

소프트웨어공학



강의노트

10주차 01차시
신속한 소프트웨어 개발

❖ 학습안내

이번 시간의 학습내용과 학습목표를 확인해보세요.

■ 학습내용

- 신속한 소프트웨어 개발의 개념
- 애자일(Agile) 및 익스트림(Extreme) 프로그래밍
- 소프트웨어 재사용

■ 학습목표

- 신속하게 소프트웨어를 개발할 수 있는 방법을 설명할 수 있다.
- 소프트웨어 개발 시 애자일 및 익스트림 프로그래밍 방식을 사용할 수 있다.
- 소프트웨어 재사용을 이용하여 소프트웨어를 개발할 수 있다.



❖ 학습내용

[1] 신속한 소프트웨어 개발의 개념

1. 소프트웨어 구현 단계


- ◆ 소프트웨어 구현 단계 이해
 - 앞으로는 **소프트웨어 구현 단계**에 대하여 배움
 - 다음 대화과정을 읽고 개발(구현)과정은 어떠한 과정인지 고민해 봄

1. 소프트웨어 구현 단계 사례 - 대화를 통한 소프트웨어 구현 단계 이해

1

와~
지금 십자수를
놓고 있구나!
나 줄 거니?

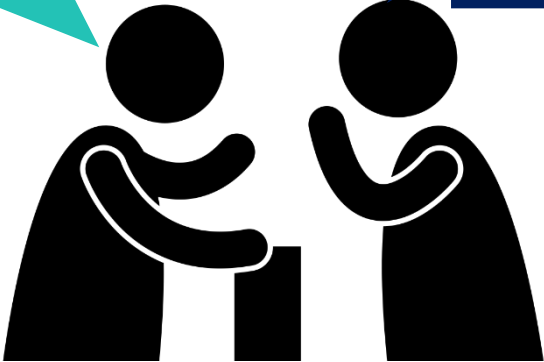
쏘리~
음.. 내 마음 속에
왕자님에게 줄 거야.
십자수를 한 뽕 한 뽕
놓는 것이 얼마나
어려운 일인 줄 아니?



2

음.. 십자수 놓는
일이나, 프로그램 개발하
는 일이나, 사람의 손이 무
척 많이 드는 일이지.

그렇지, 보기에는
단순하게 보여도,
나름 규칙도 있고
방법도 있지.



❖ 학습내용

[1] 신속한 소프트웨어 개발의 개념

1. 소프트웨어 구현 단계 사례 - 대화를 통한 소프트웨어 구현 단계 이해(계속)

3

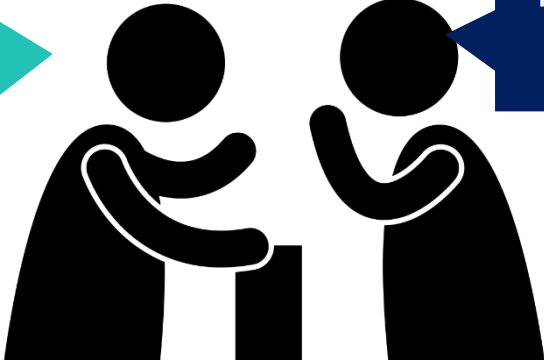
맞아, 어떻게 하면 빨리
십자수를 놓을 수
있을까? 하고
생각할 수 도 있고,
이전에 만들어 놓은
십자수를 보고
몇 개는 가져다가 만들
수도 있고,



3

만들어 놓은
부분품들을
짜맞추어 새로운
작품을 만들 수도
있지. 당연히
소프트웨어를
만드는 일도
그렇다고
볼 수 있겠지.

어쭙, 나와 같이 다
니니 천재가 되가는
구나.



이 사례로 보면,
소프트웨어를 개발하는 프로젝트의 구현 단계도
나름대로 **규칙과 방법**이 있으며,
다른 유사한 시스템의 경우도 참고하면서
개발하기도 하는 것을 알 수 있음

❖ 학습내용

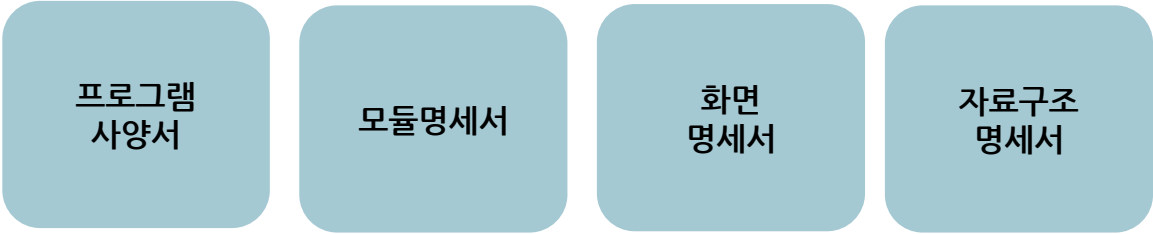
[1] 신속한 소프트웨어 개발의 개념

1. 소프트웨어 구현 단계(계속)

- ◆ 프로젝트의 구현 단계
 - 정보시스템 구축 프로젝트의 구현단계는 실제 소프트웨어(프로그램)를 프로그램 언어 등으로 작성하는 **작업 및 관련 하드웨어 등을 구축하는 단계**임
 - 프로젝트의 설계단계에서 작성된 프로그램 설계서, 데이터 설계서, 사용자 인터페이스 설계서, I/O 인터페이스 설계서의 내용이 **구체적인 프로그램이나 시스템으로 구현**되는 단계
 - 주요 활동
 - 일반적으로 프로그램 개발은 다수의 프로그램 개발자, 데이터베이스 관리자, 시스템 운영자 등 **다수의 인원**이 같이 작업이 이루어지므로 진행 전 **코딩 규칙**을 정한 후 작업
- ↓
변수, 상수, 수식, 코딩룰, 명칭규칙

◆ 프로젝트의 구현 단계

- 주요 산출물



◆ 구현 단계에서 주요 이슈

- 기업의 정보시스템은 기업이 목적으로 하는 비즈니스 업무처리를 잘 수행할 수 있는 도구이어야 함

1 기업 활동은 시간의 **즉시성(Time To Market)**을 가진 경우에는 보다 **빠르게** 소프트웨어가 개발되어야 함

❖ 학습내용

[1] 신속한 소프트웨어 개발의 개념

1. 소프트웨어 구현 단계(계속)

한 수준 아래의 글머리기호로
수정해주세요.

◆ 구현 단계에서 주요 이슈(계속)

- ◆ 기업의 정보시스템은 기업이 목적으로 하는 비즈니스 업무처리를 잘 수행할 수 있는 도구이어야 함

2

정보시스템이 재개발되는 경우나 유사한 업무를 개발하는 경우
기존 시스템이 재사용되기도 함

3

또 중요한 시스템을 구현하는 경우 보다 많은 고려사항이 있을 수
있음

4

사회 및 기술 변화에 따라 등장한 웹, 모바일, 게임시스템의
경우에도 시스템 구현의 고려해야 할 특징사항이 있음

5

아울러 이러한 정보시스템은 항상 개선되고 진화되어야 함

- 본 강의에서는 구현단계의 프로그램의 개발 활동 중 이러한 주제를 하나하나 다룸

❖ 학습내용

[1] 신속한 소프트웨어 개발의 개념

2. 신속한 소프트웨어 개발 개념

◆ 신속한 소프트웨어 개발 개요

- 매우 빠르게 변화하는 기업의 비즈니스(업무처리) 환경에 맞추기 위하여 **신속하게 소프트웨어를 개발**하여야 하는 경우가 대부분
- 매우 빠르게 변화하는 비즈니스 환경에 맞추기 위하여 신속하게 소프트웨어를 개발 하려는 경우다음과 같은 **절차와 방법**을 가지고 개발을 수행
- 이러한 개발방법의 공통적인 특징은 고객 서비스의 신속한 인도를 위하여, 모든 시스템을 최소단위의 **프로토타입**으로 먼저 개발하고 이를 **증분**하는 방식으로 개발
- 또 **사용자가 시스템개발에 참여**하여 그때그때 요구분석, 설계, 구현, 검증의 단위를 **동시에** 진행하고 검증하는 방법으로 이루어짐

◆ 신속한 소프트웨어 개발 진행순서

- 신속한 소프트웨어를 개발하기 위하여 다음과 같은 절차와 방법을 따르는 것이 효과적

최소단위 **프로토타입** 먼저 개발

그 후 필요한 것들을 **더하는 방법**으로 개발함

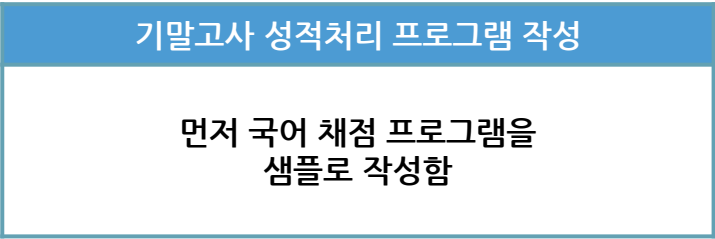
사용자가 요구분석, 설계, 구현, 배포 등
전 **개발과정에 참여**하고 **검증**함

❖ 학습내용

[1] 신속한 소프트웨어 개발의 개념

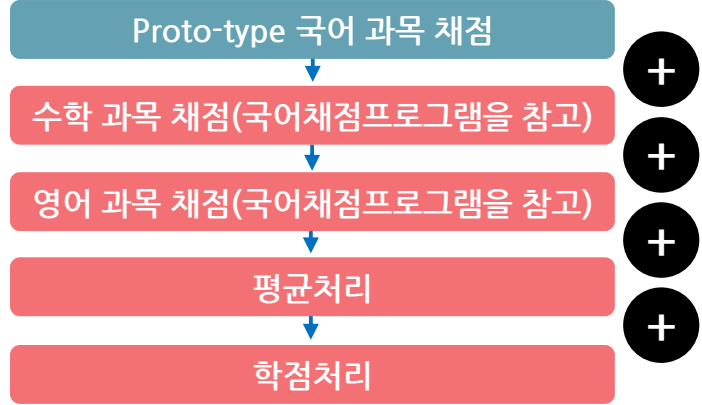
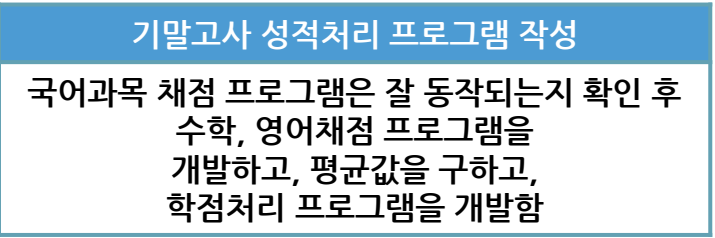
3. 신속한 소프트웨어 개발 순서

◆ 개발 순서



Proto-type 국어 과목 채점

- 최소단위 프로토타입 먼저 개발
 - 본 전체 시스템 중 최소단위의 프로토타입을 샘플로 작성함



- 그 후 필요한 것을 더하는 방법으로 개발함
 - 최소단위의 프로토타입을 **샘플 구현**하고 잘 작동되는지 확인
 - 이후 이와 비슷한 모듈을 분석하여 **동일한 부분을 참조**하여 개발

❖ 학습내용

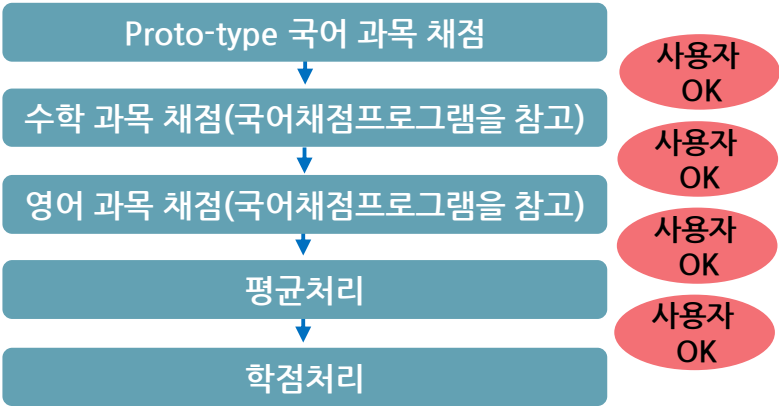
[1] 신속한 소프트웨어 개발의 개념

3. 신속한 소프트웨어 개발 순서(계속)

◆ 개발 순서(계속)

기말고사 성적처리 프로그램 작성

이 프로그램을 사용할 선생님들이
어떤 생각을 가지고 있는지 전 과정에
선생님들이 그때 그때 사용해보고
검토하라고 전달함



- 사용자가 요구분석, 설계, 구현, 배포 등 전 개발과정에 참여하고 검증함
 - 프로젝트 전 단계에서 사용자의 적극적 참여는 개발 일정 및 품질에서 가장 중요한 요소

❖ 학습내용

[2] 애자일(Agile) 및 익스트림(Extreme) 프로그래밍

1. 애자일 소프트웨어 개발

◆ 애자일 개발 개념

- 아무런 계획이 없는 개발 방법과 계획이 지나치게 많은 개발 방법들 사이에서 타협점을 찾고자 하는 방법론

계획이 없는 방법론의 경우,
앞으로의 일을 예측하기
힘들고 효율적이지 못함

계획에 너무 의존하는 경우는
그 형식적인 절차를 따르는데
필요한 시간과 비용이 많이 듭

- 고전적인 폭포수 모델 또는 나선 모형에서의 문서를 통한 개발 방법이 아니라 Less Document-oriented, Code-oriented를 추구



즉 문서를 통한 개발 방법이 아니라,
실질적인 코딩을 통한 방법론

- 애자일 개발 프로세스란 어느 특정 개발 방법론을 가리키는 말은 아니고 애자일 (Agile) 개발을 가능하게 해 주는 다양한 방법론 전체를 일컫는 말

기민한, 좋은 것을 빠르고 낭비 없게 만드는 것

- 이전에는 애자일 개발 프로세스는 경량(Lightweight) 프로세스, Lean Thinking으로 불렸음
- 익스트림 프로그래밍(XP: eXtreme Programming)은 애자일 개발 프로세스의 대표적인 방법

◆ 애자일 개발 배경

- 소프트웨어 위기의 원인과 해결방안을 찾는 데에서부터 시작
- 90년대 후반까지의 소프트웨어 공학과 개발방법론은 장기간에 걸쳐 많은 사람들을 투입하고 충분한 비용을 투입하여 진행하는 다른 공학의 프로세스와 비슷한 맥락에서 진행
- 그러나 소프트웨어는 유동적이고 개방적이고, 또한, 요구사항의 변경에 따른 작업량을 예측하기 힘들기 때문에 이미 고전적인 소프트웨어 공학이나 관리 기법만으로는 대처하는 데 한계가 있음
- 이런 문제에 대한 기술적인 해결책으로 객체지향 기법이 있음
- 객체지향 개발을 하기 위해서는 그에 적합한 개발 프로세스가 필요하였으며, 이에 따라 수많은 애자일 개발 프로세스가 만들어짐
- 따라서 애자일 개발 프로세스의 상당수는 객체지향 기술을 기반으로 하고 있음

❖ 학습내용

[2] 애자일(Agile) 및 익스트림(Extreme) 프로그래밍

1. 애자일 소프트웨어 개발(계속)

- ◆ 애자일 개발 기법
 - 빠른 소프트웨어 개발 방법 중 가장 많이 사용
 - 개발팀은 요건정의, 분석, 설계 등을 진행할 때, 기존에 익숙한 부분에 있어서 굳이 문서화, 명세화의 시간을 쏟는 것 보다는 소프트웨어의 구현, 품질 등에 초점을 맞추어야 함



Lean Thinking(낭비 없는 사고방식)을 기본 방침으로 함

“ 프로그램 개발 과정은
구현물을 만드는 목적이다.
이를 위하여 불필요한 사항이나,
비효율적인 부분은 과감히 배제한다. ”

개발팀이 설계와 문서화 보다 소프트웨어 자체에 초점을 맞추도록 하는 방식

❖ 학습내용

[2] 애자일(Agile) 및 익스트림(Extreme) 프로그래밍

1. 애자일 소프트웨어 개발(계속)

◆ 애자일 개발 기법(계속)

애자일 기법은
다음의 **원칙**을 준수



1. 고객이 **개발프로세스**에 참여
2. 시스템을 **점증적**으로 인도
3. 사람은 프로세스가 아니기 때문에 개발자 **자신만의 작업방식**의 **효율**을 찾음
4. 변경을 수용
5. 단순성을 유지

❖ 학습내용

[2] 애자일(Agile) 및 익스트림(Extreme) 프로그래밍

1. 애자일 소프트웨어 개발(계속)

◆ 애자일 기법의 원리

1 고객참여

- 고객은 **개발 프로세스 전체**에 긴밀하게 참여해야 함
- 고객의 역할은 **새로운 시스템 요구사항을 개발**하고 **우선순위를 결정**하고, **시스템의 반복을 평가**하는 것임

2 점증적인 인도

- 소프트웨어는 각 증분에 포함될 요구사항을 명세화하는 고객에게 **점증적**으로 인도됨

3 사람은 프로세스가 아님

- 개발팀의 기술이 인식되고 활용되어야 함
- 팀 구성원들은 규정된 프로세스 없이 **자신들의 작업방식을 개발**해야 함

4 변경을 수용

- 시스템 요구사항이 **변경될 것으로 예상**하여, 이 변경들을 **수용**하도록 시스템을 설계

5 단순성을 유지

- 개발 중인 소프트웨어와 개발 프로세스 모두 **단순성**에 초점을 맞춤
- 가능하면 시스템에서 복잡성을 제거하기 위해 **적극적으로 작업**함

❖ 학습내용

[2] 애자일(Agile) 및 익스트림(Extreme) 프로그래밍

2. 애자일 개발 방법 종류

◆ 애자일 개발 방법 6가지 종류

익스트림 프로그래밍 (Extreme Programming, XP)	<ul style="list-style-type: none">▪ 애자일 개발 프로세스의 대표자로 애자일 개발 프로세스의 보급에 큰 역할을 하였음▪ 이 방법은 고객과 함께 2주 정도의 반복개발을 하고, 테스트우선 개발(TDD)을 특징으로 하는 명시적인 기술과 방법을 가지고 있음
스크럼(Scrum)	<ul style="list-style-type: none">▪ 30일마다 동작 가능한 제품을 제공하는 스프린트(Sprint)를 중심으로 하고 있음▪ 매일 정해진 시간, 정해진 장소에서 짧은 시간의 개발을 하는 팀을 위한 프로젝트 관리 중심의 방법론
크리스탈 패밀리 (Crystal Family)	<ul style="list-style-type: none">▪ 이 방식은 프로젝트의 규모와 영향의 크기에 따라서 여러 종류의 방법론을 제공▪ 그 중에서 가장 소규모 팀에 적용하는 크리스탈 클리어는 익스트림 프로그래밍 만큼 엄격하지도 않고 효율도 높지 않지만, 프로젝트에 적용하기 쉬운 방법론
기능 주도 개발 방법론 (FDD: Feature Driven Development)	<ul style="list-style-type: none">▪ Feature마다 2주정도의 반복 개발을 실시▪ Peter Coad가 제창하는 방법론으로써, UML을 이용한 설계 기법과도 밀접한 관련을 가짐

❖ 학습내용

[2] 애자일(Agile) 및 익스트림(Extreme) 프로그래밍

2. 애자일 개발 방법 종류(계속)

◆ 애자일 개발 방법 6가지 종류(계속)

<p>적응형 소프트웨어 개발 방법론 (ASD:Adaptive Software Development)</p>	<ul style="list-style-type: none">▪ 소프트웨어 개발을 혼란 자체로 규정하고, 혼란을 대전제로 그에 적응할 수 있는 소프트웨어 방법을 제시하기 위해 만들어진 방법론▪ 내용적으로는 다른 방법론들과 유사하지만, 합동 애플리케이션 개발(Joint Application Development, 사용자나 고객이 설계에 참가하는 개발 방법론)을 사용하고 있는 것이 조금 다름
<p>익스트림 모델링 (Extreme Modeling)</p>	<ul style="list-style-type: none">▪ 익스트림 모델링은 UML을 이용한 모델링 중심 방법론임▪ 다만, 여타 모델링 방법들과는 달리, 언제나 실행할 수 있고 검증할 수 있는 모델을 작성하는 공정을 반복해서, 최종적으로는 모델로부터 자동적으로 제품을 생성하게 함

- 애자일 개발 프로세스들은 **각자 다른 특징과 적용 범위**가 있으며, 서로 **조합도 가능**
- 애자일 개발 프로세스를 채용하고 있는 프로젝트에서는 특정 방법론만을 채택해서 매뉴얼대로 **홍내만 내는** 것이 아니라, 여러 방법 중에서 자신의 프로젝트에 맞는 부분을 **취사 선택하여 조합**하고, 또 **독자적인 방법**을 만들어 냄으로써 큰 효과를 올리고 있음

3. 익스트림 프로그래밍

◆ 익스트림 프로그래밍 개요

- 애자일 기법 중 하나로 반복적인 개발을 수행하며, 이때 나타난 좋은 실무적 관행과 **고객의 참여를 극한(Extreme)수준**까지 끌어 올려서 개발하는 방법
- 익스트림 프로그래밍(eXtreme Programming, XP)은 **켄트 벡** 등이 제안한 소프트웨어 개발 방법
- 비즈니스 상의 요구가 시시각각 **변동이 심한 경우**에 적합한 개발 방법
- 해당 방법론에서 **수행원칙과 구체적인 실천방법(Practice)**를 제시함

❖ 학습내용

[2] 애자일(Agile) 및 익스트림(Extreme) 프로그래밍

3. 익스트림 프로그래밍(계속)

◆ 익스트림 프로그래밍 수행원칙

- ① 작은 부분의 구현, 변경과 시스템 변경적용을 자주하는 방법 등으로 **증가적으로 시스템 개발**을 함(Incremental development is supported through small, frequent system releases.)
- ② 사용자(고객)은 개발자와 같이 **한 팀으로 전 시간을 근무**하여 참여하도록 함 (Customer involvement means full-time customer engagement with the team.)
- ③ 사람은 처리프로세스가 아니기 때문에 페어 프로그래밍(두 사람 이상이 조를 이루어 같이 개발업무를 수행)이나 소스코드를 같이 공유하는 기법이나 어떤 절차(A process)등을 통하여 긴 근무시간을 피해야 함(People not process through pair programming, collective ownership and a process that avoids long working hours.)
- ④ 정기적인 시스템 변경 배포 등을 통하여 변경을 수용하여야 함(지나친 형상통제 배제)(Change supported through regular system releases.)
- ⑤ 계속적으로 소스코드 개선(Refactoring)을 통하여 유지보수를 단순하게 가져가야 함(Maintaining simplicity through constant refactoring of code.)

◆ 익스트림 프로그래밍 실천방법

Whole Team

- 이전 방법론에는 팀을 기반으로 한 소프트웨어 개발을 고려치 않은 실패가 많았음
- 애자일 방법론 및 XP 방법론이 유명세를 타기 시작하면서 팀을 기반으로 한 **협동적인 개발 방법론**이 강조되기 시작
- 첫 번째 실천 방법인 Whole Team은 모든 프로젝트에 **참여하는 팀원들**을 가리키며 개개인이 각자의 **역할**이 있고 그들의 **역할의 중요성**을 강조
- XP 에서 말하는 Whole Team은 일반적으로 Tester, Interaction designers, Architects, Project managers, Product managers, Executives, Technical writers, Programmers and users 로 구성이 됨
- 하지만 **가장 중요한 팀원은 사용자(User)**임
- 왜냐하면 그들이 프로젝트의 키를 가지고 있는 Stakeholder일 뿐만 아니라 그들을 통해 요구사항(Requirement)을 파악할 수 있기 때문임

❖ 학습내용

[2] 애자일(Agile) 및 익스트림(Extreme) 프로그래밍

3. 익스트림 프로그래밍(계속)

◆ 익스트림 프로그래밍 실천방법(계속)

Planning Game

- XP에서 계획을 세우는 데에는 중요한 2가지가 있음
- 첫 번째, '이번 반복(Iteration)에는 어떤 개발 과정을 끝마칠 것인가?' 두 번째, '그 이후 개발 반복(Iteration)에서는 무엇을 할 것인가?' 임
- XP는 일반적으로 2주를 주기로 계획을 세우고 프로토타입을 만들어서 최종 사용자 혹은 프로젝트를 의뢰한 의뢰인과 함께 무엇이 얼마만큼 개발이 되었는지 혹은 알맞은 방향으로 개발이 진행이 되어가고 있는지 **검사 혹은 회의**를 함
- 그러므로, 그 기한 안에 프로젝트 혹은 프로토타입은 반드시 개발이 완료가 되어야 함
- 그렇지 못해서 기한을 연장하게 되면, 그에 따른 **추가 비용 및 시간적 손실이 발생**하게 됨
- 이를 통해서 소프트웨어 개발의 **투명성**을 확보
- 이는 기업의 입장에서 좋은 점은 그들의 **실력 및 가능성을 보여줄 수 있는 기회**로 사용될 수 있음
- 사용자 혹은 의뢰인 입장에서는 더욱 큰 금전적 손실이 발생하기 전에 프로젝트를 취소하고 다른 개발팀을 찾아 볼 수 있는 기회로 사용될 수도 있음

Customer Tests

- 가장 흔히 발생하는 소프트웨어 개발 실패 중 하나는 개발이 끝난 제품 혹은 소프트웨어가 처음 사용자 혹은 의뢰인이 원했던 것과 다르다는 것
- 그렇기 때문에 XP에서는 **반복적으로 고객 테스트**를 거침
- 그리고 이를 통해서 그들이 잘못 이해하고 있었던 부분에 대해서 수정을 거치기도 하며 더욱더 의뢰인 혹은 최종 **사용자의 요구에 부합**하는 소프트웨어를 만들어 내게 됨

❖ 학습내용

[2] 애자일(Agile) 및 익스트림(Extreme) 프로그래밍

3. 익스트림 프로그래밍(계속)

◆ 익스트림 프로그래밍 실천방법(계속)

Small Releases

- 이 실천 방안을 통해서 개발자는 **주기적으로** 프로토타입을 의뢰인에게 보여줌
- 그리고 이를 통해서 의뢰인은 제한된 기능을 가지고 있지만 **실제로 작동이 되는 데모 모델**을 볼 수가 있으며 추가 사항을 요구할 수도 있음
- 기존의 혹은 과거의 소프트웨어 개발 과정에서 개발이 끝나기 이전에 어떤 소프트웨어가 만들어질지 알 수 없었던 것과 비교해서 **가장 차이점**을 만들어내는 실천 방안
- 또한 개발자에게 있어서는 현재까지의 개발 상황이 올바른 길로 가고 있음을 알 수 있음

Simple Design

- 보통의 프로젝트 개발 혹은 코딩의 경우 기능이 더해지면 더해질수록 **복잡**해지기 마련
- 그렇게 되면, 새롭게 총원되는 개발자의 경우에 현재까지의 개발 상황을 이해하는데 **오랜 시간**이 걸림
- 또한 버그가 발생할 경우에도 쉽게 처리를 하지 못하고 오랜 시간이 걸리게 됨
- 그러므로 XP에서 모든 코딩을 가능한 **간단하게** 할 것을 강조

Test-Driven Development

- 테스트를 기반으로 한 개발은 XP에서 가장 중요한 실천 방안 중 하나임
- **테스트**를 거치고 **코딩**을 하며 **프로젝트를 개발**해 나감

Pair Programming

- 두 명 혹은 그 이상의 프로그래머가 **함께** 코딩을 하는 것
- 두 명의 프로그래머가 함께 코딩을 하고 테스트를 통해서 개발을 할 수도 있고, 한 명은 코딩을 하고 한 명은 Quality Assurance 역할 통해서 테스트에만 집중을 할 수도 있음

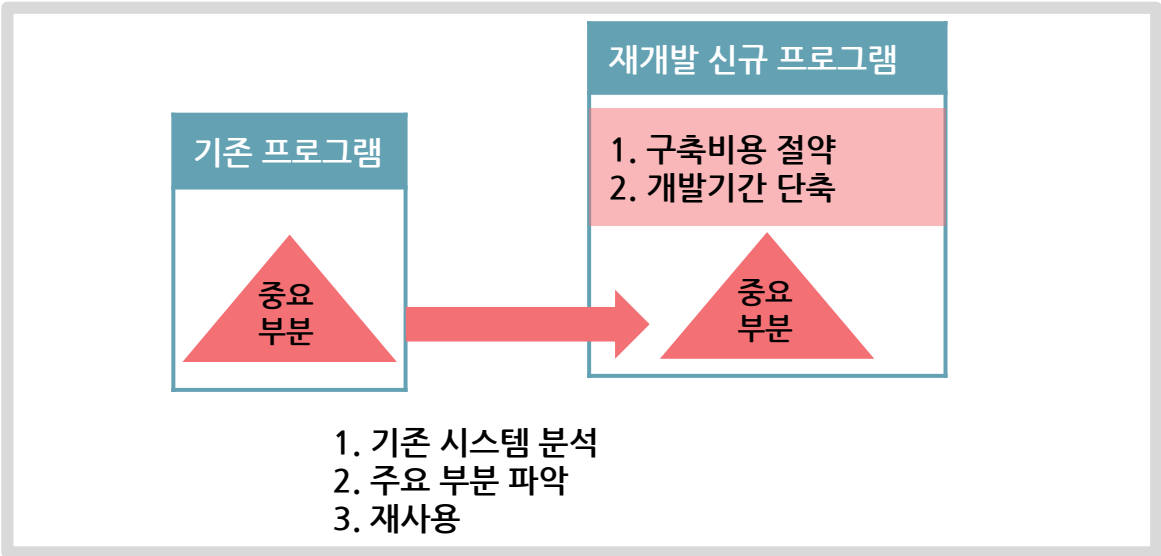
❖ 학습내용

[3] 소프트웨어 재사용

1. 소프트웨어 재사용

◆ 소프트웨어 재사용 개념

- 정보시스템을 구축하는 경우, 일반적으로 기존의 사용하던 시스템이 있거나, 유사한 시스템이 존재하는 경우가 많음
- 이런 경우 기존 시스템이나 유사 시스템의 일부를 **재사용**함으로써 **개발 효율**을 높일 수 있음



- 소프트웨어 설계는 기존 시스템이 존재한다면 **기존시스템의 분석**은 가장 중요한 부분
- 만일 기존 시스템의 많은 부분이 재사용이 가능하다면 **구축비용 및 개발기간의 단축**을 가져올 수 있음
- 소프트웨어 재사용을 효율적으로 할 수 있다면 장점이 발생하지만, 때로는 재사용 방식이 **문제점을 야기**하기도 함

◆ 소프트웨어 재사용의 장점



확실성의 증가

- 작동중인 시스템에서 시도되고, 시험된 재사용 소프트웨어는 설계와 구현상의 **결함**이 이미 **발견되어 수정**되었기 때문에, 새로운 소프트웨어보다 더욱 신뢰할 수 있음

❖ 학습내용

[3] 소프트웨어 재사용

1. 소프트웨어 재사용

◆ 소프트웨어 재사용의 장점 (계속)



프로세스 위험의 감소

- 기존 소프트웨어의 비용은 이미 알려져 있는 반면에, 개발 비용은 항상 판단을 필요로 하는 문제임
- 프로젝트 관리의 중요한 요인임
⇒ 이유: 프로젝트 비용 산정에 **오류범위**를 줄이기 때문임
- 서브시스템과 같이 비교적 **대규모 소프트웨어 컴포넌트**들이 재사용 될 때, 이러한 장점이 큼



전문가의 효과적인 이용

- 동일한 작업을 수 차례 반복하는 대신에, 이러한 응용 분야의 전문가들은 **자신들의 지식을 요약**한 재사용 가능한 소프트웨어를 개발할 수 있음



표준의 준수

- 사용자 인터페이스와 같은 어떤 표준은 **표준화된** 재사용 가능한 **컴포넌트의 집합**으로 구현될 수 있음

예

만일 사용자 인터페이스의 메뉴가 재사용 가능한 컴포넌트를 이용하여 구현되면, 모든 응용 시스템은 사용자에게 동일한 메뉴형식을 제시함

- 표준화된 사용자 인터페이스의 이용은 **확실성**을 **향상**시키는데, 그 이유는 익숙한 인터페이스가 제시될 때 실수를 할 가능성이 낮기 때문



개발속도의 향상

- 가능한 한 빨리 시스템을 출시하는 것이 전체적인 개발 비용보다 중요한 경우가 종종 있음
- 소프트웨어의 재사용은 **개발 및 검증시간**을 줄일 수 있기 때문에 **시스템생산속도**를 빠르게 할 수 있음

❖ 학습내용

[3] 소프트웨어 재사용

1. 소프트웨어 재사용

◆ 소프트웨어 재사용의 단점



유지보수 비용의 증가

- 만일 재사용된 소프트웨어 시스템이나 컴포넌트의 소스 코드를 활용할 수 없으면, **유지보수 비용이 증가**할 수 있음
⇒ 이유: 시스템의 재사용된 요소들이 **시스템 변경과 점점 일치하지 않게** 되기 때문임



도구 지원의 결여

- CASE도구 집합은 재사용을 포함하는 개발을 지원하지 않을 수 있음
- 도구들을 **컴포넌트 라이브러리 시스템과 통합**하는 것이 어렵거나 불가능 할 수 있음
- 도구들이 가정하는 소프트웨어 프로세스는 재사용을 고려하지 않을 수 있음



NIH 증후군

(Not-Invented-Here, 자신이 최고라는 생각으로 남의 것을 수용 못함)

- 어떤 소프트웨어 엔지니어들은 컴포넌트를 **다시 작성하는 것**을 선호함
⇒ 이유: 자신들이 컴포넌트를 개선할 수 있다고 믿기 때문임
- 이것은 부분적으로 **신뢰**의 관한 문제로 다른 사람의 소프트웨어를 재사용하는 것보다 소프트웨어를 원점에서 개발하는 것이 더욱 도전적으로 보인다는 사실과 부분적으로 관련이 있음



표준의 준수

- 재사용 가능한 **컴포넌트의 라이브러리**를 만들고 소프트웨어 개발자들이 이 라이브러리를 이용할 수 있게 보장하는 것은 비용이 많이 들 수 있음
- 소프트웨어 컴포넌트를 분류하고, 목록을 만들고, 검색하는 방법에 대한 기술은 현재 아직 발달되지 않았음



개발속도의 향상

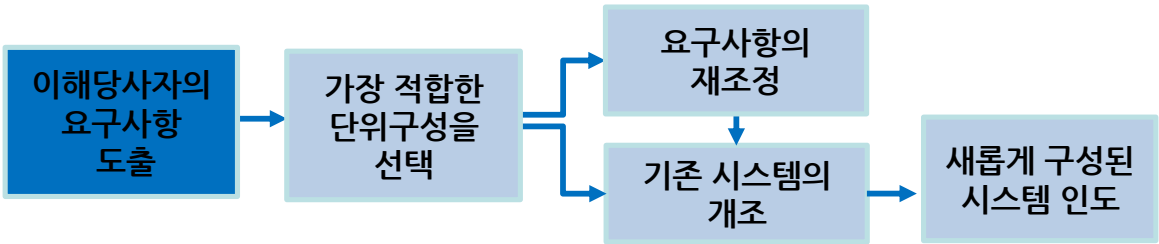
- 소프트웨어 컴포넌트들은 라이브러리에서 발견되고, 이해되고 때로는 **새로운 환경에서 동작**하도록 수정되어야 함
- 엔지니어들은 정상적인 개발 프로세스의 일환으로 컴포넌트 검색을 포함시키기 이전에 라이브러리에서 컴포넌트를 발견하는데 확신을 가져야 함

❖ 학습내용

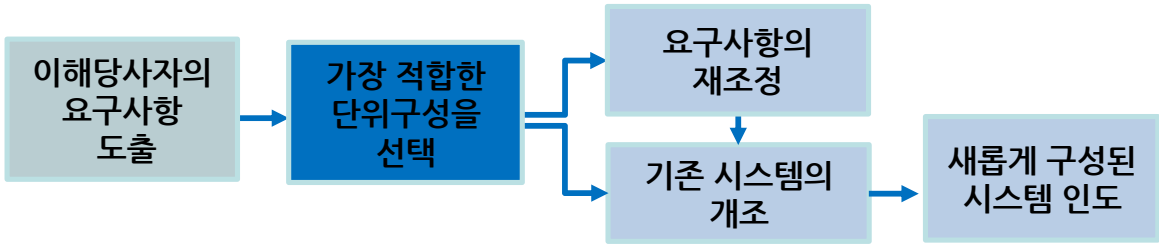
[3] 소프트웨어 재사용

2. 소프트웨어 재사용 흐름

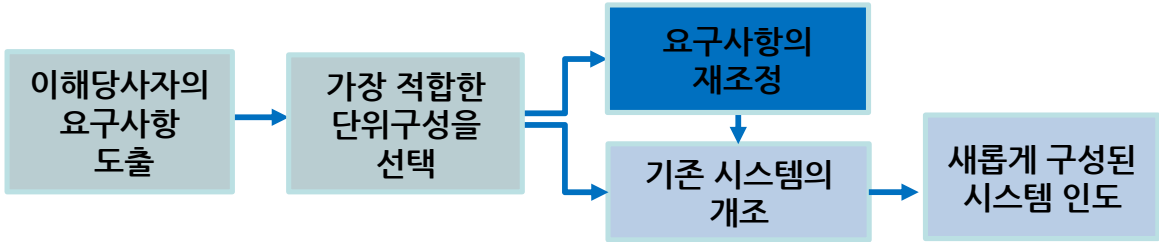
- ◆ 소프트웨어 재사용 절차
 - 응용시스템에서 소프트웨어 재사용하기 위한 절차는 다음과 같음



고객이나 사용자에게 **현재 시스템**의 여러 부분을 사용해 보도록 한 후 어떠한 기능 등을 원하고 현재의 어떤 것이 좋은지 **조사하여 리스트** 함



고객이나 사용자에게 앞 시스템과는 다른 여러 가지 시스템이나 단위부분들을 사용해 보도록 한 후 **적합한 단위 구성**을 선택



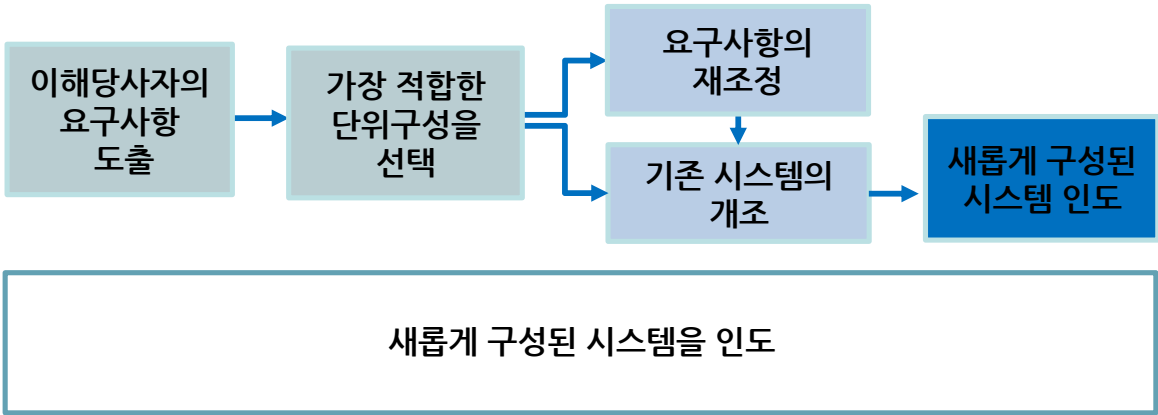
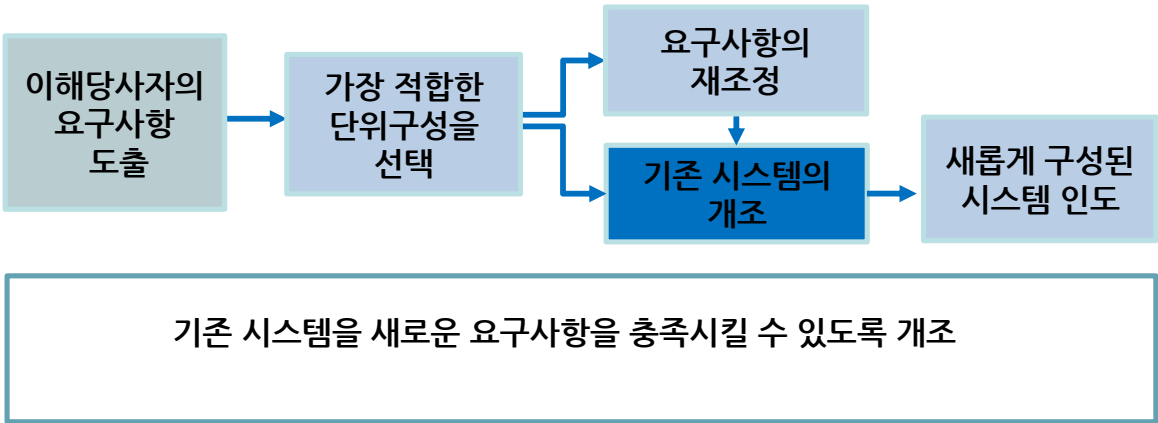
최종 사용자 요구사항을 결정하기 위하여 재조정

❖ 학습내용

[3] 소프트웨어 재사용

2. 소프트웨어 재사용 흐름(계속)

- ◆ 소프트웨어 재사용 절차(계속)
 - 응용시스템에서 소프트웨어 재사용하기 위한 절차는 다음과 같음(계속)



❖ 학습내용

[3] 소프트웨어 재사용

3. 소프트웨어 재사용의 형태

- ◆ 소프트웨어 재사용의 형태 개요
 - 소프트웨어를 재사용하는 경우 다음과 같은 형태가 있음

편의적 재사용(Opportunistic Reuse)

계획적 재사용
(Planned Reuse)

- ◆ 편의적 재사용(Opportunistic Reuse)
 - 프로젝트를 시작할 때 재사용 가능한 컴포넌트가 있는지를 찾아보고 재사용

내부 재사용(Internal Reuse)

- 팀 내에서 만든 컴포넌트를 재사용
- 어디까지나 편의상이며 계획적인 것이 아니기 때문에, 인터페이스의 조정 등에 추가적인 비용이 발생할 수 있음
- 서드파티(외부)에서 만든 컴포넌트를 구하여 사용
- 유상인 경우, 조달비용을 자신이 직접 개발할 때 드는 비용의 20% 이하로 잡는 것이 일반적
- 조달한 컴포넌트를 학습하여 활용하는데 걸리는 시간도 고려해야 함

- ◆ 계획적 재사용(Planned Reuse)
 - 컴포넌트를 차후에 재사용 가능하도록 전략적으로 설계

❖ 핵심정리

1. 신속한 소프트웨어 개발의 개념

- 기업 활동은 시간의 즉시성(Time To Market)을 가진 경우에는 보다 **빠르게 소프트웨어가 개발**되어야 하며 다음 절차로 개발
- **최소단위 프로토타입 먼저 개발, 그 후 필요한 것들을 더하는 방법으로 개발, 사용자가 요구분석, 설계, 구현, 배포 등 전 개발과정에 참여하고 검증함**

2. 애자일 및 익스트림 프로그래밍

- 애자일 개발 프로세스란 어느 특정 개발 방법론을 가리키는 말은 아니고 애자일(Agile=기민한, 좋은 것을 빠르고 낭비 없게 만드는 것) 개발을 가능하게 해 주는 **다양한 방법론 전체**를 일컫는 말
- 개발팀은 요건정의, 분석, 설계 등을 진행할 때, 기존에 익숙한 부분에 있어서 굳이 문서화, 명세화의 시간을 쏟는 것 보다는 **소프트웨어의 구현, 품질 등에 초점**을 맞추어야 함
- 익스트림 프로그래밍은 **애자일 기법 중 하나로** 반복적인 개발을 수행하며, 이 때 나타난 좋은 실무적 관행과 **고객의 참여를 극한수준까지 끌어 올려서 개발하는 방법**

3. 소프트웨어 재사용

- 정보시스템이 **재개발되는 경우나 유사한 업무를 개발**하는 경우 기존 시스템이 재사용되기도 함