겜소공 시험 언급

1주차

1. 소프트웨어를 목적에 따라 분류하면 -> 응용, 시스템

2. 소프트웨어를 계층적으로 분류하면 -> 사용자 작성 < 응용 < 플랫폼

2주차

프로그램 내장 방식으로 발전 -> 폰 노이만

3주차-2

소프트웨어 개발의 위기

"소프트웨어의 대규모화"

- 유지보수의 어려움

- 개발 일정 및 소요 예산 예측의 어려움

- 프로젝트 예산 초과

- 프로젝트 일정 지연

- 낮은 품질 ex>메모리 문제

폭포수 모델

- 단계적

- 체계적

- 순차적

4-1

프로토타이핑 모델

- 전체 소프트웨어의 일부를 빠르게 제작하는 방식

+ 요구사항을 정확하게 판단할 수 있음

+ 실현 가능성을 빠르게 확인 가능

- 완제품으로 오해 -> 모자라 보여야 함(Tip)

- 프로토타입 개발 내용이 완제품에 반영되어 품질 저하 -> 런타임 에러 발생 (특히 서버)

나선형 모델

- 개발 단계 반복적으로 수행하여 점차적으로 안정적인 소프트웨어를 개발하는 방식

+ 큰 시스템을 안정적으로 개발할 때 효과적

+ 순차적으로 검증이 이루어지기 때문에 안정적인 문제 해결 가능

+ 유지보수와 개발의 일원화 가능

- 많은 투자가 필요하여 작은 규모의 프로젝트엔 어울리지 않음

- 관리 복잡

<지금까지 다룬 내용>

- 소프트웨어란?

- 왜 소프트웨어 공학이 필요한가?

- 소프트웨어 개발의 위기(원인 및 결과)

-소프트웨어 개발 프로세스(폭포수, 프로토타입, 나선형)

목차

- 왜 프로젝트가 실패하는가?

- 프로젝트 진행 예시

- 애자일 방법론

프로젝트 왜 망했을까?

\* 인간을 중심으로 한 프로세스로 바뀌고 있다. -> 인간 사이 관계 때문

- 소프트웨어의 대규모화

- 유지보수의 어려움

- 개발 일정 및 소요 예산 예측의 어려움 등

- Customer

- 자신의 경험에 맞추어 설명하려고 함

- 우리 집에 나무가 있는데 거기에 재미있는 그네를 달아주세요

- Project Leader

- 일정에 맞추어 빨리 수행하고자 하는 목적으로 가급적 쉬운 방향으로 이해

- 집에 있는 나무에 그네를 달자

- Analyst

- 요구 분석 단계에서는 Customer와 직접 대면하지 않는 경우가 많음

- 나무를 자르고 양 옆에 지지대를 달고 그네를 설치

- Developer

- 요구사항 명세 및 분석된 내용을 기반으로만 진행하려는 경향

- 나무에 끈 두개로 지지되는 판자 하나를 설치하자!

- Business Consultant

- 기업의 고문은 해당 기술의 전체 보단 과도한 사용자 입장이 되는 경우가 많다.

- 그네를 블링블링하게 만들어야 잘 팔릴 것임

- 베타 테스터 QA

- 베타 테스터는 엉망인 프로젝트 결과물을 전달받아 테스트

- 그네를 테스트하라고 했는데 올가미가 왔다.

- 영업

- 영업을 위해 특정 포인트만 강조

- 이 그네는 감성적인 그네입니다.

- Open Source 목적의 개발

- 많은 기능을 가지지만 사용은 공짜인 컨셉으로 광고하지만 실제 사용 시 많은 난관 발생

- 유지 보수

- 프로젝트 결과물이 가져야 항 정확한 기능이 확립되지 않은 상태에서 개발될 경우 많음

- 판자에 누가 올라가지 않을 줄...

- 문제 발생 후 회복 고려 안함

- 사용자가 원했던 결과

- 앞의 문제들은 실제로 발생할 이유가 없는 문제들

- 소통이 중요

- 계획 중심의 프로젝트 진행: 고객의 의견이 누락될 가능성이 큼

- 빅뱅 릴리즈: 한꺼번에 모든 기능이 릴리즈 되기 때문에 문제 발생 여부를 사전에 파악 못할 가능성이 큼

- 산출물 중심: 산출물과 실제 동작하는 SW와는 크게 다를 가능성이 있음

4-2

프로젝트 진행 예시

- 전체 개발 프로세스가 아래와 같이 잘 짜여 있었음

프로토타입 개발 -> 내부 릴리즈 및 검증 -> 개선 -> 내부 릴리즈 및 검증 -> 개선...

프로젝트가 성공적인이유

- 실제 동작하는 어플리케이션을 만들어 검증

- 사용자와 1주~한달 간격으로 회의 및 결과 전달

- 해당 분야를 잘 아는 전문가 투입

- 내부 릴리즈 및 검증의 체계와 (Issue Tracker 활용)

내부 릴리즈 및 검증 과정

검증팀 -> Issue Tracker 활용하여 오류 올림 -> PM이 확인 -> 개발팀, 디자인팀에 넘겨줌 -> 해결 후 검증팀에 해결 확인 요청 => 반복

★애자일 선언문★ **시험문제**

- 공정과 도구보다 개인과 **상호작용을**

- 포괄적인 문서보다 **작동하는 소프트웨어를**

- 계약 협상보다 **고객과의 협력을**

- 계획을 따르기 보다 **변화에 대응**하기를

- ★가장중요한 것은 "소통"

인문학적인 접근이 SW 안정화에 도움.

5주 - 1

<애자일 방법론>

공학 - 위기 - SE - 소통 - 애자일 방법론

애자일 원칙1

[우리의 최우선 순위는, 가치 있는 소프트웨어를 **일찍** 그리고 **지속적**으로 **전달**해서 **고객을 만족**시키는 것이다.]

* “일찍 그리고 지속적 전달”

애자일 원칙2

[비록 개발의 후반부일지라도 요구사항 변경을 환영하라. 애자일 프로세스들은 변화를 활용해 고객의 경쟁력에 도움이 되게 한다.]

애자일 원칙3

[작동하는 소프트웨어를 자주 전달하라. 두어 주에서 두어 개월의 간격으로 하되 더 짧은 기간을 선호하라.]

: 목표를 주기적으로 설정하라

애자일 원칙4

[비즈니스 쪽의 사람들과 개발자들은 프로젝트 전체에 걸쳐 날마다 함께 일해야 한다.]

애자일 원칙5

[동기가 부여된 개인들 중심으로 프로젝트를 구성하라. 그들이 필요로 하는 환경과 지원을 주고 그들이 일을 끝내리라고 신뢰하라.]

애자일 원칙6

[개발팀으로, 또 개발팀 내부에서 정보를 전하는 가장 효율적이고 효과적인 방법은 면대면 대화이다.]

애자일 원칙7

[작동하는 소프트웨어가 진척의 주된 척도이다.]

애자일 원칙8

[애자일 프로세스들은 지속가능한 개발을 장려한다. 스폰서, 개발자, 사용자는 일정한 소도를 계속 유지할 수 있어야 한다.]

애자일 원칙9

[기술적 탁월성과 좋은 설계에 대한 지속적 관심이 기민함을 높인다.]

애자일 원칙10

[단순함 – 필요 없는 일의 양을 최소화]

애자일 원칙11

[최고의 아키텍처, 요구사항, 설계는 자기 조직적인 팀에서 창발한다.]

애자일 원칙12

[팀은 정기적으로 어떻게 더 효과적이 될지 숙고하고, 이에 따라 팀의 행동을 조율하고 조정한다.]