1. **(a) What is SOFTWARE ENGINEERING?**

소프트웨어 엔지니어링은 소프트웨어 개발의 모든 측면(기술 개발, 프로젝트 관리, 소프트웨어 생산 지원, 조직, 재정 등)에 관련된 엔지니어링 원칙으로, 적절한 이론과 방법 그리고 툴을 사용 및 개발하여 문제를 해결하는 것이다.

**(b) Why is it important?**

점점 더 많은 개인과 사회가 첨단 소프트웨어 시스템에 의존하고 있으므로 신뢰할 수 있는 시스템을 경제적으로 신속하게 생산할 수 있어야 한다. 일반적으로 개인 프로젝트처럼 프로그래밍하는 것보다, 소프트웨어 엔지니어링 방법과 기술을 사용하는 것이 장기적으로 봤을 때 더 저렴하다. 많은 시스템에서 비용은 소프트웨어 사용 후에 소프트웨어를 변경하는 비용이 대부분이다.

**(c) What are differences between Software Engineering and Computer Science?**

Computer Science는 이론과 근본에 중점을 둔 것이고, Software Engineering은 유용한 소프트웨어를 개발하고 제공하는 실용성에 중점을 두고 있다.

**(d) Please define “good” software. Which attributes does “bad” software have?**

* **“Good” software**: 필요한 기능(functionality)과 성능(performance)을 사용자에게 제공해야 하며 유지 보수가 가능(maintainable)하고 신뢰(dependable)할 수 있으며 사용 가능(usable)해야 한다. 즉, Maintainability (고치기 쉬워야 한다.), Dependability & security (믿을 수 있어야 한다.), Efficiency (시스템 자원을 효율적으로 사용해야 한다.), 그리고 Acceptability (사용자가 실제로 사용하기 편해야 한다.) 의 속성들을 필수적으로 가지고 있어야 한다.
* **“Bad” software**: Good software의 속성을 반대로 가지고 있는 것이다. 사용자가 원하는 기능이 없고 안 좋은 성능을 가지며 유지보수도 어렵고 신뢰할 수 없으며 사용성도 떨어지는 것이다.

1. **(a) What are the software process?**

소프트웨어 시스템을 개발하는 데 필요한 일련의 활동으로, Software specification, development, validation, 그리고 evolution으로 구성된다.

* Software Specification: 고객과 엔지니어가 생산될 소프트웨어의 기능과 운영상의 제약 조건을 정의한다.
* Software Development: 소프트웨어 시스템을 설계하고 구현한다.
* Software Validation: 소프트웨어가 고객이 원하는 것을 수행하는지 확인한다.
* Software Evolution: 변화하는 고객 및 시장의 요구사항을 반영하여 소프트웨어를 변경한다.

**(b) Please compare plan-driven vs. agile processes. Which process did your team adopt in your team project? Why?**

* Plan-driven process: 모든 프로세스 활동이 사전 계획되고, 진행 상황은 이 계획과 비교하여 측정되는 프로세스이다.
* Agile process: 계획이 점진적으로 이뤄지기 때문에 변화하는 고객의 요구사항을 반영하여 프로세스를 변경하는 것이 쉽다.
* 우리 조의 프로젝트는 agile process를 적용하고 있다. 초반에는 requirement 등의 개요를 정의하고 프로젝트의 중후반에는 개발을 진행하면서 점진적으로 프로젝트의 명세를 구체화할 예정이다. 현재 상황에서는 프로젝트의 가능한 모든 상황을 예측하여 계획을 완벽하게 세우기 어렵고 추후에 변경될 가능성이 있기 때문이다.

**(c) What are the problems with the waterfall model? How to overcome these problems?**

Waterfall model은 plan-driven 모델로, 명세와 개발 단계가 분명하게 분리되어 있다.

2.1.Waterfall-model.eps

Waterfall model의 가장 큰 단점은 프로세스가 진행되는 동안 변경 사항을 적용하기 어렵다는 것이다. 원칙적으로 다음 단계로 이동하기 전에 단계가 완료돼야 하기 때문이다. 따라서 이 모델은 요구사항을 잘 이해하고 설계 프로세스 진행 중에 변경 사항의 발생이 상당히 제한될 때만 적합하다. (하지만 이런 비즈니스 시스템은 거의 없다.) Waterfall model은 시스템이 여러 곳에서 개발되는 대규모 시스템 엔지니어링 프로젝트에 주로 사용된다. 이런 상황에서 waterfall model의 plan-driven 특성은 작업을 조정하는 데 도움된다.

**(d) What are the benefits of incremental development? How to apply incremental development process for your class project?**

2.2 Incremental-dev.eps

* 변화하는 고객의 요구사항을 수용하는 비용이 절감된다. 다시 분석해야 하는 요구사항과 문서화의 양은 waterfall model보다 훨씬 적다.
* 완료된 개발 작업에 대한 고객의 피드백을 얻는 것이 더 쉽다. 고객은 소프트웨어 데모에 대해 논평하고 구현된 양을 확인할 수 있다.
* 보다 신속하게 고객에게 유용한 소프트웨어를 제공하고 배포할 수 있다. 고객은 waterfall process에서보다 빨리 소프트웨어를 사용하고 이득을 취할 수 있다.

지금까지 수업을 통해 작성한 SSRD 문서와 UML diagram을 참고하여 개발을 진행하면서 프로젝트의 세세한 명세를 구체화한다. 개발은 프로토타입을 작성하여 기능과 성능 및 사용성을 검증하면서 진행한다.

**(e) Please draw the software process model (a set of activities) of your class project. Please discuss pros and cons of your process model. How to overcome the cons of your process model?**

2.2 Incremental-dev.eps

Outline description:

Define requirements & UML diagrams

Outline description의 내용이 완벽하지는 않아서 개발하면서 회의가 필요한 부분이 생길 수 있다는 단점이 있다. 하지만 전체적인 개요는 잘 잡혔기 때문에 세세한 부분만 조정하면 충분히 극복 가능하다. 그리고 이런 incremental development를 통해 앞으로 생길 변동에 유연하게 대처할 수 있다는 장점이 있다.

**(e) What is reuse-oriented software engineering? How to utilize them for your class project? What are their risks? How to mitigate them?**

2.3 Reuse oriented SE.eps

Reuse-oriented software engineering은 시스템의 기존 구성 요소나 응용프로그램 시스템이 통합되는 소프트웨어의 재사용을 기반으로 한다. 재사용된 요소는 사용자의 요구사항에 맞게 동작 및 기능을 조정할 수 있다. 재사용은 다양한 유형의 비즈니스 시스템을 구축하기 위한 표준 접근 방식이다.

추가) Reusable software의 유형

* 특정 환경에서 사용하도록 구성된 stand-alone application(독립 실행형 응용프로그램) 시스템.
* .Net Framework 또는 J2EE와 같은 구성요소 프레임워크와 통합될 패키지로 개발된 개체의 컬렉션.
* 서비스 표준에 따라 개발되고 원격 호출에 사용할 수 있는 웹 서비스.

우리 조 프로젝트에는 네이버 지도 API나 웹 관련 프레임워크를 재사용할 것이다. 이럴 경우 우리 프로젝트와 재사용하는 소프트웨어를 결합하면서 사용자의 요구 사항을 충족시키지 못 할 수도 있고, 재사용한 요소의 새 버전이 우리 조 프로젝트의 통제 하에 있지 않기 때문에 시스템에 대한 일부 제어가 손실될 수 있다.

소프트웨어를 재사용할 경우 발생할 수 있는 risk는 재사용할 소프트웨어를 찾고 그것을 이해해서 프로젝트에 적용해야 하는 부담이 있고, 재사용된 소프트웨어의 소스코드나, 툴이 기존의 프로젝트와 호환될 수 없다면 보수 비용이 더 높아질 것이다. 이를 완화시키기 위해서는 재사용할 소프트웨어와 관련된 requirement의 정제 단계를 소프트웨어 엔지니어링 프로세스에 추가하고, 재사용할 소프트웨어 후보를 찾고 평가하는 단계도 명시적으로 추가해야 한다.

**(f) What is V-model? Please suggest the V-model for your class project with detailed milestones (e.g., deadline of each activities; deliverables).**

V-model은 plan-driven software process의 testing 단계에 사용되는 모델로, waterfall model의 각 단계에 해당하는 소프트웨어 유효성 검사 작업을 보여준다. 모델을 옆으로 돌려서 보면 V모양이어서 V-model이라고 한다.

우리 프로젝트의 V-model

2.7 Testing-phases.eps

**(g) How to make the evolution plan of your class project? What are the top 3 risks of your class project due to evolution? How to mitigate them?**

우리 조 프로젝트의 주요 기능은 아니지만 있으면 편리한 기능들이나 프로젝트 규모가 커져야 가능한 기능을 evolution plan으로 만들었다. 사진에 해쉬태그 기능 추가나 관리자 인터페이스 추가 및 광고 창 제작 등이다.

[재작업 비용]

* 변경되거나 새로운 시스템 요구사항이 생기기 때문에 요구사항을 재분석해야한다.
* 새로운 기능을 구현해야 한다.
* 플랫폼을 변경하려면 애플리케이션도 변경해야 한다.

[Risk를 완화시키는 방법]

* 상당한 재작업이 요구되기 전에 가능한 변경을 예상할 수 있는 활동을 소프트웨어 엔지니어링 프로세스에 포함해야 한다. 예를 들어, 프로토타입 시스템은 고객에게 몇 가지 핵심 기능을 보여주기 위해 개발될 수 있다.
* 비교적 낮은 비용으로 변경 사항을 수용할 수 있도록 프로세스를 설계한다. 이것은 incremental development의 형태를 포함한다. 제안된 변경 사항은 아직 개발되지 않은 increments로 구현될 수 있다. 이것이 불가능할 경우 변경 사항을 통합하기 위해 시스템의 작은 부분(single increment)만 변경할 수 있다.