소프트웨어공학 활용

소프트웨어 개발 라이프사이클

한국기술교육대학교
온라인평생교육원

학습내용

- 소프트웨어 개발 라이프사이클 개요
- 소프트웨어 개발 라이프사이클 예시

학습목표

- 소프트웨어 개발 라이프사이클의 특성과 기능, 프로세스를 설명할 수 있다.
- 소프트웨어 개발 라이프사이클의 모델유형을 이해하고 적합한 모델을 선정할 수 있다.

소프트웨어 개발 라이프사이클 개요

1 SDLC 등장 배경과 개념

1 SDLC(Software Development Life Cycle) 정의

정의 과정

개발 과정

유지보수 과정

폐기 과정

소프트웨어를 개발하기 위해 하나의 연속된 주기로 보고, 효과적으로 수행하기 위한 모델화 과정

프로세스 모델 또는 소프트웨어 공학 패러다임

소프트웨어공학을 실제 구현하기 위해 사용되는 프레임워크로서 소프트웨어 개발 생명주기 모델

타당성 검토부터 시작하여, 개발, 폐기 등 전 과정을 생명주기로 간주하고, 정의하여 단계별 공정을 체계화한 모델

소프트웨어 개발 라이프사이클 개요

1 SDLC 등장 배경과 개념

2 등장 배경 및 필요성

1. 소프트웨어 위기로 인한 체계적인 소프트웨어 개발 필요
2. 소프트웨어 개발을 효과적으로 수행하려는 방안 모색
3. 고품질 소프트웨어의 생산성 확보 필요

3 소프트웨어 위기

NEWS

이 용어는 컴퓨터 계산 용량과 문제의 복잡성이 급격한 증가로 발생한 충격을 서술하기 위해 사용되었습니다. 이는 정확히 이해할 수 있고, 검증 가능한 컴퓨터 프로그램을 작성하는 것이 얼마나 어려운가를 뜻합니다. 소프트웨어 위기의 뿌리는 복잡성, 기대, 그리고 변화입니다.

상충하는 요구조건들은 항상 소프트웨어 개발 과정에 지장이 되어 왔습니다. 예를 들어, 사용자는 많은 수의 기능을 요구해왔지만, 구매담당자는 일반적으로 소프트웨어의 가격과 개발 시간을 최소화하기를 원했습니다.

소프트웨어 개발 라이프사이클 개요

1 SDLC 등장 배경과 개념

3 소프트웨어 위기

1 원인

- 소프트웨어 규모의 대규모화, 복잡화에 따른 개발비용 증대
- 하드웨어 비용에 대한 소프트웨어 가격 상승 폭 증가
- 유지보수의 어려움과 개발 정체 현상 발생
- 프로젝트 개발 및 소요예산 예측의 어려움
- 신기술에 대한 교육 및 훈련의 부족

소프트웨어 개발 라이프사이클 개요

1 SDLC 등장 배경과 개념

3 소프트웨어 위기

2 증상

- 프로젝트 예산 초과
- 프로젝트 일정 지연
- 비효율적인 소프트웨어
- 소프트웨어 품질 저하
- 요구 사항을 만족하게 하지 못하는 소프트웨어
- 관리 불가능한 프로젝트와 관리가 어려운 코드
- 소프트웨어가 고객의 손에 전달 되지 못함

소프트웨어 개발 라이프사이클 개요

1 SDLC 등장 배경과 개념

3 소프트웨어 위기

3 대응 방안

1. 객체 지향 프로그래밍, 캡슐화
2. 구조적 프로그래밍
3. 통합 개발 환경, 신속 응용 프로그램 개발
4. 소프트웨어 컴포넌트화
5. 프로토타이핑
6. 애자일 개발 프로세스, 반복형 개발
7. 버그/이슈 관리 시스템, 버전 관리 시스템 도입
8. 디자인패턴
9. 가비지 콜렉션
10. 멀티스레드 프로그래밍

소프트웨어 개발 라이프사이클 개요

1 SDLC 등장 배경과 개념

4 SDLC 특성과 기능

SDLC 특성

- 프로젝트 수행 절차를 이행하기 위한 효과적인 도구
- 프로젝트의 유형, 관점, 개발 방침, 표준정책에 따라 다양한 방법 존재

SDLC 기능

- 프로젝트 비용의 산정
- 프로젝트의 개발 계획 수립
- 용어, 산출물 구성 등의 표준화 지원
- 개발 진행 상황 파악 지원 및 프로젝트 관리의 지원

5 SDLC 장·단점

장점

- 소프트웨어 개발과정에 대한 확실한 통제 가능
- 대규모 시스템 개발에 적합
- 문서화 용이

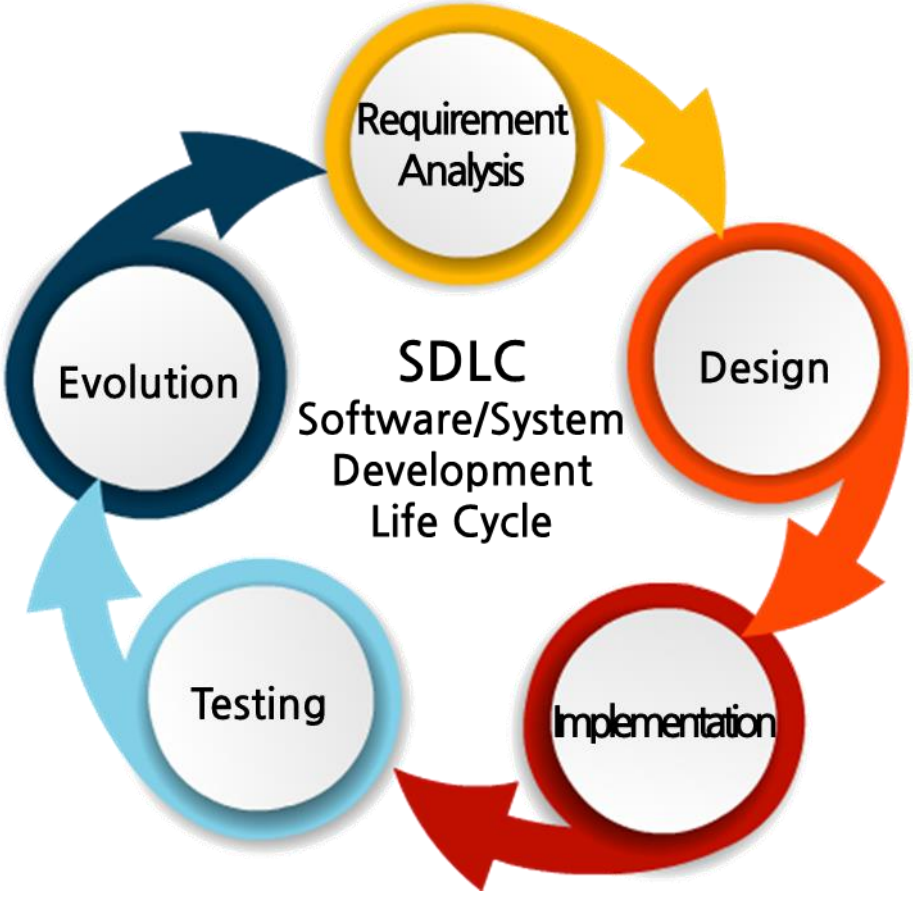
단점

- 사용자 요구사항의 변경에 적절히 대응하기 어려움 (시스템 개발의 경직성)
- SDLC의 후반부에는 사용자의 요구사항을 반영하기 어려움

소프트웨어 개발 라이프사이클 개요

2 SDLC 프로세스

1 정의 단계



소프트웨어 개발 라이프사이클 개요

2 SDLC 프로세스

1 정의 단계

타당성 검토	요구사항 분석	설계
• 요구사항을 만족하게 하기 위한 대안을 분석하는 작업 수행 • 고객과 사용자가 원하는 바를 명세화하고 타당성 조사	• 요구사항 식별 및 상세화 과정 (추출, 분석, 정의 검증, 관리) • 타당성 검토 시 선택된 시스템 개발에 대한 요구사항을 식별, 상세화하는 과정	• 고객의 요구사항에 기초하여 프로그램 개발을 위한 사양 작성 (아키텍처, 프레임워크, 디자인 패턴) • 개발 프로세스에 대한 새로운 요구사항을 관리하기 위한 공식적인 변경관리 프로세스 정의

2 개발 단계

개발

- 프로그래밍을 착수하고 시스템을 운영할 수 있는 형태를 갖춤
- 프로그래밍 및 실행코드 생성 (프로그래밍기법, 표준화)

테스트

- 다양한 수준의 테스트를 통해 개발 시스템에 대한 검토와 검증이 수행

소프트웨어 개발 라이프사이클 개요

2 SDLC 프로세스

3 지원 단계

설치 / 이행

- 새로운 정보시스템에 대하여 운영환경을 구축하고 사용자 인수 테스트를 수행
- 운영환경의 구축 후 사용자 인수 테스트 및 교육

유지 보수

- 문제점 수정 및 기능 추가 등 인수 완료 후 발생하는 모든 개발 활동

폐기

- 새로운 정보시스템 개발, 비즈니스 변화로 인한 시스템 폐기

4 정리

단계	프로세스	내용
정의 단계	타당성 검토	고객과 사용자가 원하는 바를 명세화하고 타당성 조사
	요구사항 분석	요구사항 식별 및 상세화 과정
	설계	요구사항에 기초하여 시스템 및 프로그램 설계
개발 단계	개발	프로그래밍 및 실행코드 생성
	테스트	시스템에 대한 검토와 확인

소프트웨어 개발 라이프사이클 개요

2 SDLC 프로세스

4 정리

단계	프로세스	내용
지원 단계	설치 / 이행	운영환경의 구축 후 사용자 인수 테스트 및 교육
	유지보수	문제점 수정 및 기능 추가 등 인수 완료 후 발생하는 모든 개발 활동
	폐기	새로운 정보시스템 개발, 비즈니스 변화로 인한 시스템 폐기

소프트웨어 개발 라이프사이클 예시

1 SDLC 모델 유형

1 대표적 모델 유형 종류

- 폭포수 모델
- 원형모델
- 나선형 모델
- 반복적 모델
- RAD 기법 모델
- 4세대 모형
- 클린룸 모델
- V모델

2 폭포수 모델(Waterfall)

1 정의

폭포수 모델이란? 개념 정리에서 구현까지 하향식 접근 방법으로 추상화 Modeling 실행하는 모델

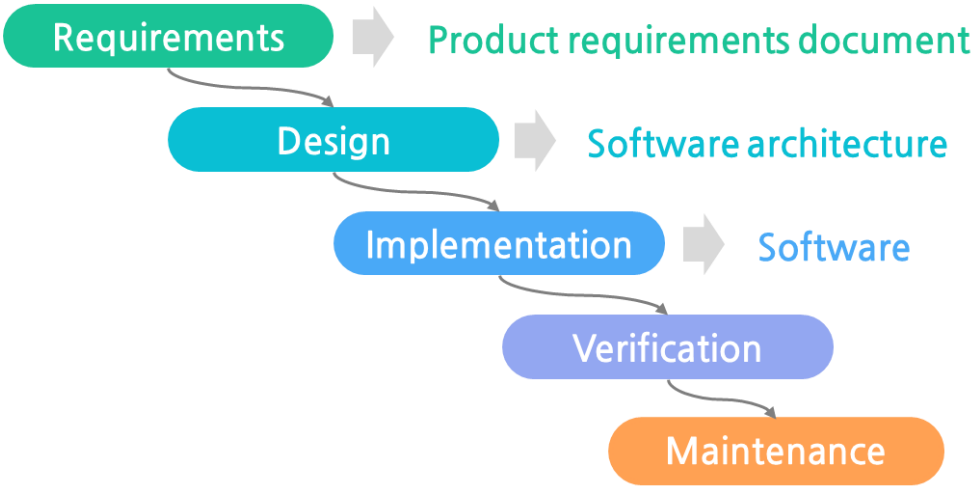
1. 가장 많이 사용되었던 전통적인 모델
2. 소프트웨어 개발을 단계적, 순차적, 체계적 접근 방식으로 수행
3. 각 단계별로 철저히 매듭짓고 다음단계로 진행함
4. 검토(Validation) 및 검증(Verification), 검사(Test)에 의해 프로젝트 전반의 품질 향상 추구

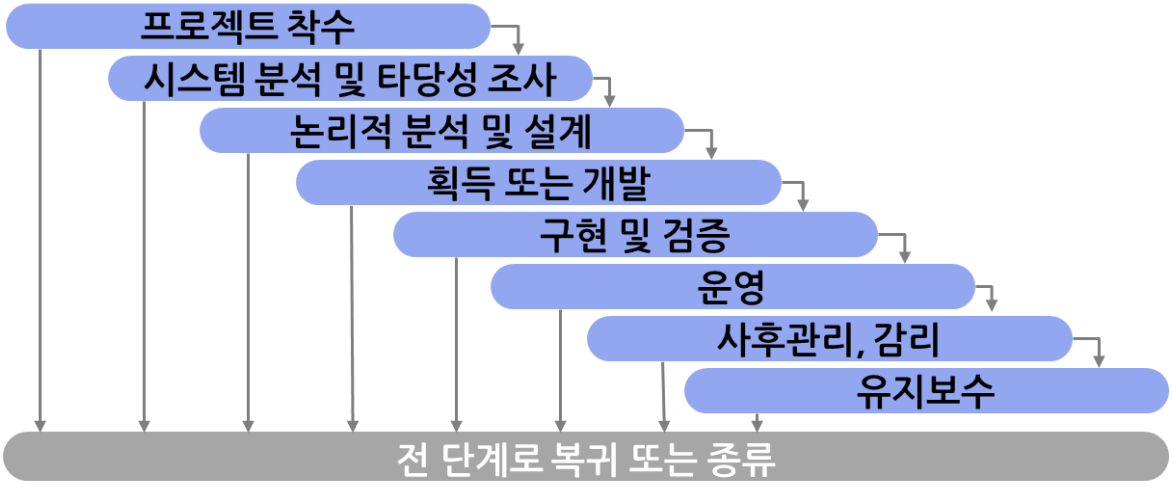
소프트웨어 개발 라이프사이클 예시

1 SDLC 모델 유형

2 폭포수 모델(Waterfall)

② 특징





소프트웨어 개발 라이프사이클 예시

1 SDLC 모델 유형

2 폭포수 모델(Waterfall)

3 장점과 단점

장점

- 복잡성이 낮고 진행 과정을 세분화하여 관리가 용이
- 사례가 풍부하고, 전체 과정에 대한 이해가 용이

단점

- 소프트웨어의 거대화 및 요구사항의 구체화가 어려움
- 각 단계마다 불필요한 Document가 많이 양산될 가능성
- 시스템이 개발 완료되는 시점에 완성이 가능
- 각 진행단계 문제 발생 시 이전단계로 피드백이 되는 경우 발생

소프트웨어 개발 라이프사이클 예시

1 SDLC 모델 유형

3 원형 모델(Prototyping)

1 정의

원형 모델이란?

- 고객과 원활한 의사소통을 통한 개발모델 (폭포수 모델 단점 보완)
- 사용자의 요구사항을 충분히 분석할 목적으로 시스템의 핵심적인 기능을 먼저 만들어 평가한 후 구현하는 점진적 개발 방법

2 특징

사용자 요구를 더 정확히 반영 가능

시스템 이해도가 낮은 관리자가 있는 경우 유용함

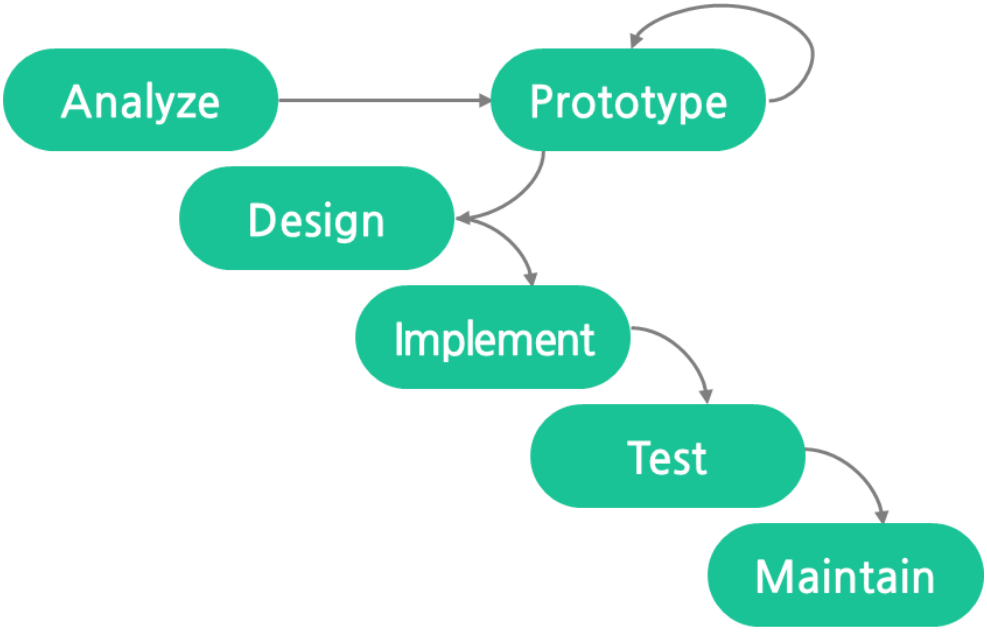
개발 중에도 유지보수 효과가 있음

소프트웨어 개발 라이프사이클 예시

1 SDLC 모델 유형

3 원형 모델(Prototyping)

2 특징



3 장점과 단점

장점

- 사용자 요구사항의 불명확 시 사용 용이
- 관리자의 이해가 용이
- 예상 운영상태의 예측이 가능함
- 제품의 추적성, 시험 가능성 확보

단점

- 개발 완료로 착각하기 쉬움
- 문서작성 미흡 및 경시, 산출물 부재 발생
- 폐기 시 Overhead, 낭비 발생
- 사용자의 과다한 요구사항 발생 가능

소프트웨어 개발 라이프사이클 예시

1 SDLC 모델 유형

3 원형 모델(Prototyping)

4 프로세스

- 사용자가 개발된 프로토타입을 실제 사용함으로써 요구사항이 이행되고 있는지 확인
- 프로토타입의 보완을 위한 여러 가지 제안

- 기본적인 사용자 요구사항을 분석
- 시스템 설계자는 기본 요구사항이 도출되기까지 사용자와 함께 작업

1단계 → 2단계 → 3단계 → 4단계 → 5단계

- 시스템 설계자가 1단계 도출된 요구사항을 만족하게 하는 프로토타입을 프로그래밍 언어 또는 알고리즘 도구를 이용해 개발
- 프로토타입은 앞으로 개발될 시스템의 가장 핵심적인 기능 위주로 개발

- 프로토타입의 수정과 보완이 이루어짐
- 시스템 설계자는 사용자가 요구한 모든 제안사항과 이에 따르는 보완작업을 하게 됨
- 프로토타입이 수정된 후에는 3단계로 돌아가며, 사용자가 만족할 때까지 3단계와 4단계는 계속 반복

- 가장 중요한 단계로 평가는 고객과 개발자가 함께해서 개발될 소프트웨어의 요구사항을 정제하는 데 중요한 정보로 활용

소프트웨어 개발 라이프사이클 예시

1 SDLC 모델 유형

4 RAD 기법 모델(Rapid Application Development)

1 정의

RAD 기법 모델이란?

- 2~3개월의 짧은 개발 주기 동안 소프트웨어를 개발하기 위한 순차적인 프로세스 모델
- Code Generation에 의한 신속한 시스템 개발
- 폭포수 응용 형태로 컴포넌트 기반 지원

2 특징

1. 원형 모델 방식 기준 사용자 요구사항, 분석, 설계 개발을 신속한 시스템으로 개발
2. 제한된 범위의 단독 시스템을 CASE와 같은 다양한 도구를 활용하여 신속히 개발
3. 고급언어의 모호성을 해결하기 위해 형식 규격 언어로 표현하려는 노력 진행
4. 전통적인 SDLC의 사용자 참여 미흡, 늦은 생명주기를 극복하기 위한 대안

소프트웨어 개발 라이프사이클 예시

1 SDLC 모델 유형

4 RAD 기법 모델(Rapid Application Development)

3 장점과 단점

장점

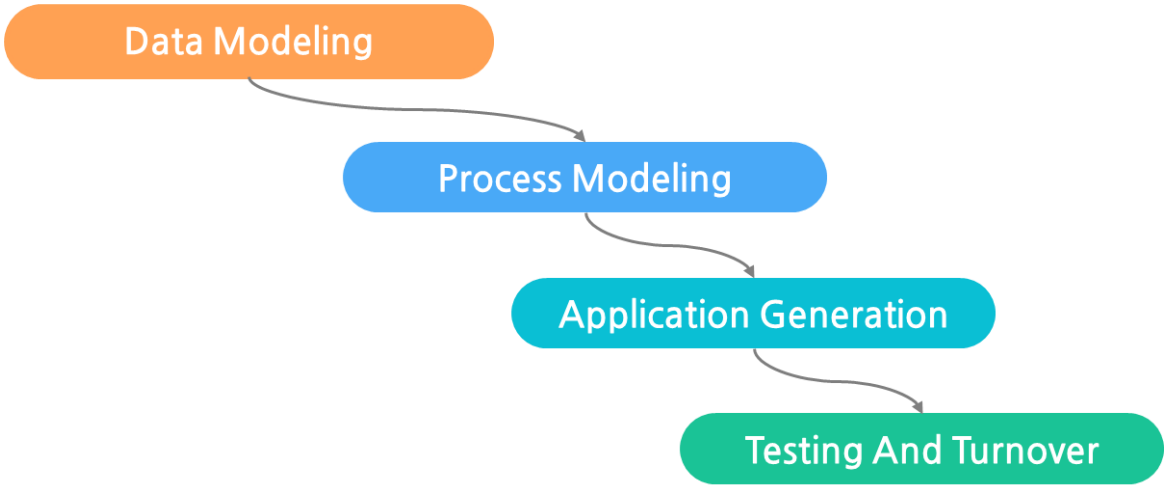
- 요구사항의 완전한 이해와 프로젝트 범위의 명확한 설정 시 신속한 개발 가능
- COTS 기반 개발을 활용하여 많은 응용시스템이 매우 낮은 비용으로 구현 가능

단점

- 위험성이 높거나 대규모 프로젝트에 부적합
- 구성원의 책임감이 낮을 경우 실패 가능성이 높음

COTS(Commecial Off The Shelf) :
특별히 주문 없이 구할 수 있는 기존 제품

4 프로세스

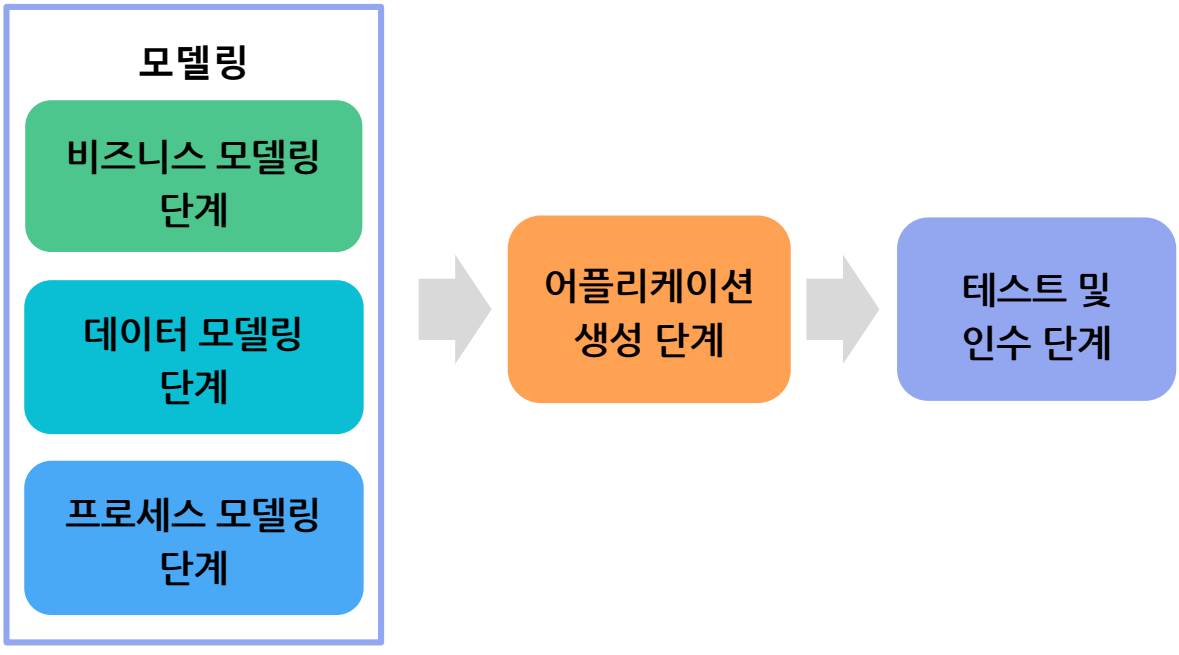


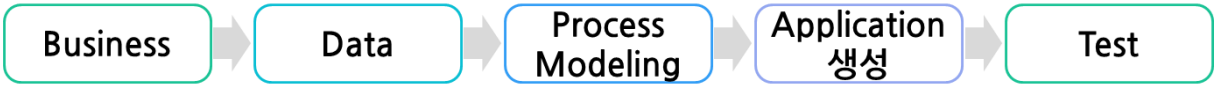
소프트웨어 개발 라이프사이클 예시

1 SDLC 모델 유형

4 RAD 기법 모델(Rapid Application Development)

4 프로세스





1단계 - 분석	• JRP : Joint Requirement Planning • 고객과 Business 모델 작성/검토 반복을 통한 분석
2단계 - 설계	• JAD : Joint Application Design • 고객과 원형모델 개발/수정/보안 반복 통한 시스템 설계
3단계 - 구축/운영	• 관련 기술을 이용하여 시스템 구축/운영 • CASE(Computer-aided Software Engineering) 사용

소프트웨어 개발 라이프사이클 예시

1 SDLC 모델 유형

5 나선형 모델(Spiral)

1 정의

나선형 모델이란?

소프트웨어의 기능을 나누어 점증적으로 개발하는 모델로, 시스템을 개발하면서 생기는 위험을 최소화하기 위해 나선을 돌면서 점진적으로 완벽한 시스템으로 개발하는 모델

2 특징

- 리스크 최소화를 위해 "위험분석" 단계가 존재
- 점진적으로 단계를 반복 수행해 나가는 모델
- 위험 부담이 큰 대형 시스템 구축에 적합

3 장점과 단점

장점

- 대규모 시스템 개발에 적합
- 프로젝트의 완전성 및 위험감소와 유지보수의 용이
- 반복적인 개발 및 테스트(강인성 향상)
- 한 사이클에 추가 못한 기능은 다음 단계에 추가 가능

단점

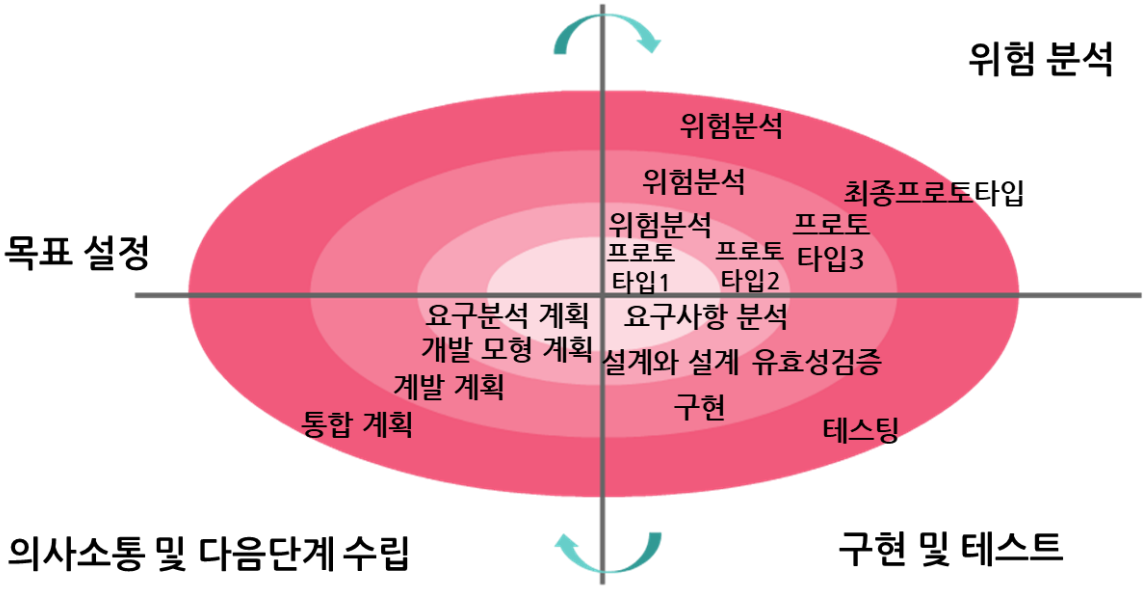
- 관리가 중요하나 매우 어렵고 개발시간이 장기화할 여지 있음
- 위험분석이 진행이 어려움
- 복잡한 프로세스에 대한 적응의 어려움

소프트웨어 개발 라이프사이클 예시

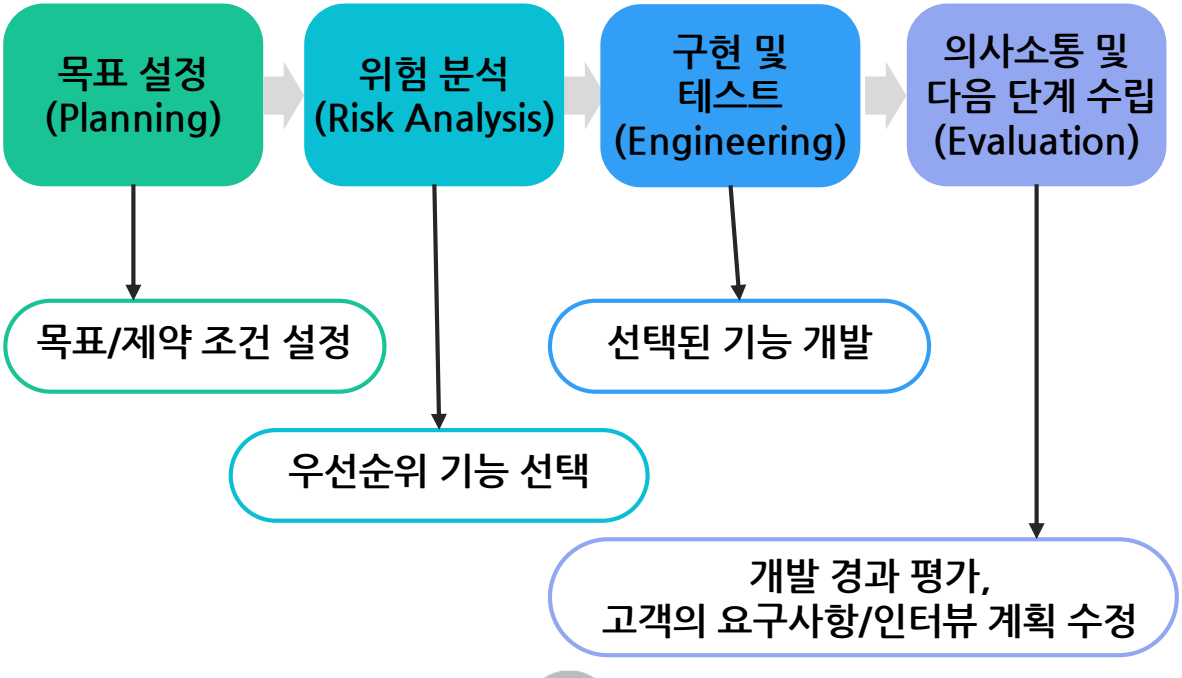
1 SDLC 모델 유형

5 나선형 모델(Spiral)

3 장점과 단점



4 프로세스



소프트웨어 개발 라이프사이클 예시

1 SDLC 모델 유형

6 4세대 모형(4th Generation Technique)

1 정의

4세대 모형이란?

CASE 및 자동화 도구를 이용하여 요구사항 명세로부터 실행 코드를 자동으로 생성할 수 있게 해주는 기법

2 특징

- 그래픽 표기법을 사용해 소프트웨어를 명시하는 능력에 초점을 맞춤
- CASE 도구들과 코드 생성기 결합
- 대규모 소프트웨어 개발에 있어 4GT(4 Generation Technique) 사용은 코딩을 제거함으로써 얻을 수 있는 실질적인 시간 절약을 위해 많은 분석, 설계 테스팅을 요구

3 장점과 단점

장점

- 생산성 향상
- 형식규격 언어를 사용함으로써 명세서 해석과 이해에 정확성을 향상할 수 있음
- 개발 과정을 자동화

단점

- 불필요한 많은 양의 코드 생성
- 유지보수가 어려움
- 4세대 모델 도구의 활용이 활성화되지 않음

소프트웨어 개발 라이프사이클 예시

1 SDLC 모델 유형

7 V 모델

1 정의

V 모델이란?

- 폭포수 모형에 시스템 검증과 테스트 작업을 강조한 모델
- 모듈의 상세 설계를 단위테스트 과정에 검증하고, 시스템 설계는 통합테스트 단계에, 사용자의 요구는 시스템 테스트 과정에서 검증하는 방법

2 특징

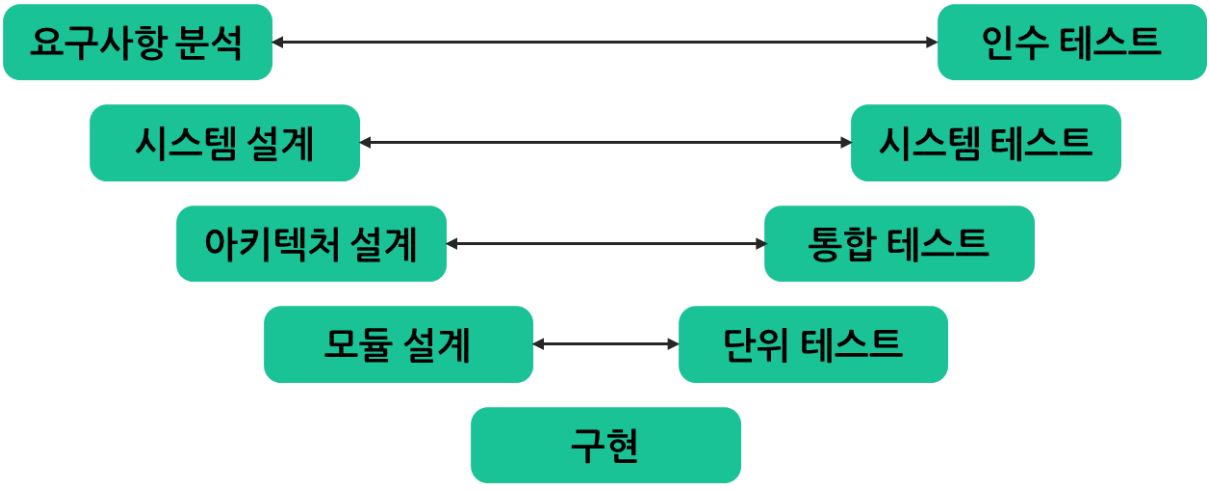
1. 소프트웨어 생명주기와 테스트 간의 관계를 도식화
2. 단계별로 상세문서화를 강조
3. 테스트 설계를 코딩 이후가 아닌 프로젝트 시작부터 함께 시작
4. 테스트 시간, 비용 감소

소프트웨어 개발 라이프사이클 예시

1 SDLC 모델 유형

7 V 모델

2 특징



3 장점과 단점

장점

- 간단하고 쉽게 이해하고 사용 가능
- 작은 프로젝트에 용이함
- 오류를 줄일 수 있음

단점

- 생명 주기 반복을 허용하지 않아서 변경을 다루기 어려움
- 작업 종료 후 리뷰 후에는 결과물이 동결됨

소프트웨어 개발 라이프사이클 예시

2 SDLC 모델 선정하기

1 모델 선정 기준

2 특징

1. 수행프로젝트의 규모와 성격에 따른 개발주기 선정 및 개발 생명 주기기반의 개발방법론과 관리 방법론을 도입
2. 도입된 개발, 관리 방법론과 연계하여 최대한의 생산성 확보 여부를 고려해야 함
3. 개발 소요시간, 비용, 기대 품질을 고려하여 불필요한 작업항목을 최소화하여 진행할 수 있는 기준을 도입
4. 개발 과정의 통제수단과 소프트웨어 산출물 인도 방식에 따라 개발모형을 선정해야 함

2 프로젝트 유형에 따른 SDLC 선정

일반적인 프로젝트구축	대규모 재구축 프로젝트	프로젝트 타당성 검토	연구형 개발	소규모
특성	위험성이 적고, 기존에 유사한 Reference가 많으며 한정된 자원에 대한 제약이 존재			
적용 가능한 SDLC 모델	폭포수 모델			

소프트웨어 개발 라이프사이클 예시

2 SDLC 모델 선정하기

2 프로젝트 유형에 따른 SDLC 선정

일반적인 프로젝트 구축 | **대규모 재구축 프로젝트** | 프로젝트 타당성 검토 | 연구형 개발 | 소규모

특성	프로젝트의 위험성이 높고 CBD 방법론을 활용, 대규모 프로젝트로 개발 도메인 간 연관성이 적음
적용 가능한 SDLC 모델	점진적 모델

Component Based Development) : 문제를 조각들로 나누어 각각의 컴포넌트를 생성한 후, 재조합하는 방식

일반적인 프로젝트 구축 | 대규모 재구축 프로젝트 | **프로젝트 타당성검토** | 연구형 개발 | 소규모

특성	프로젝트를 해야 하는지 결정이 어렵고 타당성 검증이 필요함
적용 가능한 SDLC 모델	프로토타이핑, 나선형 모델

소프트웨어 개발 라이프사이클 예시

2 SDLC 모델 선정하기

2 프로젝트 유형에 따른 SDLC 선정

일반적인 프로젝트 구축	대규모 재구축 프로젝트	프로젝트 타당성 검토	**연구형 개발**	소규모

특성	요구사항이 불명확하고 지속적인 검증이 필요, 충분한 비용이 확보되어 있음
적용 가능한 SDLC 모델	나선형 모델

일반적인 프로젝트 구축	대규모 재구축 프로젝트	프로젝트 타당성 검토	연구형 개발	**소규모**

특성	단기간 내 요구사항 만족, 자동화 도구 사용 가능, 고객의 참여를 통한 효율성 확보 가능
적용 가능한 SDLC 모델	RAD 모델

학습정리

1. 소프트웨어 개발 라이프사이클 개요

- SDLC(Software Development Life Cycle)는 소프트웨어를 개발하기 위한 정의 과정, 개발 과정, 유지보수 과정, 폐기 과정까지를 하나의 연속된 주기로 보고, 효과적으로 수행하기 위한 모델화 과정임
- SDLC 특성으로는 프로젝트 수행 절차를 이행하기 위한 효과적인 도구이며, 프로젝트의 유형, 관점, 개발 방침, 표준정책에 따라 다양한 방법이 존재함
- 소프트웨어 개발 과정에 대하여 정확한 통제가 가능하며 대규모 시스템 개발 시 적합함
- 개발 후반부에는 사용자의 요구사항을 반영하기 어렵기 때문에 초기 설계 시 충분한 회의 및 검토가 필요함
- SDLC는 타당성 검토, 요구사항 분석, 설계, 개발, 테스트, 설치/이행, 유지보수, 폐기 단계의 프로세스로 구성됨

학습정리

2. 소프트웨어 개발 라이프사이클 예시

- SDCL 모델 유형으로는 폭포수 모델, 프로토타이핑 모델, 나선형 모델, 반복적 모델, 클린룸 모델 등으로 발전함
- 폭포수 모델은 개념정리에서 구현까지 하향식 접근 방법으로 추상화 Modeling을 실행하는 모델
- 원형 모델은 프로토타이핑 모델이라고도 하며, 고객과 원활한 의사소통을 통한 개발 모델
- RAD 모델은 폭포수 응용형태로 짧은 주기의 소프트웨어 개발 프로세스 모델
- 나선형 모델은 소프트웨어 기능을 나누어 점증적으로 개발하는 모델로 발생할 수 있는 위험요소를 최소화하기 위하여 점진적으로 개발하는 모델
- 4세대 모형은 Case 및 자동화 도구를 이용하여 코드를 자동으로 생성할 수 있게 함
- V 모델은 폭포수 모델에 시스템 검증과 테스트를 강조한 모델
- SDCL 모델을 선정할 때에는 수행 프로젝트의 규모, 성격, 개발 주기, 개발 소요시간 및 비용 등 여러 기준을 토대로 적합한 모델을 선정해야 함