1. 소프트웨어와 소프트웨어공학

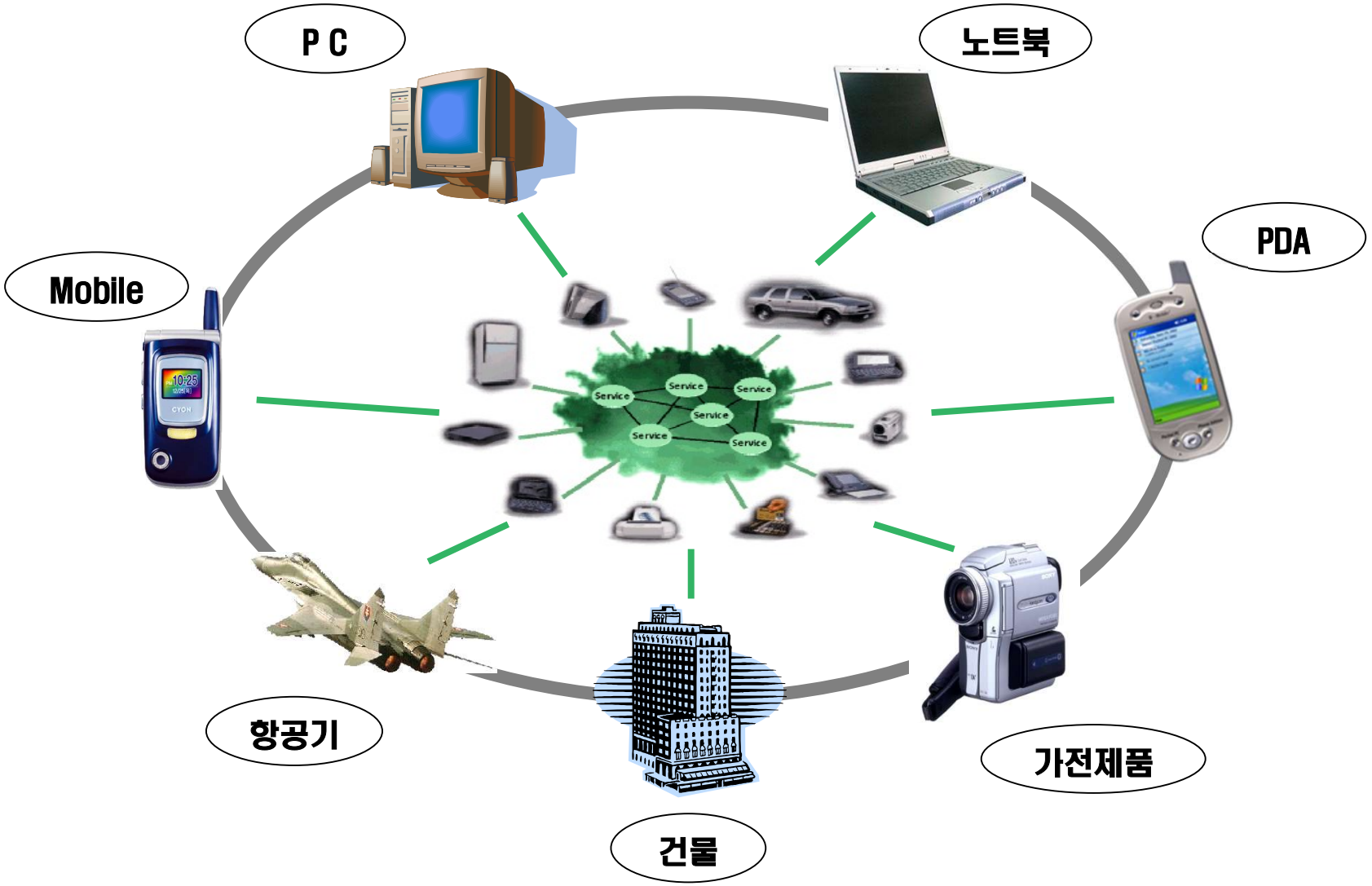
주요 내용

- 소프트웨어의 특징
- 소프트웨어 개발의 특징
- 소프트웨어 공학이란?
- 소프트웨어 공학의 탄생
- 소프트웨어 공학의 필요성
- 소프트웨어 공학의 범위

목차

- 강의 내용
  - 소프트웨어
  - 소프트웨어 개발
  - 소프트웨어 공학
- 팀 프로젝트(다음 주)
  - 1차 팀 구성

모든 곳에 사용되는 소프트웨어



소프트웨어

What is software?

- Computer programs and associated documentation. Software products may be developed for a particular customer or may be deve- loped for a general market.

- What is software engineering?
  - Software engineering is an engineering discipline which is concerned with all aspects of software production

    – by Sommerville

소프트웨어란?

- 프레스만(Pressman)의 정의
  - Software is instruction(computer programs) that when executed provide desired function and performance
  - data structures that enable the programs to adequately manipulate information
  - documents that describe the operation and use of the programs

소프트웨어의 분류 (1/4)

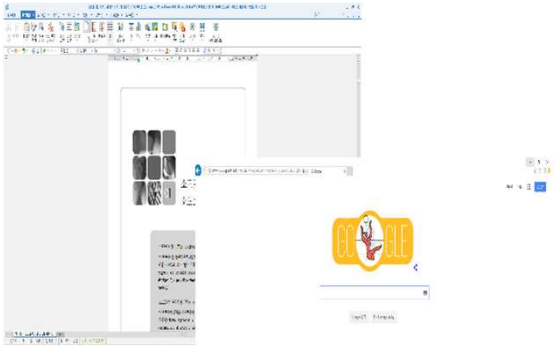
❖ 용도에 따른 분류

– 응용 소프트웨어

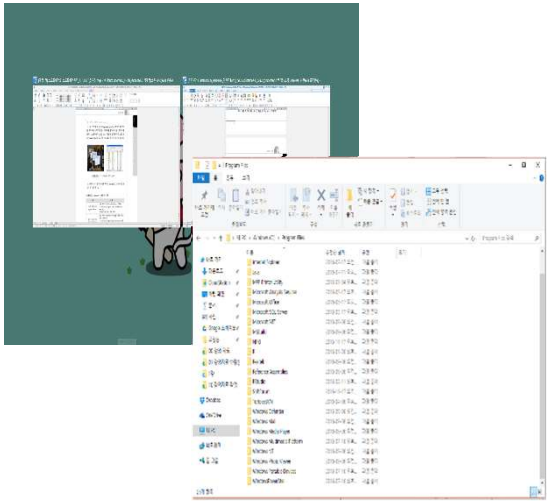
- 사용자의 원하는 목적에 맞게 개발된 소프트웨어
  - ➢ 예) 워드 프로세서(Word Processor), 스프레드 시트(Spread Sheet), 브라우저(Browser) 등

– 시스템 소프트웨어

- 하드웨어를 관리하고 응용 소프트웨어를 지원하는 소프트웨어
  - ➢ 예) 운영 체제(Operating System), 네트워크 관리 프로그램, 파일 관리 프로그램 등



워드프로세서와 브라우저



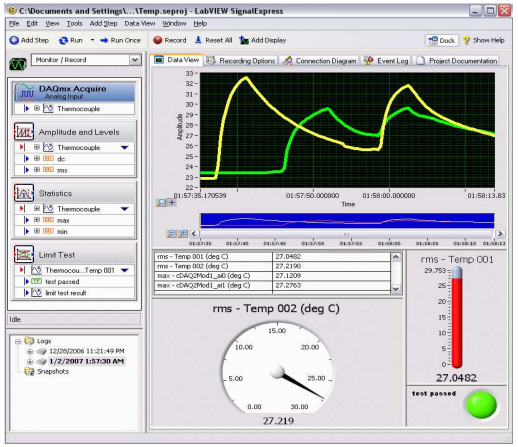
운영체제와 파일 관리 프로그램

소프트웨어의 분류 (2/4)

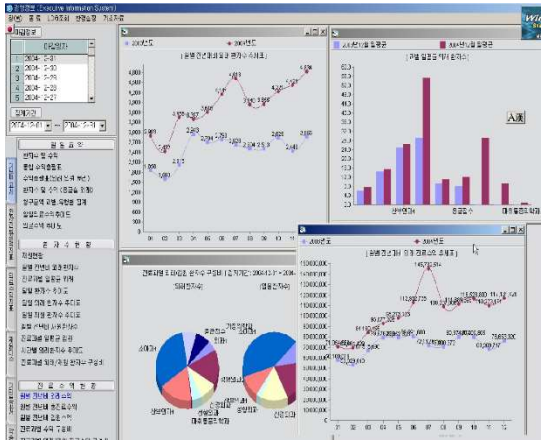
유형	설명
System software (시스템 소프트웨어)	▪ A collection of programs written to service other programs ▪ 예) 컴파일러, 에디터, 파일 관리자
Real-time software (실시간 소프트웨어)	▪ A software that monitors/analyzes/controls real-world events as they occur is called real time ▪ 예) 자동차 엔진 제어 소프트웨어
Business software (비즈니스 소프트웨어)	▪ A software that facilitates business operations or management decision making ▪ 예) MIS(Management Information System, 경영 정보 시스템)
Engineering and scientific software (공학 및 과학용 소프트웨어)	▪ A software that have been characterized by "number crunching" algorithms ▪ 예) 인공위성궤도분석 소프트웨어, CAD(Computer-aided design)
Embedded software (내장형 소프트웨어)	▪ A software that perform very limited and esoteric functions in read-only memory ▪ 예) 전자오븐 제어 소프트웨어, 자동차 브레이크 제어 소프트웨어, PDA 내장 소프트웨어
Personal computer software (개인용 소프트웨어)	▪ A software that support personal purpose ▪ 예) 워드 프로세서, 스프레드시트
Web-based software (웹 기반 소프트웨어)	▪ A software that executes in the web with executable instrunctions(ex, html) and data(ex, text) ▪ 예) 온라인 회원 관리 소프트웨어
Artificial intelligence software (인공지능 소프트웨어)	▪ A software that uses nonnumeric algorithms and straightforward analysis to complex problem ▪ 에) pattern recognition(image and voice) 소프트웨어

출처: Roger S. Pressman, Software Engineering A Practitioner's Approach, 9p

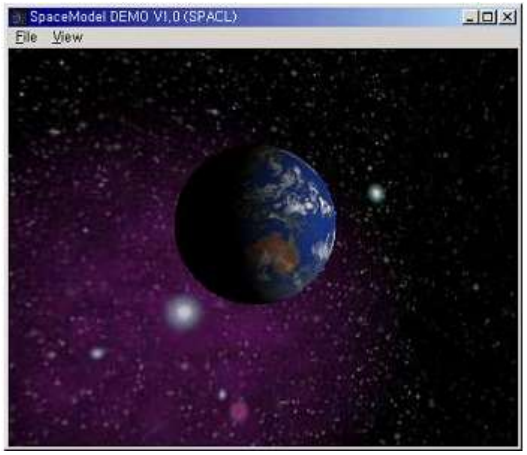
[예] Pressman의 소프트웨어의 분류



자동차 엔진 제어(**LabView SignalExpress**)
(**http://zone.ni.com/devzone/cda/tut/p/id/6318**)



경영 정보 시스템
(**Bit사의 Executive Information System**)



인공위성**3**차원 시뮬레이션
(**Kaist의 SpaceModel**)

온라인 회원 관리
(**Zeroboard**)

소프트웨어의 분류 (3/4)

유형	설명
Stand-alone (독립형)	▪ residing on a single computer ▪ not connected to other software or hardware ▪ 예) 워드 프로세서
Embedded (내장형)	▪ part of unique application involving hardware ▪ 예) 자동차 제어 소프트웨어, PDA 내장 소프트웨어
Real time (실시간형)	▪ must execute functions within small time limit ▪ 예) Radar 관측 소프트웨어
Network (네트워크형)	▪ consist of parts interacting across a network ▪ 예) 웹 기반의 비디오 게임

출처: Eric J. Braude, Software Engineering An Object-Oriented Perspective, 18p

소프트웨어의 분류 (4/4)

- **일반 소프트웨어**
  - 일반적으로 PC 및 대형 시스템상에서 수행되는 소프트웨어를 말함
  - 비기능적인 부분에 대한 고려가 많지 않음
  - 규격화된 하드웨어 및 OS를 대상으로 좋은 범용 개발 환경을 갖추고 있어, 개발자들은 소프트웨어 분야의 지식만으로도 개발이 가능
  - 예) 워드 프로세서, 스프레드 시트, 미디어 플레이어 등
- **임베디드 소프트웨어**
  - 특정 기계 또는 시스템 상 특정 목적만을 위해 수행되는 소프트웨어를 말함
  - 특정 응용을 위해 설계되고, 확장성이 적을 뿐 아니라, 비기능적인 부분을 충분히 고려해야 함
  - 처리 기한이 주어지거나, 외부 영향에 따라 실시간으로 대응해야 함
  - 개발자들은 해당 도메인, 하드웨어 및 소프트웨어 분야의 지식을 충분히 갖추고 있어야 함
  - 예) 전자오븐 제어 소프트웨어, 자동차 브레이크 제어 소프트웨어

소프트웨어의 특징

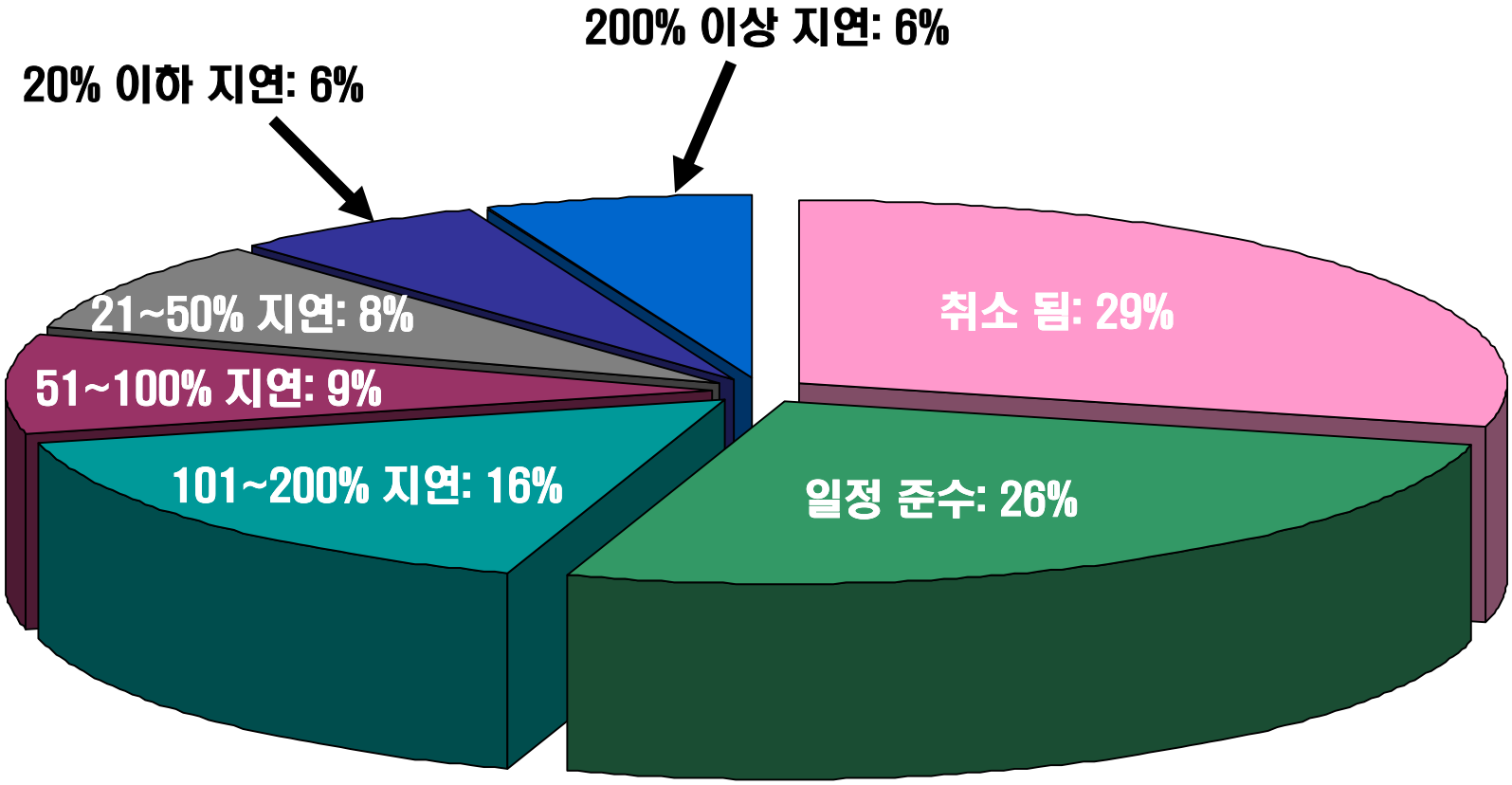
- 소프트웨어의 비가시성(Invisibility)
  - 소프트웨어 완제품의 구조가 개발된 코드 안에 숨어 있어 파악하기 힘든 특징

- 프레스만(Pressman)이 정의한 소프트웨어의 특징
  - 소프트웨어는 고전적인 의미의 '제조(Manufacture)'가 아니라, '개발(Development)'외는 것이다.
  - 소프트웨어는 닳지 않지만, 요구사항의 변경과 주변 환경의 변화에 따라 수정되고 진화한다.

소프트웨어의 특성으로 인한 개발의 어려움

- **소프트웨어는,**
  - **물리적인 형태가 없는 무형의 논리적인 요소**
    - 개발 과정에 대해 정확하게 이해하기 어려움
    - 개발 진행 상황을 파악하기도 어려움
  - **최종 산출물이 개발 과정에서 확인되지 않음**
    - 오류를 발견해야 할 시기를 놓치거나,
    - 오류에 대한 해결책을 못 찾는 경우가 발생
  - **프로젝트의 지연 및 예상 범위 초과로 인한 프로젝트 실패 가능성이 높음**

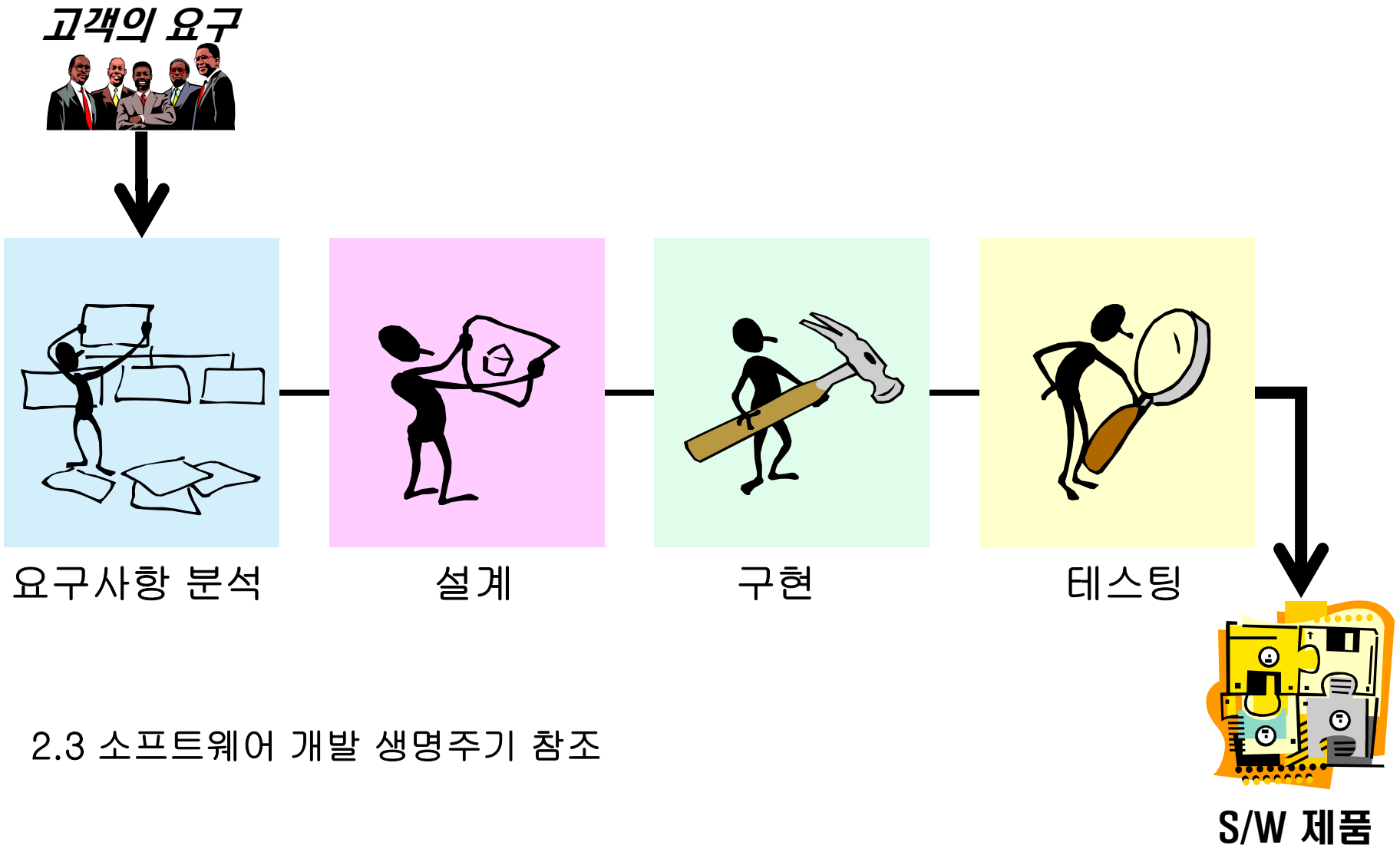
2001년 미국 소프트웨어 프로젝트 결과



출처: **Software Industry Benchmarking Study 2001**

소프트웨어 개발

소프트웨어 개발



2.3 소프트웨어 개발 생명주기 참조

[예] 만년 달력 (1/4)

❖ 1. 고객의 요구

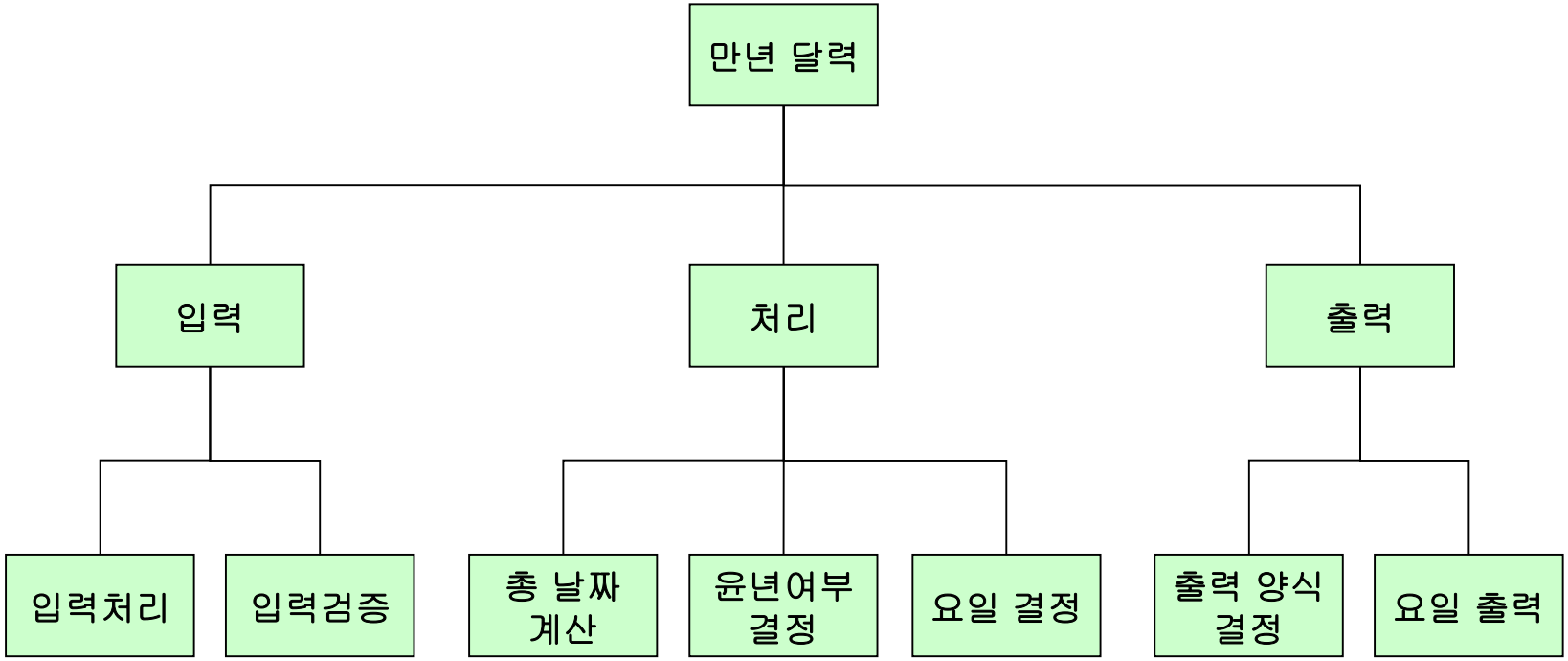
- **"년/월/일을 입력하면 요일을 출력하는 만년 달력 프로그램을 작성해 주시오."**

❖ 2. 요구사항 분석

- **만년 달력의 입력 범위는?**
  - 서기 01년 1월 1일부터 10000년 12월 31일까지로 함
- **입력의 양식은?**
  - 년/월/일을 순서대로 질문하고, 사용자가 응답하게 함
  - 입력 범위를 벗어나면, 다시 입력하게 함
- **출력의 형태는?**
  - 요일

[예] 만년 달력 (2/4)

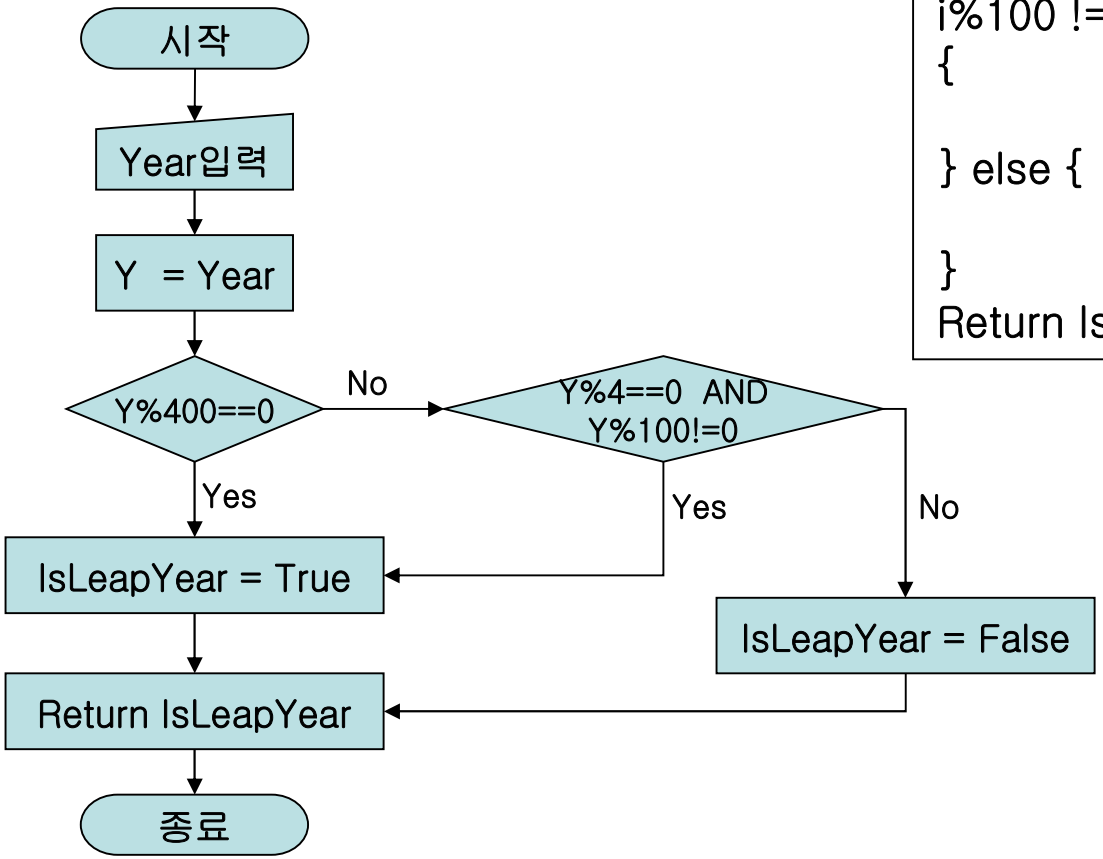
❖ 3. 설계



[예] 만년 달력 (3/4)

❖ 4. 구현(예: 윤년 여부 결정)

- 알고리즘



- 소스코드

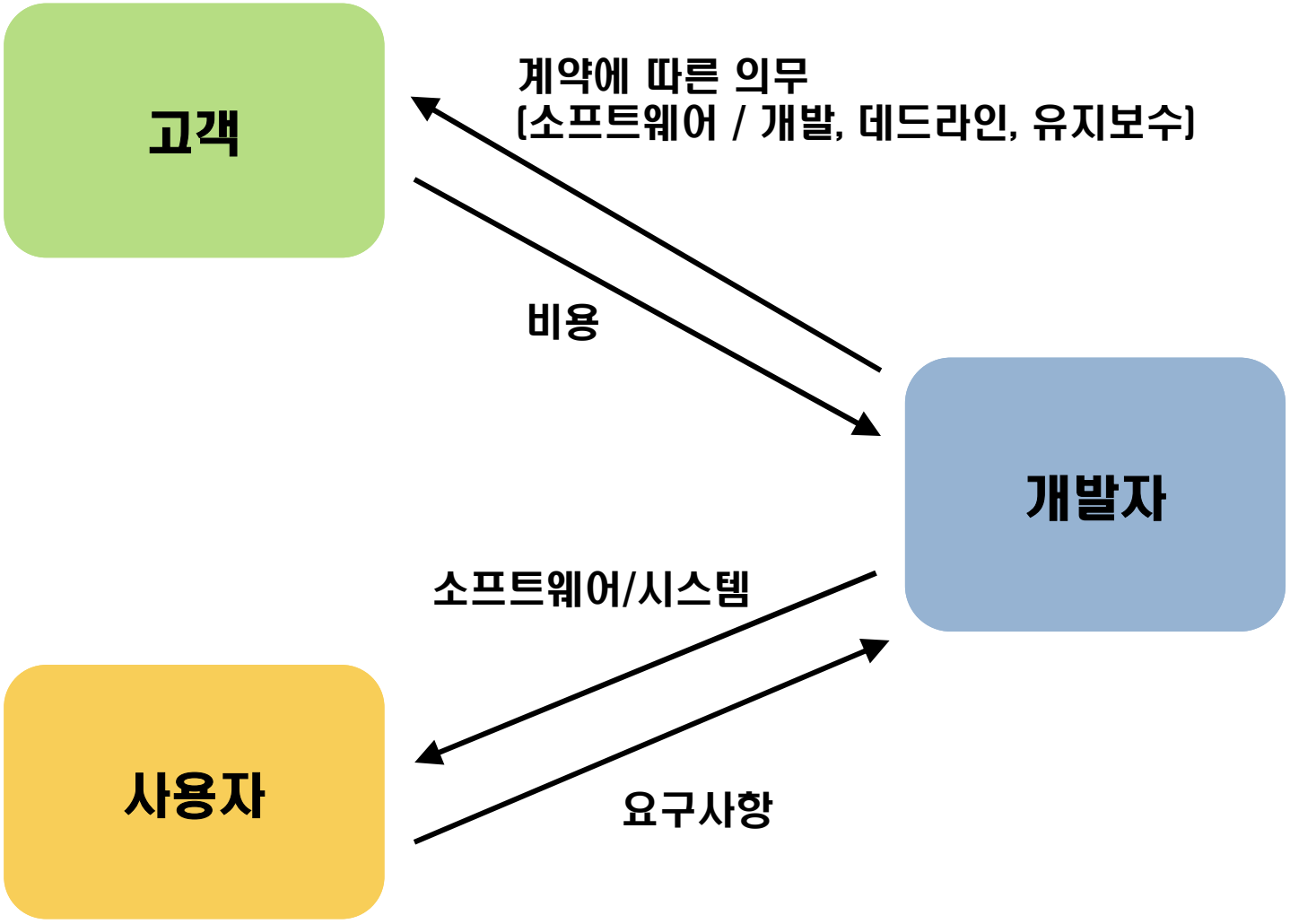
```
if((i % 400 == 0) || (i%4==0 &&
i%100 !=0))
{
        IsLeapYear = True;
} else {
        IsLeapYear = False;
}
Return IsLeapYear;
```

[예] 만년 달력 (4/4)

❖ 5. 테스트

테스트 케이스 ID: ST-0001					
목적	입력에 대한 올바를 출력을 확인한다.				
테스트 조건	1년 1월 1일부터 10000년 12월 31일까지로 한다.				
테스터	한동석	테스트 일자	2016.02.11~2016.02.11		
단계	입력값	예상 출력값	실행 결과	조치사항	조치 결과
1	-10년 입력	오류: 재입력	오류: 재입력	-	-
2	2016년 2월 2일	토요일	토요일	-	-
3	12000년 1월 1일	오류: 재입력	일요일	디버깅 요구	정상 출력

소프트웨어 개발에 연관된 역할들 (1/2)



소프트웨어 개발에 연관된 역할들 (2/2)

- **고객(Customer)**
  - 소프트웨어의 개발 필요성을 결정
  - 사업적 타당성을 판단하여 개발자에게 소프트웨어 시스템 개발 의뢰, 개발비를 제공
- **사용자(User)**
  - 개발자에게 소프트웨어 시스템에 대한 사용자 측면에서의 요구사항을 제공
  - 고객이 사용자의 역할을 같이 할 수도 있음
  - 다양한 사용자가 존재하는 소프트웨어의 경우
    - 사용자의 작업을 이해하고, 요구사항을 이끌어내는 것이 매우 중요함
- **개발자(Developer)**
  - 고객과의 계약대로 주어진 시간 및 비용 내에서 사용자들의 요구사항을 기반으로 소프트웨어 시스템을 개발하는 역할
  - 1명, 또는 팀을 구성하여 작업

개발자에 따라 달라지는 구현 형태

```c
#include <stdio.h>

int PrintMaxA(int value_A, int value_b, int value_c){
    int max, middle, min;

    if(value_a > value_b){
        max = value_a;
        nim = value_b;
    }else{
        max = value_b;
        min = value_a;
    }
    if(max > value_c){
        if(value_c > min){
            middle = value_c;
        }else{
            middle = min;
            min = value_c;
        }
    }else{
        max = value_c;
        middle = value_b;
    }

    return max;
}
```

예제 프로그램 A

```c
#include<stdio.h>

int PrintMaxB(int a, int b, int c){
    int max;

    max = a > b ? a : b;
    max = max > c ? max : c;

    return max;
}
```

예제 프로그램 B

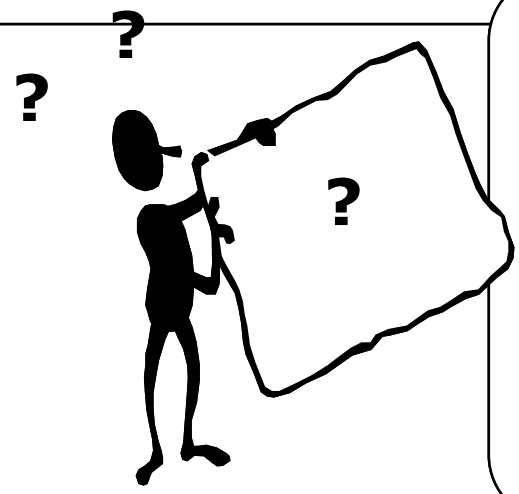
```c
#include<stdio.h>

int PrintMaxC(int a, int b, int c){
    int max;

    if((a > b) && (a > c)){
        max = a;
    }else if((b > a) && (b > c)){
        max = b;
    }else{
        max = c;
    }

    return max;
}
```

예제 프로그램 C

과거의 소프트웨어 개발

소프트웨어 프로그래밍 = 예술

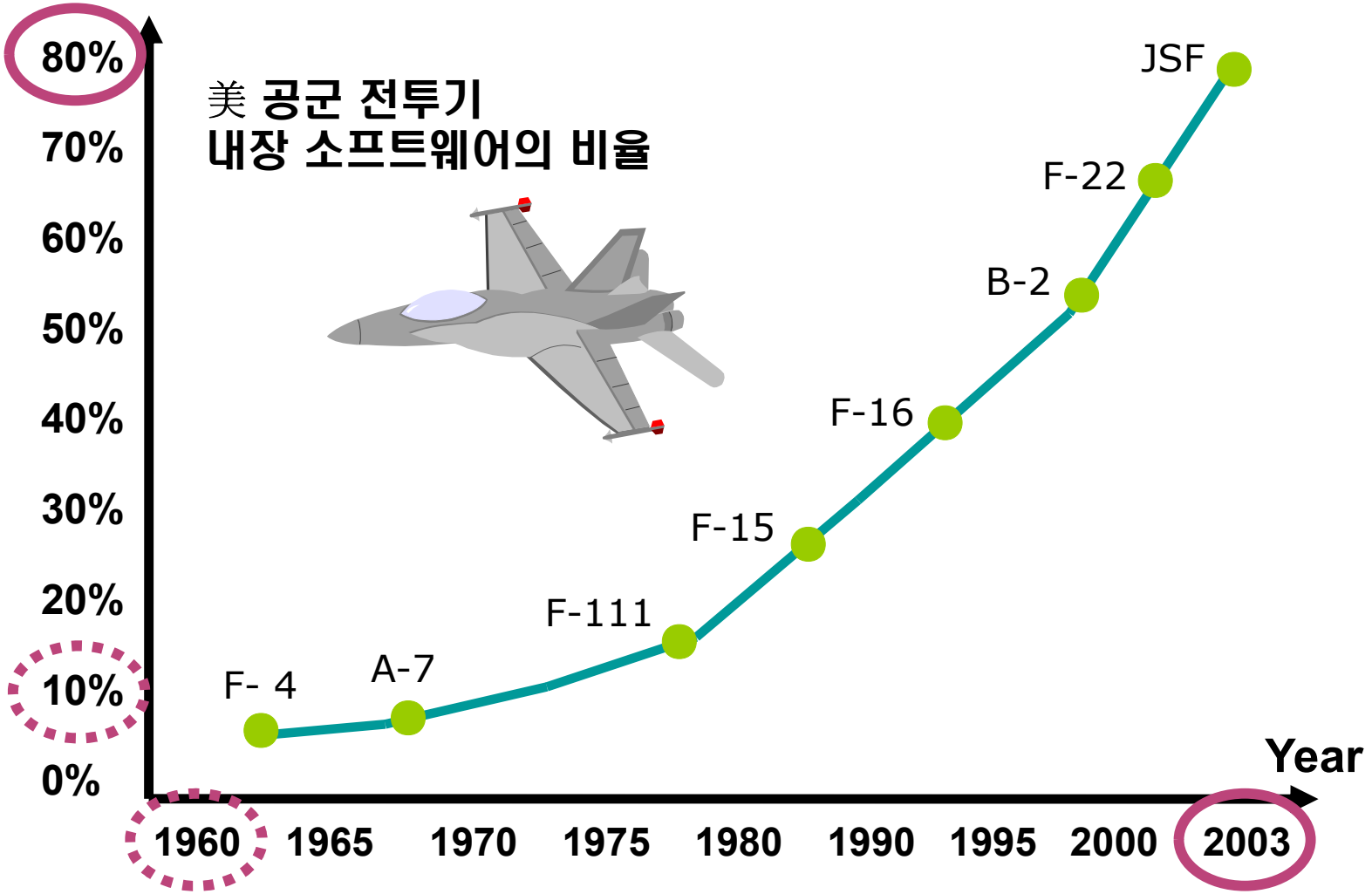
- 개발자에 따라 다양한 방식이 존재
- 사용자 = 프로그래머 = 유지보수 담당자



체계적 방법의 부재

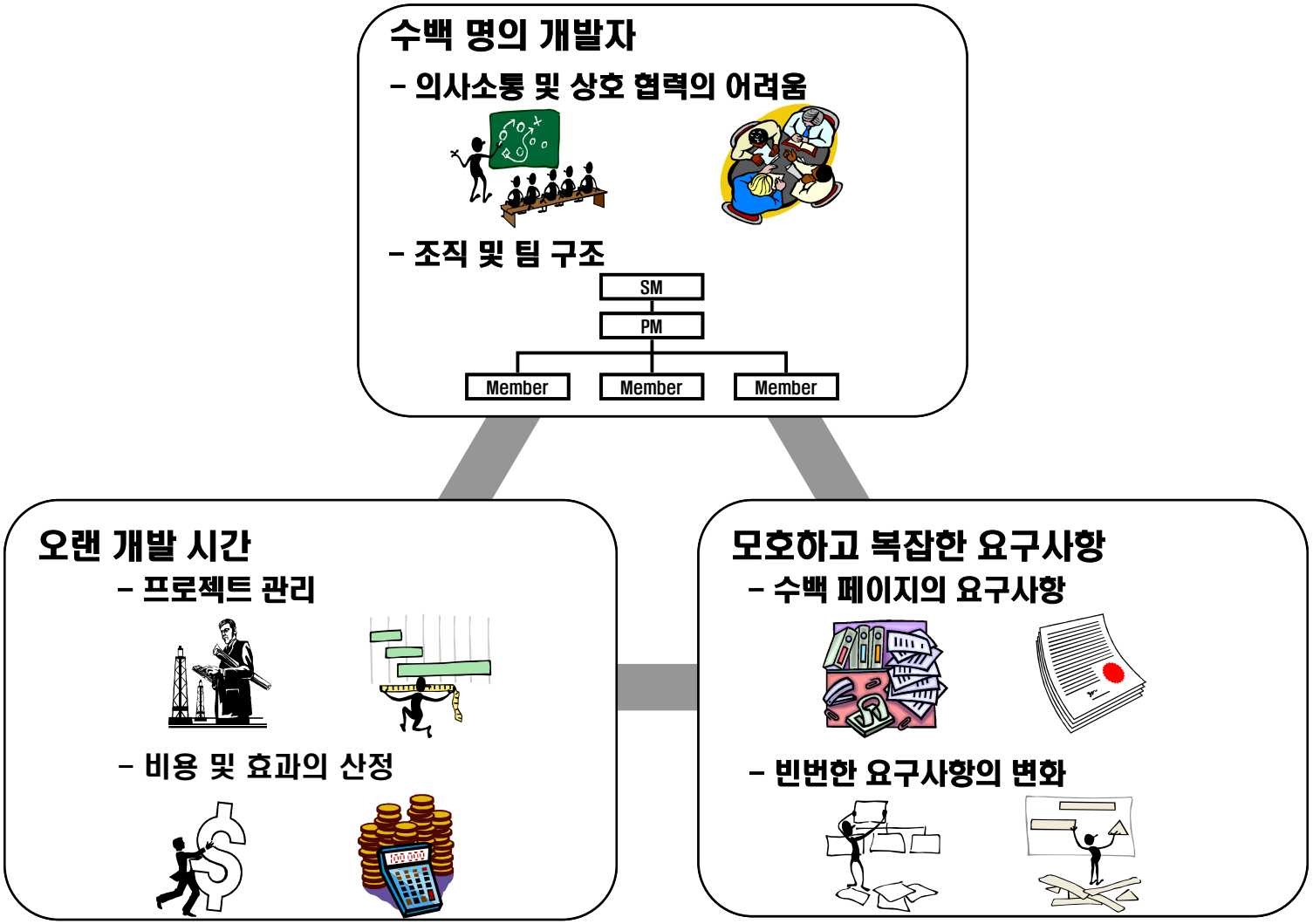
- 정형적인 방법론이 거의 없고, 그것을 사용할 수 있는 사람도 거의 없음
- 프로그래머는 시행착오에 의해 기술을 습득 함

점점 더 중요해지는 소프트웨어



Source: U.S Airforce

대규모 프로젝트의 어려움



소프트웨어 공학의 대두 배경

- 소프트웨어 위기(Software crisis)
  - 소프트웨어 수요 증가에 비해 공급 및 개발의 어려움

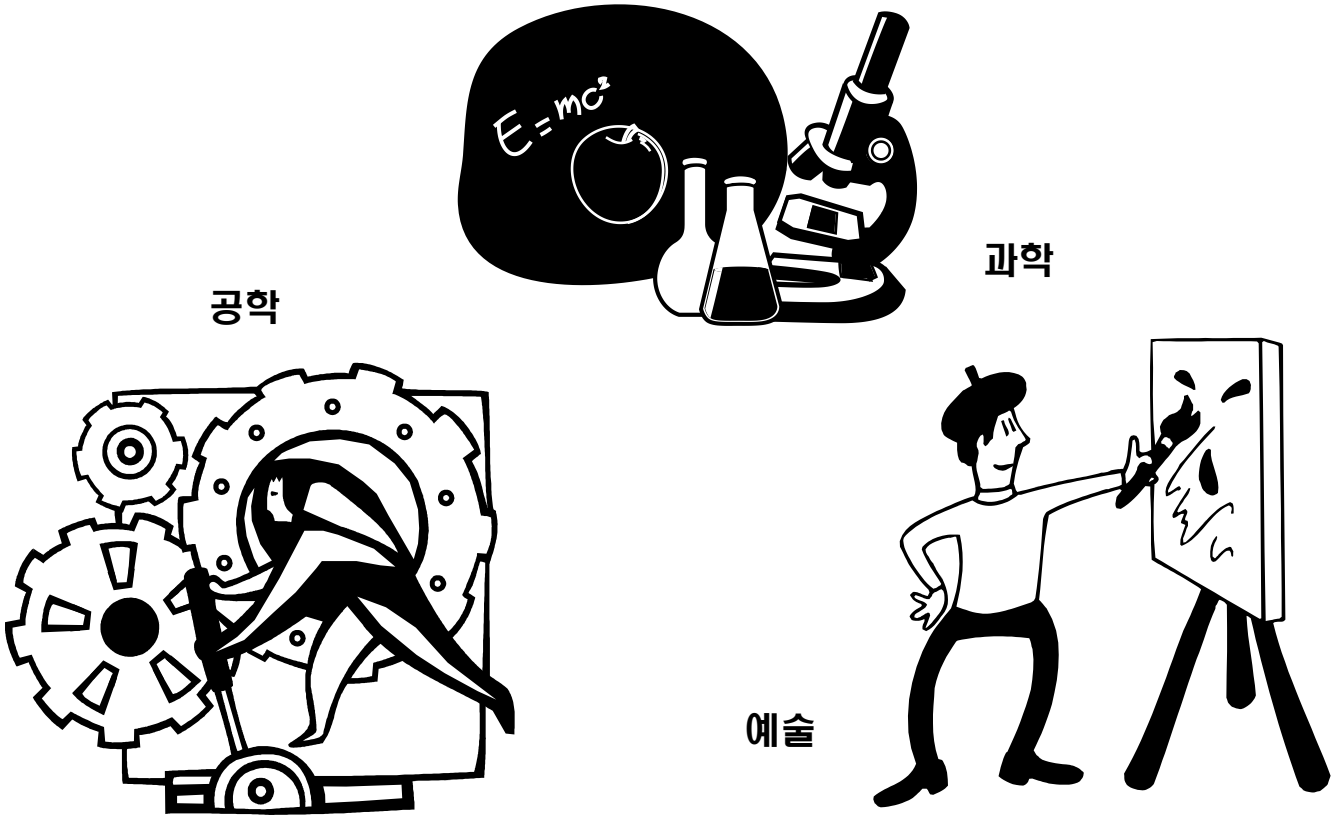
- 소프트웨어 위기의 해결
  - 다른 분야에서 사용했던 공학(Engineering) 패러다임을 이용하자는 결론
  - 1968년 NATO conference에서 소프트웨어 공학(Software Engineering) 제안됨

소프트웨어 공학

소프트웨어 공학이란?

- 정의
  - 소프트웨어의 개발, 운용, 유지보수 및 폐기에 대한 체계적인 접근 방법
- 특징
  - 소프트웨어 개발 전 과정에 걸쳐 필요한 이론, 개념 및 기술을 다룸
  - 소프트웨어 개발 과정에서 생성되는 모든 산출물이 그 대상이 됨
- 목표
  - 소프트웨어 개발이 체계적이고 공학적인 방법으로 이루어져 추정된 비용과 기간에 고객이 원하는 품질 높은 소프트웨어를 개발하는 것

과학, 공학, 예술의 차이



공학이란?

❖ 의미

- 실제적 문제(Practical Problem)를 해결하거나
- 실제적인 산출물을 생산해내기 위해
- 자원과 비용을 효과적으로 활용하면서
- 과학적 지식을 적용하는 것

❖ 공학과 소프트웨어 공학

- **공학**
  - 업무분야에서 문제 발생 시, 실무자가 적절한 해답을 찾을 수 있도록 체계적으로 정리된 기술적 지식을 제공
- **소프트웨어 공학**
  - 소프트웨어 개발 기술, 절차 및 도구의 우수한 사례(Best Practice)들을 정리하여 소프트웨어 개발 시, 누구나 당면한 문제를 해결할 수 있도록 체계적인 기술적 지식을 제공

소프트웨어 공학의 주요 영역들

소프트웨어 공학의 영역들(1/4)

영역	주요 내용
요구공학 (Requirement Engineering)	▪ 소프트웨어 개발에서 수행되는 첫 번째 작업 ▪ 개발될 시스템에 대한 고객의 요구를 이해하고 목표와 제약사항을 확립하여 시스템을 만족시킬 기능, 성능 그리고 다른 시스템과의 인터페이스 등을 정의하는 과정 ▪ 비용 증가, 납기 지연, 품질 저하를 방지하기 위한 필수 요건 ▪ 요구사항의 추출, 저장, 변경 프로세스 및 요구사항 관리 지원 도구 등이 연구되고 있음
아키텍처 (Architecture)	▪ 아키텍처 구성 요소와 이 구성 요소들 간의 관계, 그리고 시스템의 기능, 속성 및 제약사항 등을 적절히 반영하는 구조가 서로 조직화되어 목표 시스템의 전체적인 형태를 표현 ▪ 적절히 반영하는 구조란 기존의 아키텍처 스타일을 문제 영역에 적절하게 변형 또는 조합하고 해당 스타일에서 언급하는 컴포넌트(Component)와 커넥터(Connector)로 시스템을 분할하여 구조화 하는 것 ▪ 아키텍처의 유형 분류, 아키텍처의 정의 언어, 아키텍처 분석 방법론 등이 연구되고 있음`
개발 방법론 (Development Methodology)	▪ 시스템을 개발하기 위해 어떠한 방법으로 진행할 것인가를 다루는 분야 ▪ 구조적 방법론, 객체지향 방법론, 컴포넌트 방법론 등 ▪ 개발 기술의 진화에 따라 계속적으로 연구, 발전되고 있음 ▪ 개발 조직의 특성 및 여건에 맞게 조정/ 재정의 될 수 있음

소프트웨어 공학의 영역들(2/4)

영역	주요 내용
테스팅 (Testing)	▪ 단위 테스팅, 통합 테스팅, 시스템 테스팅 등 ▪ 효과적인 테스트 케이스 산출 방법론, 각 개발 방법론 및 분산 환경에서의 다양한 테스트 방법이 연구되고 있음
프로세스 (Process)	▪ 소프트웨어의 개발 및 진화에 사용되는 활동, 방법 및 실무 활동(practice) 들의 집합 ▪ 최종 소프트웨어 제품을 생산하기 위하여 요구되는 인력, 절차, 방법, 장치 및 도구 들을 통합하는 수단 ▪ 프로세스 정의 방법, 프로세스 관리 조직 및 관리 기반 구조 등에 대해 연구되고 있음 ▪ 소프트웨어 프로세스의 특성을 설명하는 모형 및 효과적인 소프트웨어 프로세스 실현을 위한 단계적 접근 방법을 명시하는 모델에 관해 연구되고 있음
형상 관리 (Configuration Management)	▪ 소프트웨어 구성 요소에 대한 변경 관리 대상인 형상 항목을 식별하고 변경을 통제, 기록함 ▪ 형상 식별, 형상 통제, 형상 상태 확인, 형상 감사 등의 활동이 있음

소프트웨어 공학의 영역들(3/4)

영역	주요 내용
품질 (Quality)	▪ 소프트웨어 분야에서 품질은 제품 품질(Product quality)과 프로세스 품질(Process quality)로 분류됨 ▪ 제품 품질은 제품 자체가 가지는 품질을 의미하며, 프로세스 품질은 소프트웨어를 개발하는 프로세스가 정확하고 우수하면 좋은 품질의 소프트웨어를 생산할 가능성이 높다는 것을 의미 ▪ SQA(Software Quality Assurance) 활동, 제품 검사, 검토 등을 지원하는 평가 모델, 국제 표준 등이 연구되고 있음
재사용 (Reuse)	▪ 코드 뿐만 아니라 응용 분야에 관한 지식, 개발 경험, 설계에 관한 결정, 시스템에 대한 지식, 요구 분석 사항, 설계, 문서 등의 재사용 ▪ 코드 재사용의 한계를 극복하기 위해 코딩 단계 이전의 분석 및 설계 단계에서 만들어진 산출물을 재사용하려는 노력이 계속되고 있음
프로젝트 관리 (Project Management)	▪ 프로젝트의 일정 및 인력, 예산 등을 관리하여 프로젝트를 성공적으로 이끌기 위해 요구되는 영역 ▪ 프로젝트 통합, 범위, 일정, 비용, 품질, 인적자원, 의사소통, 위험, 조달관리의 9가지 영역으로 분류함[PMBOK(Project Management Body of Knowledge)]

소프트웨어 공학의 영역들(4/4)

영역	주요 내용
정형기법 (Formal Method)	▪ 수학과 논리학을 기반으로 하드웨어나 소프트웨어 시스템을 명세하거나 검증하는 기법 ▪ 정형논리(Formal Logic) 또는 수리 논리(Mathematical Logic)등을 이용하여 시스템이 동작할 환경, 시스템이 만족해야 할 요구사항, 요구사항을 수행할 시스템 설계 등을 기술하는 정형 명세 기법과 수식이나 기호를 사용하여 시스템을 정형화함으로써 보다 체계적으로 시스템의 전체 범위를 검증 할 수 있는 정형 검증 기법으로 분류
유지보수 (Maintenance)	▪ 소프트웨어가 고객에게 인도된 후에 폐기될 때까지 개선을 목적으로 소프트웨어 부분 또는 전반적으로 수정, 보완하는 일 ▪ 잘못된 것을 수정하는 목적으로 수행되는 수정 유지보수(corrective maintenance), 시스템을 새로운 환경에 적응시키는 목적으로 수행되는 적응 유지보수(adaptive maintenance), 새로운 기능을 추가하거나 시스템의 구조와 성능을 개선하여 소프트웨어 완전하게 만드는 목적으로 수행하는 완전 유지보수(perfective maintenance), 소프트웨어의 잠재적인 결함을 사전에 예방할 목적으로 수행하는 예방 유지보수(preventive maintenance)로 분류

연습문제

1. 소프트웨어가 가지고 있는 특성에 대해 설명하라.

2. 소프트웨어 공학이란 무엇인가?

3. 소프트웨어 공학이 나타나게 된 배경은 무엇인가?

4. 소프트웨어 위기를 설명하라.

5. 소프트웨어 공학의 분야들을 나열하라.

6. 소프트웨어와 관련된 고객, 사용자, 개발자의 역할에 대하여 설명하라.

팀 프로젝트

이번 주 할일

- 프로젝트를 진행하기 위한 팀을 구성합니다.

- 제출 내용
  - 팀명
  - 팀원, 팀장(각 이름, 학번)

다음 주 제출 문서

- 팀 구성을 확정합니다.