 기능**(기존 요구사항)에
추가로 **총 던지는 기능**(새로운 요구사항)
을 추가 해주세요.

진화적 프로세스 모델

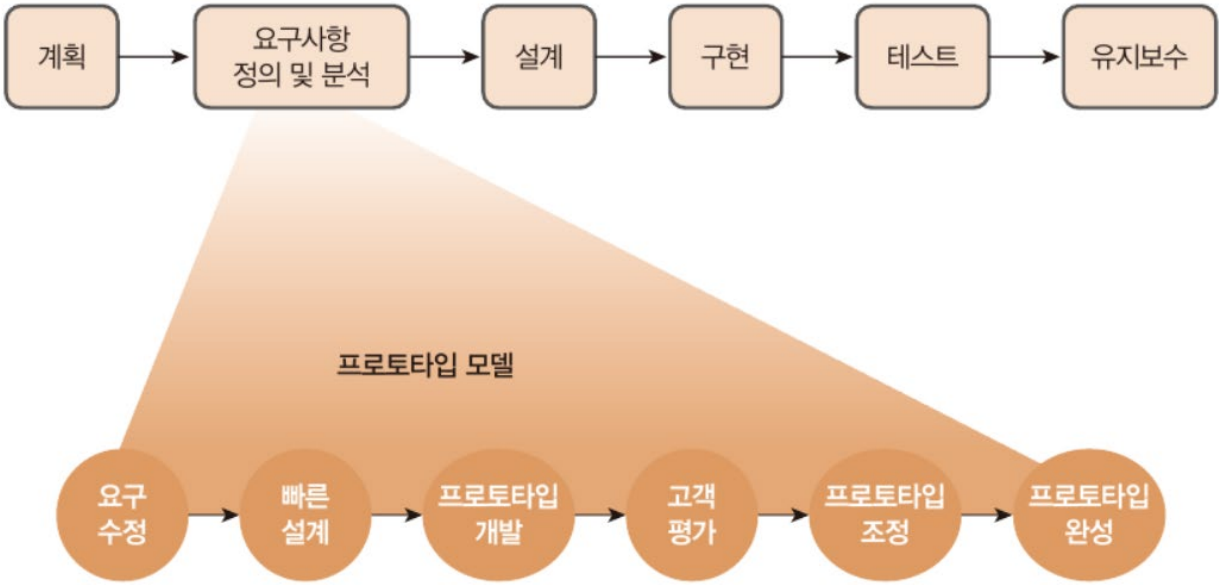
- **진화적 프로세스 모델**
  - 주로 **요구분석 단계**를 고려한 모델
  - 특징
    - (1) 초기의 사용자 요구에 따라 **초기 버전의 프로토타입 개발**
    - (2) 이를 바탕으로 나타나는 **가상의 결과 화면을 관찰**
    - (3) 변경된 요구 사항을 반영하거나 추가해 **지속적인 프로토타입 개선**



프로토타입: 대량생산에 앞서 미리 제작해보는 원형 또는 시제품

진화적 프로세스 모델 - 프로토타입 모델

- **프로토타입 모델**
  - 사용자의 **요구사항을 충분히 분석할 목적**으로 개발된 모델
  - 시스템의 일부분을 시험적으로 구현하고, 사용자의 피드백을 받아 다시 요구사항에 반영하는 과정을 반복하는 개발 모델 (**사용자 중심 모델**)



모델 하우스

<u>프로토타입 모델은 설계 이후의 과정에서는 반복이 이루어지지 않음</u>

진화적 프로세스 모델 - 프로토타입 모델

- **프로토타입 모델 vs. 폭포수 모델**
  - **요구사항 정의 분석**
    - **폭포수 모델**: **요구사항을 완전하게 명세한 후 다음단계**로 넘어 감
    - **프로토타입 모델**: 다양한 **요구사항을 반복적으로 수용**하여 완성도를 높여서 최종 프로토타입을 개발
  - **프로토타입 설계 및 개발**
    - **폭포수 모델**: 완성된 요구사항을 바탕으로 설계 및 개발
    - **프로토타입 모델**: 완제품을 보여주는 것이 아니라, 출력결과가 사용자가 원하는 것인지 보여주기 위해 설계 및 개발
  - **사용자에 의한 평가**
    - **폭포수 모델**: 테스트 과정에서 평가가 이루어지지 않음
    - **프로토타입 모델**: 반복된 프로토 타입을 기반으로 평가가 이루어짐

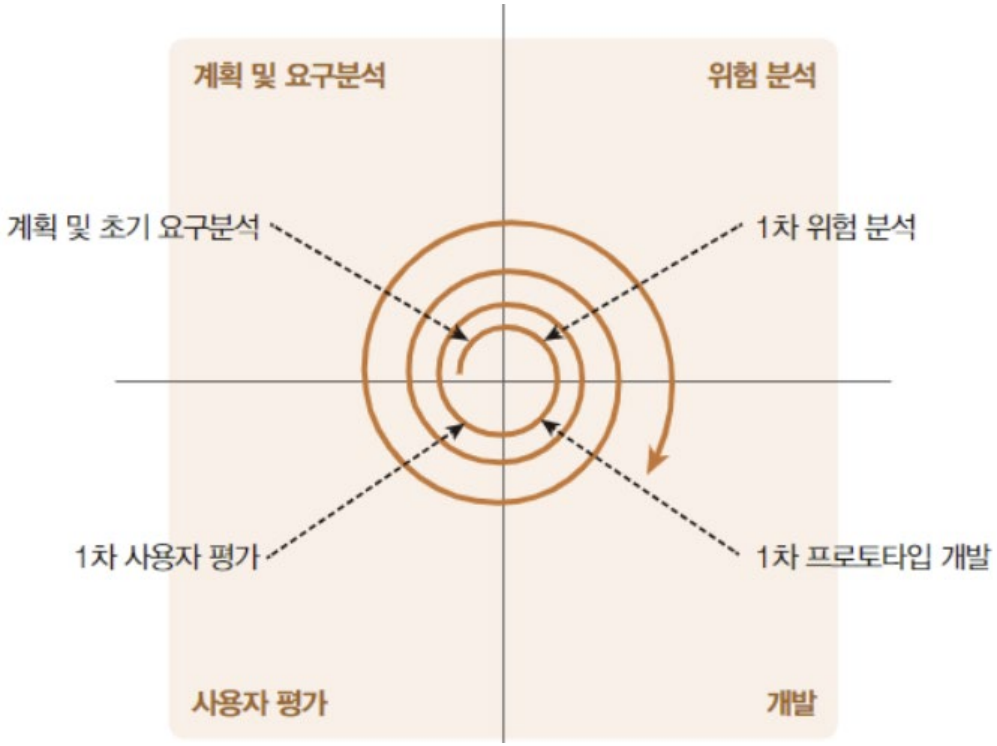
진화적 프로세스 모델 - 프로토타입 모델

- **프로토타입 모델 (장점)**
  - 사용자의 요구가 지속적으로 발생하는 경우에 적합
  - **사용자의 요구가 충분히 반영**되어 최종 제품이 나오므로 **유지보수에 필요한 노력과 시간을 많이 줄일 수 있음**

- **프로토타입 모델 (단점)**
  - 반복적인 개발 단계로 인해 필요한 투입 인력과 비용산정이 어려움
  - 개발자 입장
    - 프로토타이핑 과정을 관리/통제가 어려움
    - 개발범위가 명확하지가 않아 언제 종료될 지 모름

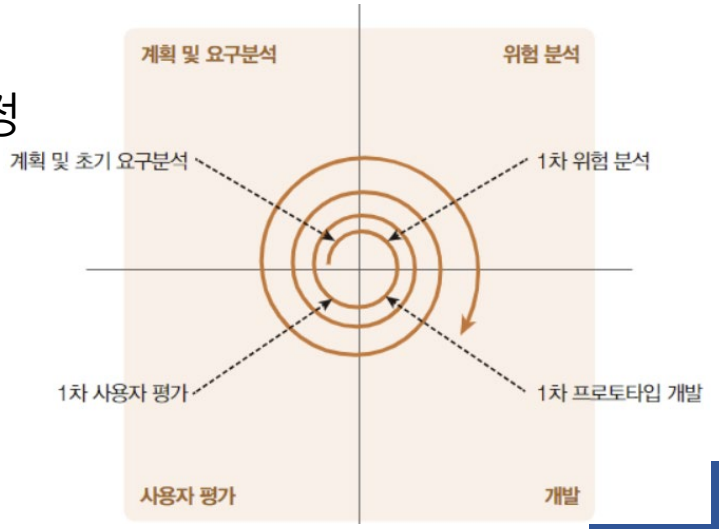
진화적 프로세스 모델 - 나선형 모델

- **나선형 모델**
  - 시스템 개발시 **위험을 최소화 하기 위해** 점진적으로 완벽한 시스템으로 개발해 나가는 모델
  - 진행순서: 요구분석 → 위험분석 → 개발 → 평가
  - 대규모 프로젝트, 국책사업 및 **위험 부담이 큰 시스템 개발에 적합**



진화적 프로세스 모델 - 나선형 모델

- **나선형 모델**
  - **요구분석**
    - **고객의 요구사항 분석 및 타당성 검토, 프로젝트 수행 여부 결정**
    - **각 단계에 대한 목표** 수립
      - 목표: 본 cycle 안에서 완수되고 **새로운 cycle이 진행되면 변경되는 목표**
  - **위험분석**
    - 프로젝트 진행 시 **고객 요구사항을 기반으로 예측되는 위험 사항**에 대해 분석
  - **개발**
    - 위험 분석 완료 후 개발 모델 선택
  - **평가**
    - 고객이 평가 후 추가 반복에 대한 여부를 결정
    - **사용자 평가 단계는 핵심**



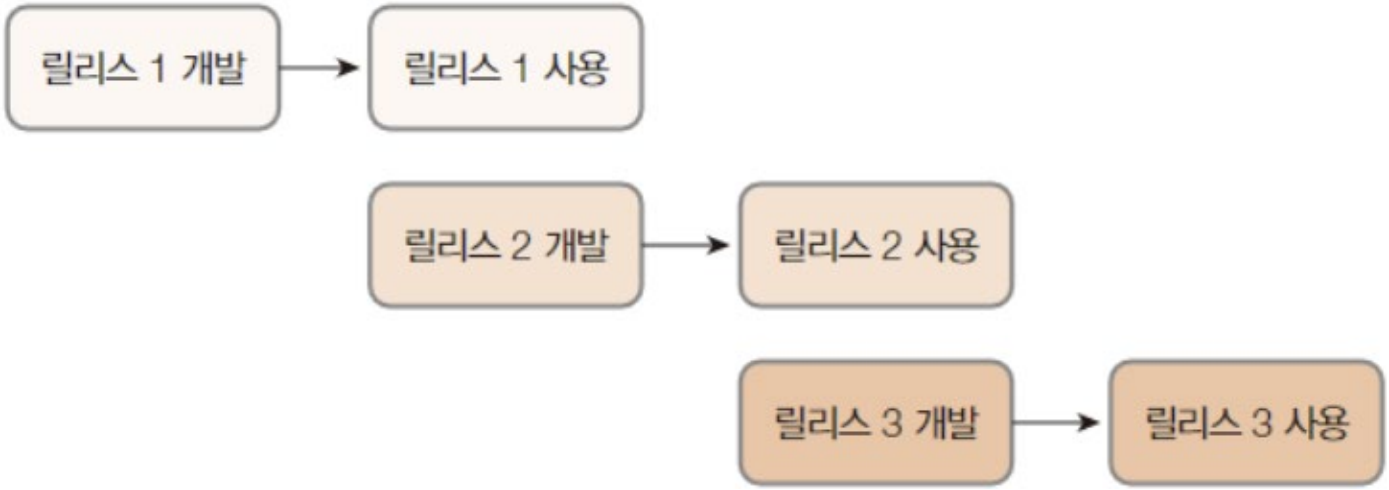
진화적 프로세스 모델 - 나선형 모델

- **나선형 모델 (장점)**
  - **위험 분석 단계가 존재**하기 때문에 개발 후반에 **갑작스럽게 발생하는 위험으로 인해 프로젝트가 중단되는 사태**가 비교적 **적음**
  - 사용자 평가가 반영된 개발 방식이므로 사용자 요구를 충분히 반영함

- **나선형 모델 (단점)**
  - 요구분석, 위험분석, 개발, 사용자 평가가 **반복적으로 진행**되기 때문에 프로젝트 기간이 길어질 수 있음
  - 반복횟수가 많아질 수 있어 **프로젝트 관리가 어려움**
  - 위험 관리가 중요한 만큼 **위험 관리 전문가가 필요**함

단계적 개발 모델

- **단계적 개발 모델**
  - 개발자가 먼저 릴리즈1을 개발해 사용자에게 제공해 이를 사용
  - 사용자가 릴리즈 1을 사용하는 동안 개발자는 릴리즈 2를 개발
  - **개발과 사용을 병행하는 과정을 반복**



단계적 개발 모델 - 점증적 개발 모델

- **점증적 개발 모델 (Incremental)**
  - 중요하다고 생각되는 부분부터 개발하고 **개발 범위를 점차 늘려가는 방식**
  - **요구분석명세서**에 명시된 시스템 전체를 **서브(Sub) 시스템으로 분할**
  - **서브 시스템을 단계적으로 하나씩 릴리스**해 완성하는 방법

유저 간 대결 기능

좋은 선수 추가 기능

나만의 선수 육성 기능

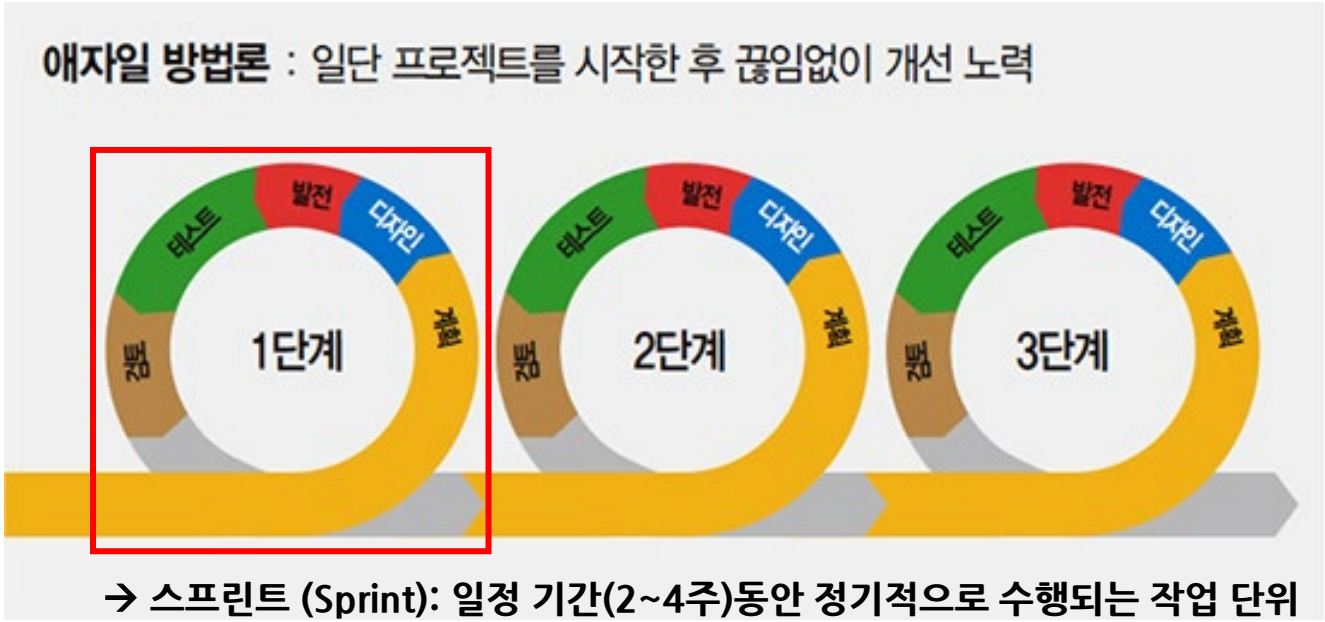
단계적 개발 모델 - 점증적 개발 모델

- **점증적 개발 모델 (장점)**
  - 한꺼번에 많은 비용을 들이지 않아도 됨
  - 완전히 새로운 시스템 전체를 한 번에 주었을 때 조직이 받는 충격을 완화할 수 있음
  - 이미 사용하고 있는 서브 시스템이 있어 어떤 유형으로 개발해야 하는지 잘 알 수 있음

- **점증적 개발 모델 (단점)**
  - 처음 설계할 때부터 이후에 개발할 **다른 서브 시스템과의 연관성을 고려**해야함
  - 이미 개발된 **서브 시스템과 통합 시 어려움**을 겪을 수 있음

애자일 프로세스 모델

- **애자일 프로세스 모델**
  - 애자일 (Agile): 민첩한, 날렵한
  - 진화적 프로세스 모델의 장점을 살리면서 요구사항의 변화를 주기적으로 수용하여 반복적으로 개발하는 모델
  - 고객의 **요구에 민첩하게 대응**하고, 그때그때 주어지는 문제를 풀어나가는 방법론

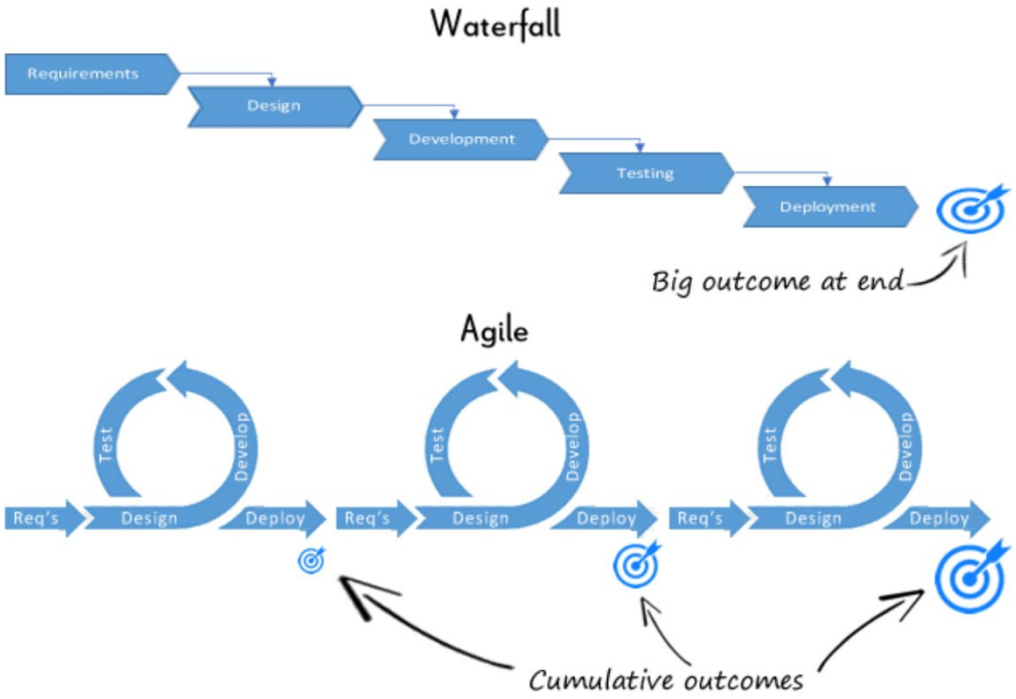


애자일 프로세스 모델

- **애자일 프로세스 모델의 목표**
  - 프로세스 중심이 아닌, **개개인과의 상호 소통**을 중시
  - 문서 중심이 아닌, **실행 가능한 소프트웨어**를 중시
  - 계약과 협상 중심이 아닌, **고객과의 협력**을 중시
  - 계획 중심이 아닌, **변화에 대한 민첩한 대응**을 중시

애자일 프로세스 모델

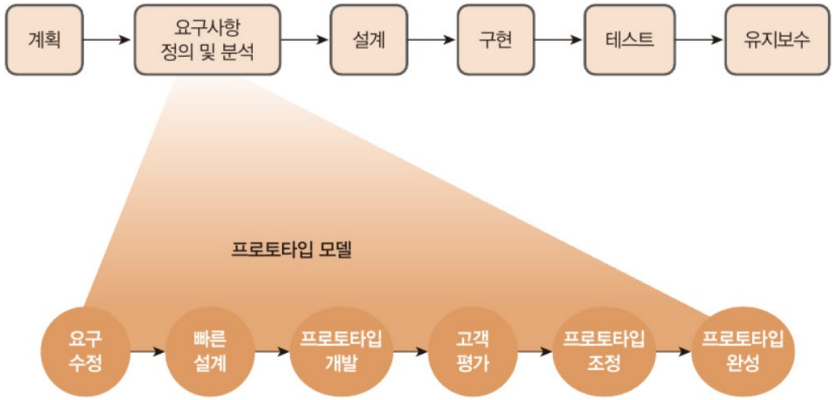
- **폭포수 모델 vs. 애자일 프로세스 모델**
  - **폭포수 모델**
    - 계획, 프로세스, 프로젝트 관리, 문서에 중점
  - **애자일 프로세스 모델**
    - 고객과의 협업, 빠른 시간 안에 고객이 작동해볼 수 있는 소프트웨어, 환경과 고객의 변화에 능동적으로 대처하는 것을 강조



애자일 프로세스 모델

- **프로토타입 모델 vs. 애자일 프로세스 모델**
  - **프로토타입 모델**
    - 요구사항을 매 단계마다 모두 반영하여 개발
    - 실제 설계가 이루어진 후에 **급격하게 생기는 요구사항은 반영하지 못함**
  - **애자일 프로세스 모델**
    - 스프린트 단위로 요구사항의 변화를 수용하면서 개발

    (구성원 간 검토를 거쳐 **우선순위화 하여 개발**, 우선순위가 낮은 것은 우선 반영하지 않을 수 있음)



프로토 타입 모델

애자일 프로세스

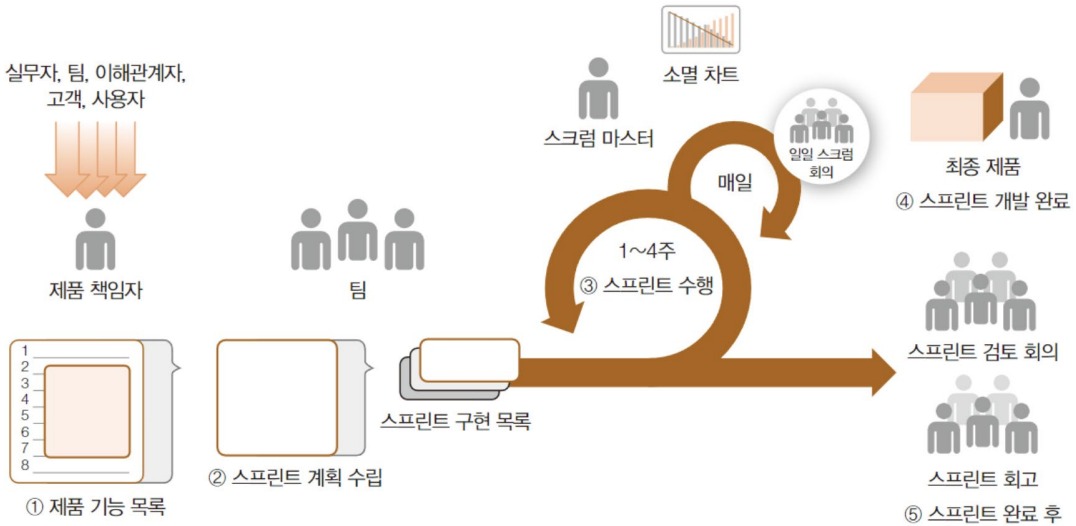
애자일 프로세스 모델 - 스크럼

- 스크럼(Scrum)
  - 애자일 프로세스 모델 중 가장 대표적
  - 스크럼: 럭비에서 반칙이 있을 때, 양편 선수가 밀집하려 서로 팔을 꼭 끼고 뭉치는 일 → **"팀(Team)" 이라는 의미**
  - 애자일 프로세스 모델 스크럼: **팀원들이 마치 하나처럼 움직이고 따라줘야 성공**할 수 있는 프로세스

스크럼

애자일 프로세스 모델 - 스크럼

- 스크럼 방식
  - 제품 기능 목록 작성
  - 스프린트 계획 회의
  - 스프린트 수행 (1~4주)
  - 스프린트 개발완료: 최종 제품
  - 스프린트 검토 회의 및 회고



단계	수행 목록	내용
1	제품 기능 목록 작성	• 요구사항 목록에 우선순위를 매겨 제품 기능 목록 작성
2	스프린트 계획 회의	• 스프린트 구현 목록 작성 • 스프린트 개발 기간 추정
3	스프린트 수행	• 스프린트 개발 • 일일 스크럼 회의 • 스프린트 현황판 변경 • 소멸 차트 표시
4	스프린트 개발 완료	• 실행 가능한 최종 제품 생산
5	스프린트 완료 후	• 스프린트 검토 회의 • 스프린트 회고 • 두 번째 스프린트 계획 회의

스크럼 방식

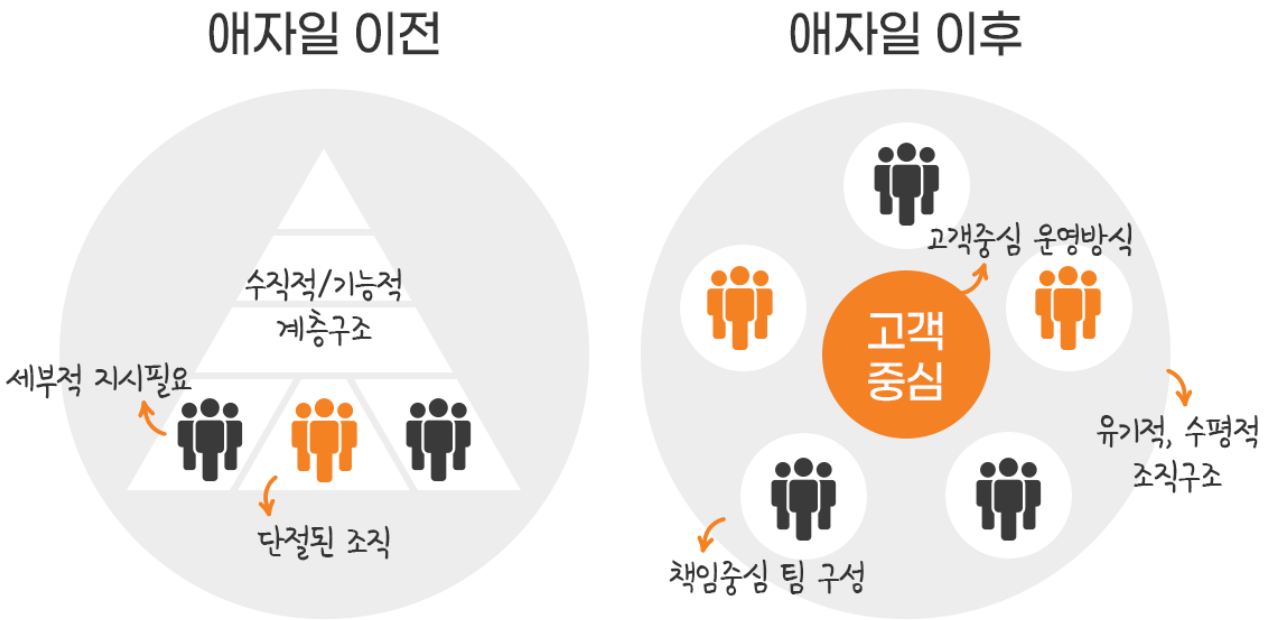
애자일 프로세스 모델

- **애자일 프로세스 모델 (장점)**
  - 프로젝트 계획에 걸리는 시간을 최소화
  - **점진적으로 테스트**할 수 있음 → **버그를 쉽고 빠르게 발견**할 수 있음
  - 계획 혹은 기능에 대한 **수정과 변경에 유연하게 대처가능**
  - **프로젝트의 진행 현황**을 볼 수 있어, **목표에 맞게 변화를 시도**할 수 있음

- **애자일 프로세스 모델 (단점)**
  - 빈번한 수정으로 개발 시 **개발 인력들의 피로도가 높음**
  - 불완전하고 예측 불가능한 요소(비용, 시간)를 지닌 채 진행할 수 있음
  - 애자일 프로세스 모델이 추구하는 **잦은 소통과 미팅은 대형 프로젝트에 적합하지 않은 평가**를 받음

애자일 프로세스 모델 (성공사례)

- **마이크로소프트 (Microsoft)**
  - 2000년 후반 - 심각한 경영난
  - 2014년, 사티아 나델라 취임후 조직문화 재정의
    - Top-down 방식 축소, 상호 적극적인 커뮤니케이션 피드백이 가능한 구조로 재구축
    - 결과 5년만에 클라우드 통합 서비스 점유율 세계 1위 도약



애자일 프로세스 모델 (실패사례)

- **실패한 사례의 특징**
  - 효율만 강조한 조직운영 → “효율”이라 쓰고 “비용절감”이라 읽는다
  - 효율과 속도에 치우쳐 안전을 도외시
  - 일부 경영진은 애자일을 조직 내에 빨리 도입해서 성과를 얻고 싶어함 → 결국 Top-down

→ 000님, 한달 동안 애자일 공부 해보시고, 사내 전파해주세요

→ 듣자 하니, 000님이 애자일 경험 좀 있다고 들었는데, 그럼 000님이 리딩 해주세요

→ 어제 모임에서 애자일이 좋다고 하던데, 우리도 내일부터 스크럼으로 일합시다

Questions?