1. 소프트웨어 개발 프로세스(Software Development Process)

- 소프트웨어 제품을 개발하기위해 필요한 과정 또는 구조
- 비슷한 말로 소프트웨어 생명 주기와 소프트웨어 프로세스가 있다

2. 개발 모형

1) 개발 모형의 종류

애자일 소프트웨어 개발, DSDM, 스크럼, 클린룸, 순차 점증적 개발, 반복형 개발, RAD, RUP, 나선 모형, 폭포수 모델, 익스트림 프로그래밍(XP), V 모델, TDD 등의 여러 모형이 존재한다 (Agile, Clean Room, Incremental, Prototyping, Spiral, V model, Waterfall)

2) 개발 모형 정리

[애자일 소프트웨어 개발(Agile Software Development)]

- 문서를 통한 개발방법이 아닌, 실질적인 코딩을 통한 방법론
- 일정한 주기를 가지고 끊임없이 프로토 타입을 만들어 나가며 그때 그때 필요한 요구를 더하고 수정하여 하나의 커다란 소프트웨어를 개발해 나가는 적응형 스타일
- 애자일 개발 방법론은 애자일(Agile : 기민한, 좋은 것을 빠르고 낭비없이 만드는 것) 개발을 가능하게 해 주는 다양한 방법론 전체를 일컫는 말
- 프로젝트의 생명주기 동안 반복적인 개발 촉진.

Framework	
Adaptive software development(ASD)	소프트웨어 개발을 혼란 자체로 규정하고, 혼란을 대전제로 그에 적응할 수 있는 소프트웨어 방법을 제시하기 위해 만들어진 방법론. 다른 방법론과 유사하나 합동 애플리케이션 개발(Joint Application Development : 사용자나 고객이 설계에 참여하는 개발 방법론)을 사용함
Agile modeling	모범 사례를 기반으로 소프트웨어 시스템을 모델링하고 문서화하는 방법론. 소프트웨어 개발 프로젝트에 적용될 수 있는 가치와 원칙의 모음으로 기존 모델링 방법보다 더 유연하여 빠르게 변화하는 환경에 적합함 장점 : 스크럼, 익스트림 프로그래밍과 같은 다른 애자일 방법론 보완 단점 : 개인적인 의사소통과 고객 협력에 크게 의존
Agile unified process(AUP)	Rational Unified Process(RUP)의 단순화된 버전. 간단하고 이해하기 쉬운 접근 방식. Disciplined agile delivery 방식으로 대체됨
Disciplined agile delivery(DAD)	점진적이고 반복적인 솔루션 제공에 대한 프로세스 결정 단순화 가능. 스크럼, 애자일 모델링, 린 소프트웨어 개발 등을 포함함. 사람 중심의 학습 지향 하이브리드 접근 방식 이용
Extreme Programming(XP)	고객과 함께 2주 정도의 반복 개발 진행, 테스트 우선 개발(TDD)을 특징으로 하는 명시적인 기술과 방법을 가짐
Feature-driven development(FDD)	기능 마다 2주정도의 반복 개발 실시. UML을 이용한 설계 기법과 밀접한 관련을 가짐

Dynamic systems development method(DSDM)	DSDM 애자일 프로젝트 프레임워크는 전체 프로젝트 라이프 사이클에 걸친 광범위한 활동을 다루며 다른 애자일 방법과 차별화되는 강력한 기반과 거버넌스를 포함 원칙 1) 비즈니스 요구에 집중 2) 정시에 배달 3) 협업 4) 품질을 타협하지 않기 5) 확고한 기반에서 점진적으로 구축 6) 반복적으로 개발 7) 지속적이고 명확하게 커뮤니케이션 8) 통제력 입증
Lean software development	도요타 생산 시스템으로부터 유래된 린 제조 원칙과 관행을 소프트웨어 개발 도메인으로 변환한 것. 린은 애자일 조직을 지원하는 견고한 개념적 프레임워크로, 가치 및 원칙은 물론 경험에서 파생된 모범 사례 제공 원칙 1) 낭비 제거 2) 학습 확대 3) 가능한 한 늦게 결정 4) 가능한 한 빨리 제공 5) 팀 역량 강화 6) 무결성 구축 7) 전체 최적화
Lean startup	스타트 업 회사가 초기 고객의 요구를 충족하기 위해 제품이나 서비스를 반복적으로 구축하는데 시간을 투자할 때 회사가 시장 위험을 줄이고 대량의 초기 프로젝트 자금 및 값 비싼 제품에 대한 필요성을 피할 수 있다는 가정이 핵심.

Kanban	인간 시스템 전반에서 작업을 관리하고 개선하는 린 방법. 수요와 가용 용량의 균형을 맞추고 시스템 수준 병목 현상 처리를 개선하여 작업을 관리하는 것을 목표로 함.
Rapid application development(RAD)	계획에 덜 중점을 두고 적응 프로세스에 더 중점을 둠. RAD는 사용자 인터페이스 요구 사항에 따라 구동되는 소프트웨어를 개발하는데 특히 적합함. 빠르고 쉽게 응용 프로그램을 만들 수 있는 시각적 도구 방법 1) 요구 사항 계획 단계 2) 사용자 설계 단계 3) 구축 단계 4) 컷 오버 단계 장점 : 더 좋은 품질을 얻을 수 있음. 위험 완화. 시간과 예산 내에서 프로젝트 완료 가능 단점 : 새로운 접근 방식의 위험. 정상 작동 시 최종 사용자에게 보이지 않는 비 기능적 요구 사항에 대한 강조 부족. 부족한 자원의 시간 필요. 통제력 감소. 형편없는 디자인과 확장성 부족
Scrum	30일마다 동작 가능한 제품을 제공하는 스프린트(Sprint) 중심. 매일 정해진 시간에 정해진 장소에서 짧은 시간의 개발을 하는 팀을 위한 프로젝트 관리 중심의 방법론
Scrumban	Scrum + Kanban. 스크럼에서 칸반으로 전환하는 방법으로 설계됨.

Scaled Agile Framework – SAFe	대규모 스크럼(LeSS), 훈련된 민첩한 전달(DAD), 넥서스와 함께 단일 팀을 넘어 확장 시 발생하는 문제를 해결하고자 하는 프레임 워크 중 하나. 애자일 소프트웨어 개발, 린 제품 개발, 시스템 사고라는 세 가지 주요 지식을 이용하여 개발되었으며, 수많은 팀에 걸친 조정, 협업 및 전달을 촉진함. 너무 계층적이고 융통성이 없다는 비판을 받고 있음 기본 원칙 1) 경제적 관점 취하기 2) 시스템 사고 적용 3) 가변성 가정 4) 빠른 통합 학습 주기로 점진적으로 구축 5) 작업 시스템의 객관적인 평가에 대한 기본 이정표 6) 진행중인 작업을 시각화 및 제한하고, 배치 크기를 줄이고, 대기열 길이 관리 7) 케이던스(타이밍) 적용, 도메인 간 계획과 동기화 8) 지식 근로자의 본질적인 동기 부여 9) 의사 결정 분산화 10) 가치를 중심으로 구성

[클린룸 소프트웨어 개발(Cleanroom Software Development)]

- 인증 가능한 수준의 신뢰성을 가진 소프트웨어를 제작하기 위한 방법

- 반도체 제조 과정에서 결함이 발생하는 것을 방지하는 클린룸에서 이름 유래

- 중심 원칙

1) 공식적인 방법에 기반한 소프트웨어 개발

2) 통계적 품질 관리 하에서 점진적 구현

3) 통계적으로 건전한 테스트

[점진적 모델(Incremental build model)]

- 소프트웨어 개발 제품이 되어 설계, 구현 및 테스트 제품이 완료될 때까지(변경될 때 마다 추가) 점진적 개발과 유지관리가 모두 포함됨
- 제품은 모든 요구 사항을 충족할 때 완제품으로 정의됨
- 폭포수 모델의 요소와 프로토타이핑의 반복적인 철학을 결합함

- 특성

  1) 시스템을 많은 미니 개발 프로젝트로 나눔

  2) 최종 시스템을 생산하기 위해 부분 시스템이 구축

  3) 우선 최우선 요구사항 해결함

  4) 증분 부분이 개발되면 부분의 요구사항이 동결됨

- 장점

  1) 각 반복 후에 회귀 테스트 수행. 이 테스트 중 단일 반복 내에서 거의 변경되지 않기 때문에 소프트웨어의 결함 요소를 빠르게 식별할 수 있음

  2) 일반적으로 다른 소프트웨어 개발 방법보다 테스트 디버깅이 더 쉬움

  3) 고객은 기능에 응답하고 제품에서 필요하거나 유용한 변경 사항을 검토할 수 있음

  4) 초기 제품 배송이 더 빠르고 비용이 적게 듦

- 단점

  1) 결과 비용이 예산을 초과할 수 있음

  2) 추가 기능이 제품이 추가됨에 따라 이전 프로토타입에서는 분명하지 않았던 시스템 아키텍처와 관련된 문제가 발생할 수 있음

[소프트웨어 프로토타이핑(Software Prototyping)]

- 소프트웨어 응용 프로그램의 프로토타입, 즉 개발중인 소프트웨어 프로그램의 불완전한 버전을 만드는 활동
- 프로토타입은 일반적으로 최종 제품의 몇 가지 측면만 시뮬레이션 하면 최종 제품과 완전히 다를 수 있음
- 소프트웨어 설계자와 구현자가 프로젝트 초기에 사용자로부터 귀중한 피드백을 얻을 수 있고, 고객과 계약자는 소프트웨어 프로그램이 구축 된 소프트웨어 사양과 일치하는지 비교할 수 있음
- 사용자와 많은 상호작용이 있는 시스템에서 가장 유용함. 일괄 처리 또는 대부분 계산을 수행하는 시스템과 같이 사용자 상호 작용이 거의 없는 시스템은 프로토타이핑의 이점이 거의 없음

프로토타이핑 유형	
일회용 프로토타이핑	최종 제공 소프트웨어의 일부가 되지 않고 결국 폐기될 모델을 만드는 것. 신속한 프로토타이핑, 사용자가 테스트 할 수 있는 인터페이스 구현
진화형 프로토타이핑	구조화 된 방식으로 매우 강력한 프로토타입을 구축하고 지속적으로 개선하는 것. 구축 될 때 새로운 시스템의 핵심을 형성하고 개선 및 추가 요구 사항이 구축됨
증분 프로토타이핑	최종 제품은 별도의 프로토타입으로 제작됨. 마지막으로 별도의 프로토타입이 전체 디자인에 병합됨. 증분 프로토타이핑을 통해 사용자와 소프트웨어 개발자 간의 시간 차이가 줄어듦
익스트림 프로토타이핑	개발 프로세스로서의 익스트림 프로토타이핑은 특히 웹 애플리케이션 개발에 사용됨

- 장점

  시간 및 비용 감소, 사용자 참여 향상 및 증가

- 단점

  불충분한 분석, 프로토타입과 완성된 시스템의 사용자 혼동, 사용자 목표에 대한 개발자의 오해, 프로토타입에 대한 개발자 부착, 과도한 프로토타입 개발 시간, 프로토타이핑 구현 비용

[나선형 모델(Spiral model)]

- 프로젝트의 고유한 위험 패턴을 기반으로 나선형 모델은 팀이 증분, 폭포, 진화 프로토타이핑과 같은 하나 이상의 프로세스 모델 요소를 채택하도록 안내

- 개발 단계

  요구 사항 및 위험 분석 – 계획 및 아키텍처 반복 – 구현 – 테스트 및 확인

- 나선형 모델의 6가지 불변성

  1) 동시에 이슈 정의
  2) 매 주기마다 네 가지 기본 활동 수행
  3) 위험은 노력의 수준 결정
  4) 위험은 세부 사항의 정도 결정
  5) 앵커 포인트 마일스톤 사용
  6) 시스템에 초점 된 라이프 사이클

- 장점 : 클라이언트가 최소한 프로토타입을 미리 볼 수 있고, 피드백 가능. 반복적 개발 단계에서 어떠한 문제가 생겼다면 그것은 가장 최근에 변경한 작업 혹은 추가된 작업일 확률이 높으므로 잘못된 부분을 쉽게 알아 낼 수 있어서 버그로 인한 시간낭비를 줄일 수 있음

- 단점 : 반복에서 오는 장점을 극대화 시키기 위해서 자주 반복을 실시해야 함

[폭포수 모델(Waterfall model)]

- 개발의 흐름이 폭포수처럼 지속적으로 아래로 향하는 모델
- 개발 단계

  소프트웨어 요구사항 기술 -〉 소프트웨어 설계 -〉 소프트웨어 구현 -〉 시험과 디버깅 -〉 설치 -〉 소프트웨어 유지보수
- 완전히 순차적으로 한 단계씩 진행해 나가야 하며, 전 단계가 수행되어 완료되기 전까지는 다음 단계로 진행할 수 없음

[브이 모델(V-Model)]

- 폭포수 모델의 확장된 형태 중 하나
- 폭포수 모델과 달리 코딩 단계에서 위쪽으로 꺾여서 알파벳 v자 모양으로 진행됨
- 개발 생명 주기의 각 단계와 그에 상응하는 소프트웨어 시험 각 단계의 관계를 보여줌
- 소프트웨어 개발의 각 단계마다 상세한 문서화를 통해 작업 진행
- 테스트 설계와 같은 테스트 활동을 코딩 이후가 아닌 프로젝트 시작 시에 함께 시작하여, 전체적을 많은 양의 프로젝트 비용과 시간을 감소 시킴

- 검증(Verification) 단계

  요구사항 분석, 시스템 설계, 아키텍처 설계, 모듈 설계

- 유효화(Validation) 단계

  단위 테스트, 통합 테스트, 시스템 테스트, 인수 테스트

3. 팀에 적용할 방법론 제안

- 애자일 모델 중 하나인 칸반 방식으로 스케줄을 조정할 수 있고, 프로토타이핑을 이용하여 빠르게 개발을 확인해 볼 수 있기 때문에 이 두가지 방법을 같이 사용하는 것이 좋을 것 같다.