Software Engineering

Software's chronic Crisis

200710116 박석희
200710117 양다빈
200711475 허원선
200711476 홍창현

Do you know Denver airport?

- 미국 콜로라도주(州) 덴버 근교에 있는 국제공항.
- 공항 면적은 1억 3758만 8140㎡이며, 이는 맨하탄의 두 배 크기에 해당.
- 스키장비용 컨베이어벨트를 설치하여 승객들에게 편의성을 제공.

Primary concern of this topic

Baggage-handling system

- Danver airport 지하에 있는 자동 수하물 처리 시스템.
- 21마일 길이의 철길을 따라 4000여개의 telecar들이 움직임.
- 카운터, 게이트, 20여개 타 항공사 claim area사이에서 짐을 배달.
- 여러 하드웨어가 중앙 통제 시스템으로 연결.

Unexpected error in this system

Failure to find their destination

- 4000여개의 telecar들이 목적지를 잃어버리고 방황.
- 방황하는 telecar들간의 접촉사고.

Failure to expect telecar's ability

- 빈 cart는 계속 짐을 싣지 않고, 꽉 찬 cart들은 짐을 더 싣음.
- Cart에 너무많은 짐을 싣어 결국 cart가 지정된 길을 이탈.

Result of this error

Waste of much money

- 이 에러로 인하여 11개월동안 하루당 백만달러 이상의 적자를 냄.
- 결국 타 시스템을 5100만달러에 구매함.

Lose trust of consumer

- 항공관계자들에 의해 덴버공항의 등급이 최저로 변경됨.
- 사람들의 인식이 부정적으로 바뀜.

What Denver debacle means

Need of software process

- Nato 과학위원회에서 software roadmap 필요성의 제기.
- Software engineering의 탄생.

Present situation of software

NATO conference in 1968

- NATO회의에서 software roadmap의 필요성이 제기됨.
- 실제 software roadmap을 만드는 데에 실패함.

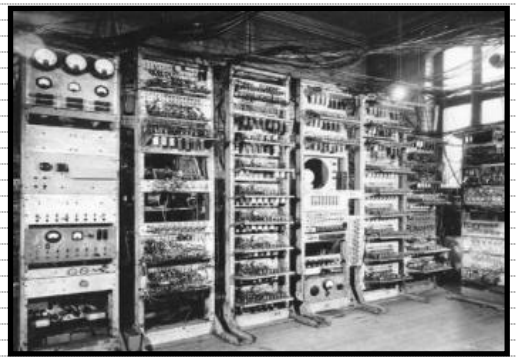
At present

- 예견된 software의 거대한 변화에 따른 적절한 대응책이 없음.
- 현재 종잡을 수 없는 상황.

In preparation for changes

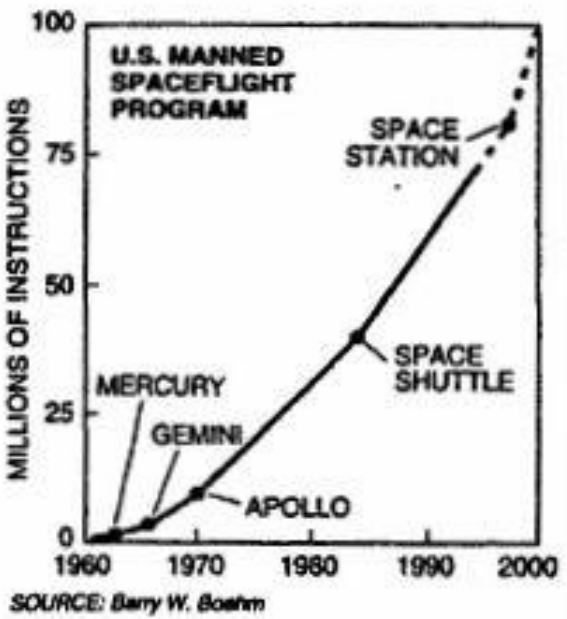
Hardware's change

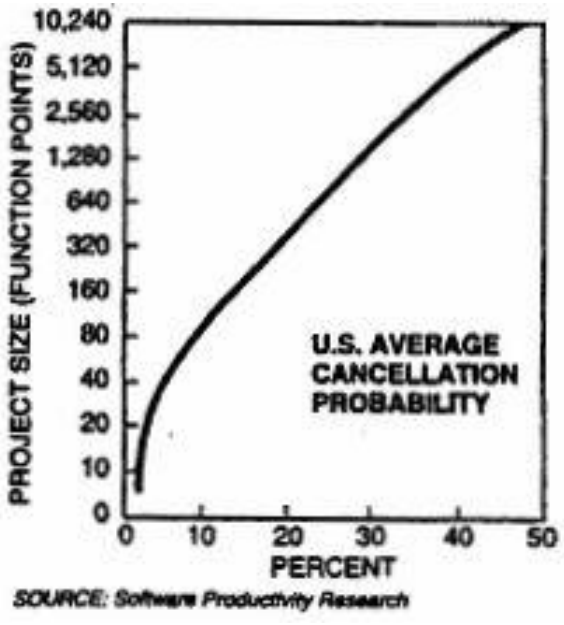
- 현재 hardware들은 좀더 빠르고, 좀더 작고, 좀더 값싸게 변화하고 있음.

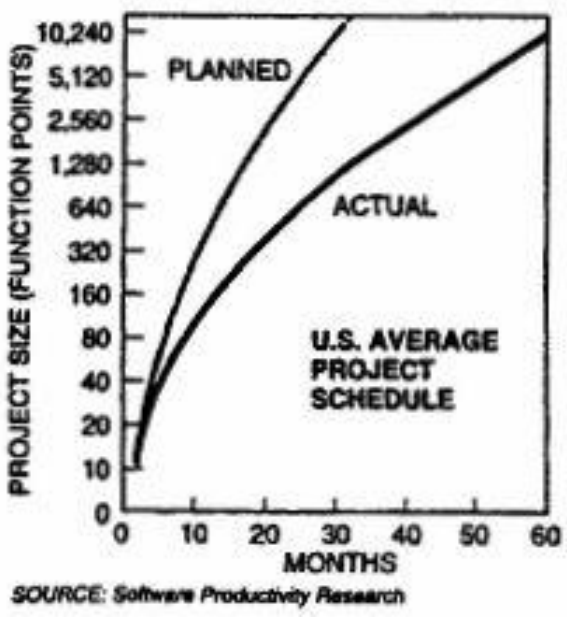





Software is exploding





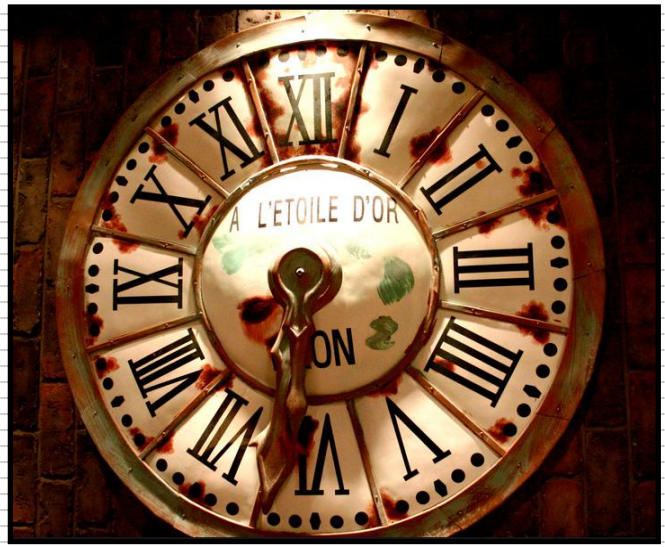


- 전통적 프로그래밍 습관은 프로젝트 규모의 큰 변화에 적응하지 못하고 실패를 거듭하고 있음.

Fundamental assumption

About traditional practices

- software를 개발하면서 당연히 문제점이 있을거라는 것을 인정하는 것.
- 이런 습관은 Hardware의 변화에 대응하여 바뀌어야함.



Do you know Clementine?

- 미국 국방부와 NASA가 합작하여 1994년 1월에 쏘아올린 위성.
- 주요 임무는 달의 탐사 및 Targetting system의 실험.
- Targetting system을 위해 국방부는 업격한 기준을 제시.

Failure of Clementine

Failure of Clementine

- 달궤도를 돌다가 달에 착륙하려했지만 엔진에 11분간 불이 붙음.
- 결국 달에 착륙하지 못함.

Cause of failure

- Climentine 프로그램에 버그가 발생하여 엔진에 불이 붙음.
- 엄격한 기준에도 불구하고 이와같은 실시간 에러는 예측하기가 힘듬.

Distributed system

Definiton

- 많은 네트워크 컴퓨터에서 협력적으로 실행되는 프로그램.
- 성능의 향상, 신뢰성의 확보, 시스템의 확장 용이성을 가짐.

Demand about distributed system

- 이 시스템에 대한 기업들의 수요가 점차적으로 증가되는 추세.
- 장점에 매혹돼 전략적인 무기로 사용하려는 기업이 늘고 있음.

Dark side of distributed system

Difficulty in implementation

- 대부분의 프로그램들은 실제로 올바른 종류의 데이터가 통과하는 다루기 힘든 로직으로 복잡하게 얽힌 망임.
- 각각의 시스템에 기능은 비슷하지만 디자인은 다른 함수들이 존재.

Meaning of “graunch”

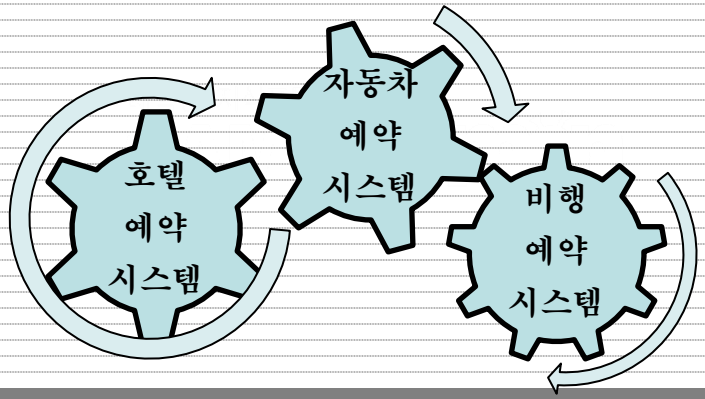
- ‘과도한 힘의 사용에 의해 맞춰지게하다.’ 라는 뜻으로 사용됨.
- 시스템 병합의 어려움을 나타냄.

Examples of graunching

California's Department of Motor Vehicles

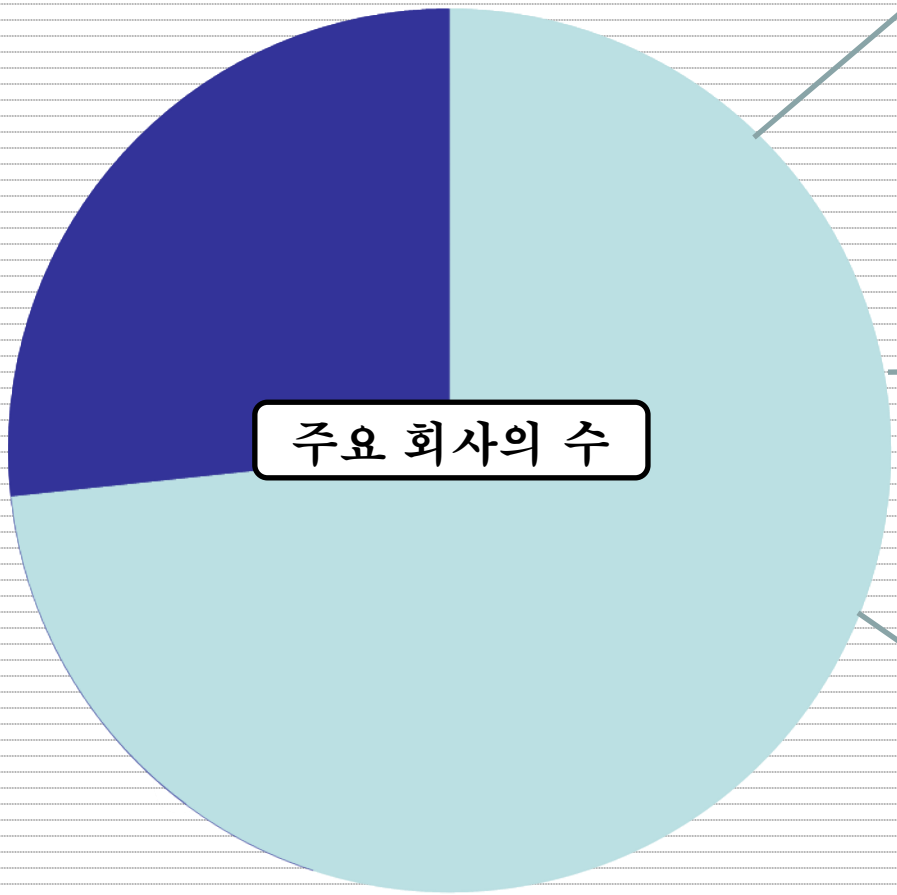
- 주의 운전기사와 차량 등록 시스템을 합병하려 함.
- 예상한 비용보다 6.5배 더 사용했으며, 결국 7년 만에 중단됨.

American airline



- 세 시스템의 융합을 꾀함.
- 엄청난 빚과 수많은 소송 사이에서 중단됨.

Research about Distributed system



55% cost more than expected

63% overran their schedules

88% substantially redesigned.

The challenge of complexity

Cause of complexity

- 예기치 않은 오류의 발생을 예상하지 못한 것.
- 네트워크가 위 문제를 증폭시킴.

What we have to do

- 프로그래머의 일하는 자세를 바꿔야 됨.
- “당신은 목수를 이용해 고층건물을 지을 수 없습니다.”

Advanced Automation System

Background of appear

- 공중 교통 통제 시스템을 교체하려는 시도에서 제안됨.

About AAS

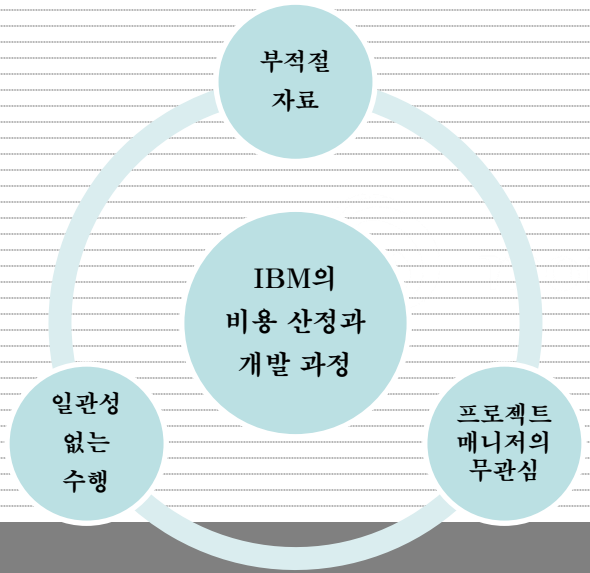
- 그 당시 모든 computing의 모든 기술이 집약됨.
- 예측할 수 없는 실시간 사건에 24시간내내 대응해야함.
- 오류발생의 작은 가능성은 공공의 안전을 위협함.

Failure of the project

FAA's expectation

- 프로젝트의 비용과 개발기간을 산정하기 위해 최첨단기술의 사용.
- 요구사항과 에러검출이 용이한 디자인을 적용할 것 이라고 예상.

What came to light



As a result

Damage to FAA

- 개발된 코드의 대부분을 다시 작성해야 함.
- ASS 소프트웨어에 라인당 700~900달러 지불.

Confrontation of FAA

- 6개의 시스템 중 2개의 ASS의 주요 부분을 취소.
- 남은 4개의 시스템 중 3번째 것을 축소.

About CMM's mean

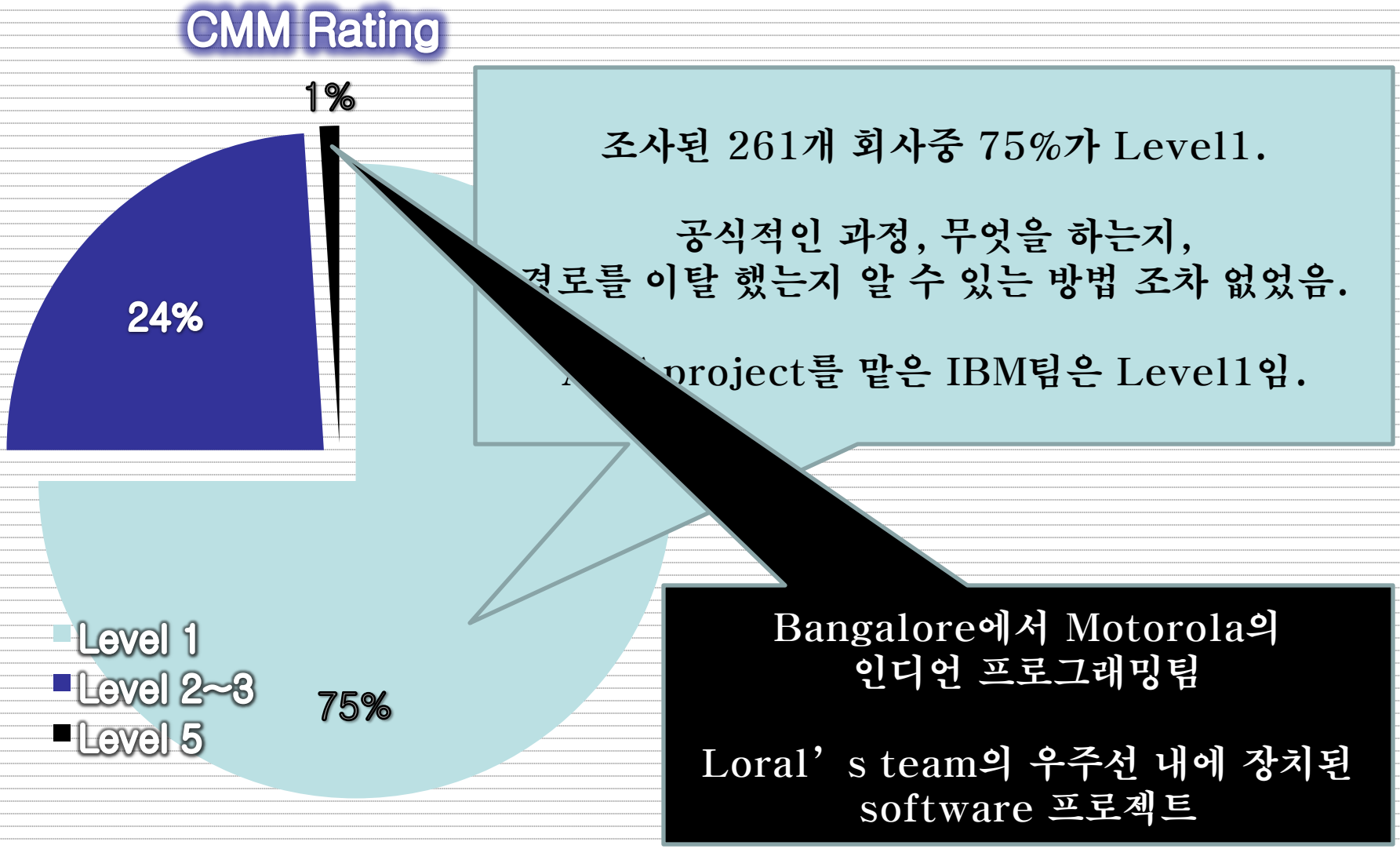
Basic fact of CMM

- Capability Maturity Model은 Think-tank에서 개발됨.
- Software engineering과 완벽한 관리의 비전을 제시함.

Effect of CMM

- 기존의 막무가내식 개발과정을 좀 더 일관되게 함.
- CMM을 이용한 소비자의 요구에 얼마나 부합하나에 따른 레벨의 구분.

Level system using CMM



Why Loral's team is ranked in L5

Situation at the time

- 미국 프로그래머의 90%가 그들이 발견한 오류의 개수를 세지 않음.
- 10%에서도 결점의 1/3이상을 잡아내는 사람이 몇 없었음.

Special about Loral's team

- Loral's team은 버그를 다루는 방법을 잘 배웠음.
- 버그를 잡는 것을 넘어 취약점이 있는 test과정의 결함을 고치려고 노력.

Unavoidable bugs

Example 1

- 1981년 우주왕복선의 첫 발사가 2일 연기되었다.
- 결함은 5개 컴퓨터의 동기화를 막았다.

Example 2

- 왕복선 집결 프로그램은 1992년 인텔셋 통신위성구조미션을 위태롭게 함.

Persuasion of SE Institute

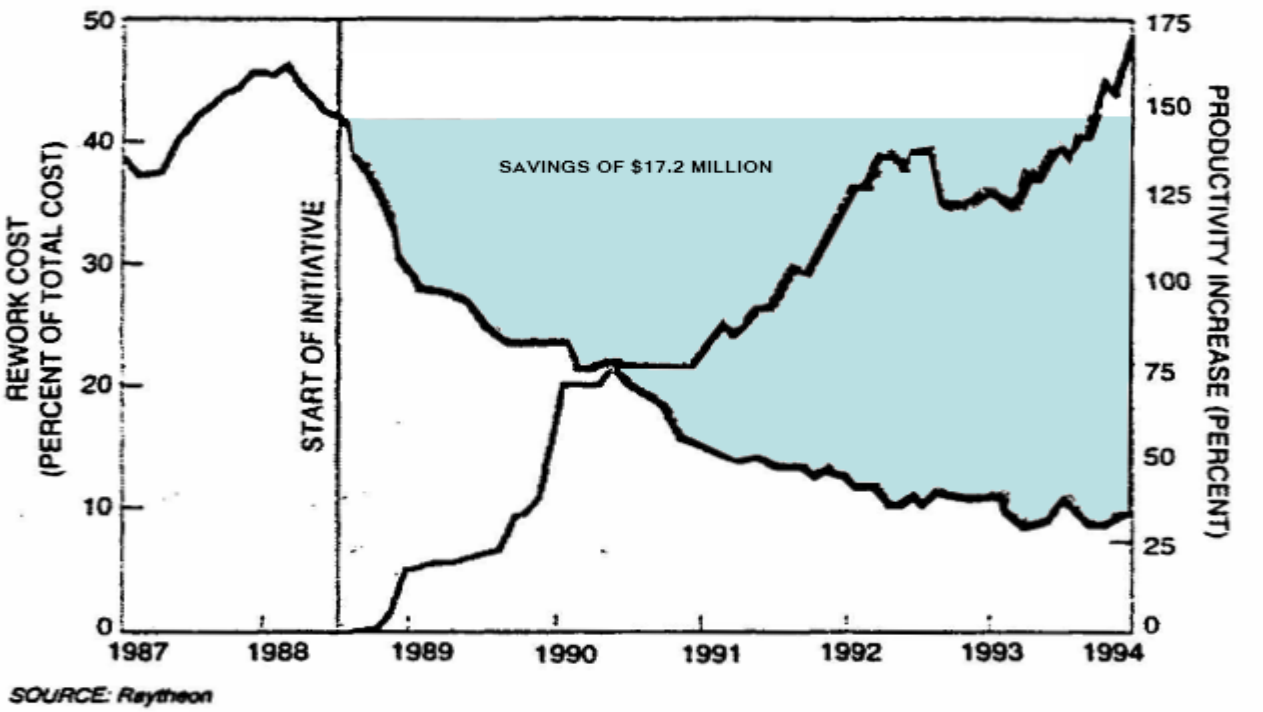
Persuasion

- SE 연구소는 양적 품질관리는 언젠가는 망할 것이라고 설득함.

Example of Innovation

- 1988년, Raytheon의 장비과는 CMM test에서 낙제 후 SE계획을 구상.
- 엄격한 기준과 테스트 가이드라인을 만들고 다듬음.
- 400명의 프로그래머를 가르치는 데 일년에 백만 달러를 투자함.

Success of Raytheon's equipment division



- 시간이 지남에 따라 재개발비용이 줄어듬.
- 시간이 지남에 따라 생산성이 높아짐.

Critical error

Limitaion

- 잘 짜여진 디자인에서도 에러는 발견될 수 있음.
- 사람이 프로그램을 만드는 한은 에러는 생김.

Characteristics of Bugs

- 버그가 초창기에 잡힐 경우는 프로젝트의 기한과 예산에 큰 영향이 없음.
- 버그가 나중에 발견될 경우 치명적으로 작용할 수 있음.

Error detection method in MS

Beginning

- 대형 소프트웨어 회사들은 버그를 고치는 데 소극적이었음.

Example of M.S

- 새로운 버전의 윈도우즈가 나오면 2만명의 지원자에게 베타테스트를 맡김.
- 이 방법은 효과적이지만, 비싸고 비효율적이고 비실용적.

Growing software

Overlooking

- 초기에는 소프트웨어가 한번 계획되면 아무 문제 없을 줄 암.
- 현실은 소프트웨어는 개발되면서 문제가 생김.

A First step for solution

- 프로그래머들은 그래픽 인터페이스를 사용하여 그들의 프로토타입을 만듬.
- 시스템 프로토타입은 논리적 기반이 무너지기 전에 고객과 개발자 사이의 오해를 해소하는데 도움을 줌.

Problem of A First step

One

- 프로토타입은 디자인 중심이므로 논리적인 모순에는 도움이 안됨.

Two

- 관심을 둬야하는 오류는 누락의 오류임.

Three

- 디자인에 따라 코드가 써지면 버그를 감지하는 것은 어려움.

The origin of Formal Method

Necessity

- 디자인이 실제 어떻게 사용될 지 알기 위해 수학적 분석이 필요했음.
- 물리적 수학으로는 컴퓨터의 이진수를 해결할 수 없어 이산수학을 이용함.

Successful case

- Praxis’ s Project.
- GEC Alsthom.

Praxis's Project

Explanation of Project

- 중복된 시스템 동기화의 유지.
- 동기화가 두개의 네트워크를 통해 적절한 순서로 이루어지는 것.

A Happy failure

- Formal method를 사용하여 초반에 시스템의 오류를 발견하여 재시작함.
- 다시 시작했음에도 예정한 시간을 초과하지 않음.

A wise choice of GEC Alsthom

Their choice : “B”

- GEC는 6천대의 기차의 길을 인도하는 시스템과 속도를 조절하는 시스템에 Formal method “B” 를 이용하여 총 10번의 업그레이드를 행함.
- 3억 5천만 달러가 사용됨.

Effect of Formal Method “B”

- 기차의 속력이 증가하고 길인도가 빠르게 이루어짐.
- “B” 를 사용하지 않았을 경우 10억달러의 비용이 요구되었음.

Limitations of Formal method

Limitations of Formal method

- 프로그래머들은 종종 증명과정에서 실수.
- 설계적인 측면은 보증하지만 실제 구동은 보증하지 못함.
- 수학 공식을 읽는 것은 코드를 다시 보는 것보다 더 어려운 일임.

난 미국인들이 Formal Method를 다양한 범위에 적용하는 충분한 교육을 받았다는 것에 의문이 든다

– David A. Fisher

What is the Clean Room Process?

Exception

- David A. Fisher의 말에 대한 예외로 clean-room식 접근 방법을 사용하는 회사가 많아진 사례가 있음.

Clean room process

- Formal Method 융합을 시도.
- 한번에 완벽하게 작동할 수 있게 엄격한 공학 기술을 사용.
- 프로그래머는 한번에 하나의 기능을 하는 완벽한 프로그램을 만들어 그것을 하나로 합침.

Testing method about program.

Change about test case

Used case → Growing software → All case

Process of Clean-Room

Clean-Room과정은 모든 case를 test함.

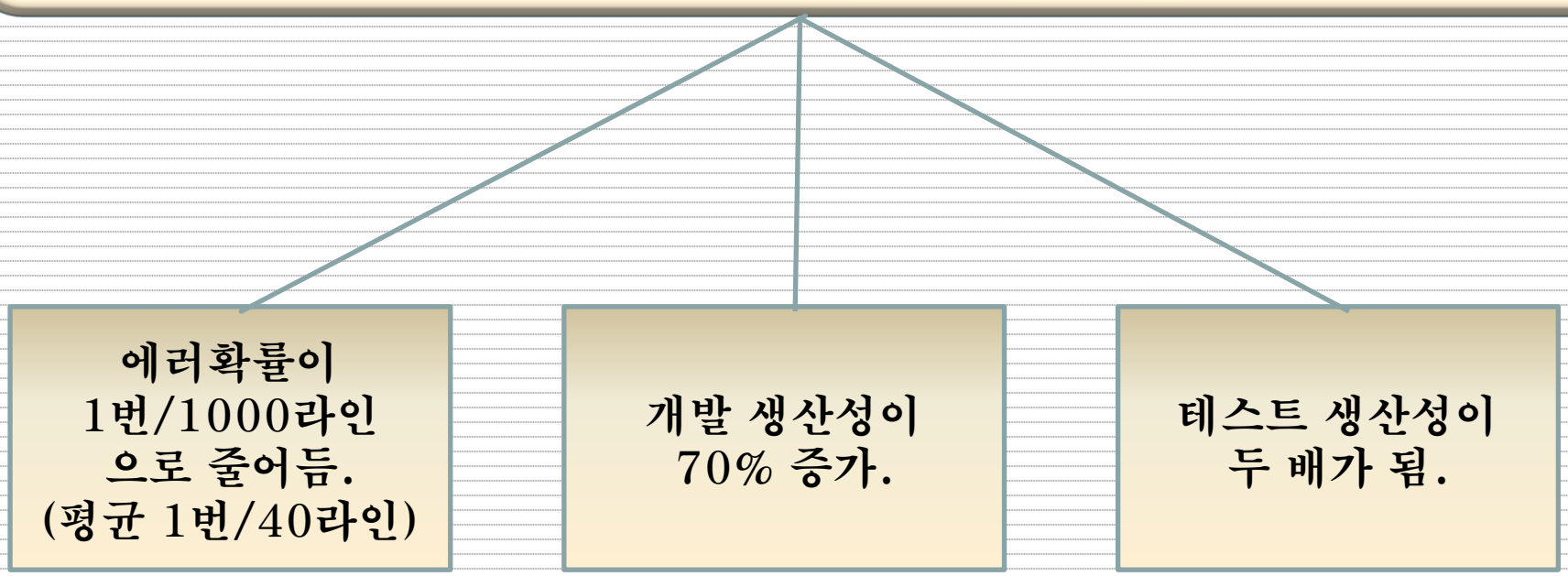
case를 모아 test case를 만듦.

프로그램은 이 test case 상에서 실패까지 도달하는 시간을 잼.

이 시간들은 프로그램의 신뢰성을 계산하는 모델에 반영됨.

Example of Clean-Room Process

스웨덴의 이동통신사 에릭슨은
자동전화교환방식 컴퓨터들을 위한
OS제조에서 Clean-Room Process를 사용함.



Software Standardization

Effect of Standardization

- 머스켓 제작자들의 생산성 향상.
- “Just Add Water”

Barrier of Standardization

- 다른 프로그래밍 언어, 컴퓨터 플랫폼, 운영체제로 옮겨졌을 때 작동 불가.
- 소프트웨어를 부분으로 분리하는 데 어려움이 있음.

Brad Cox's Component Company

Problem

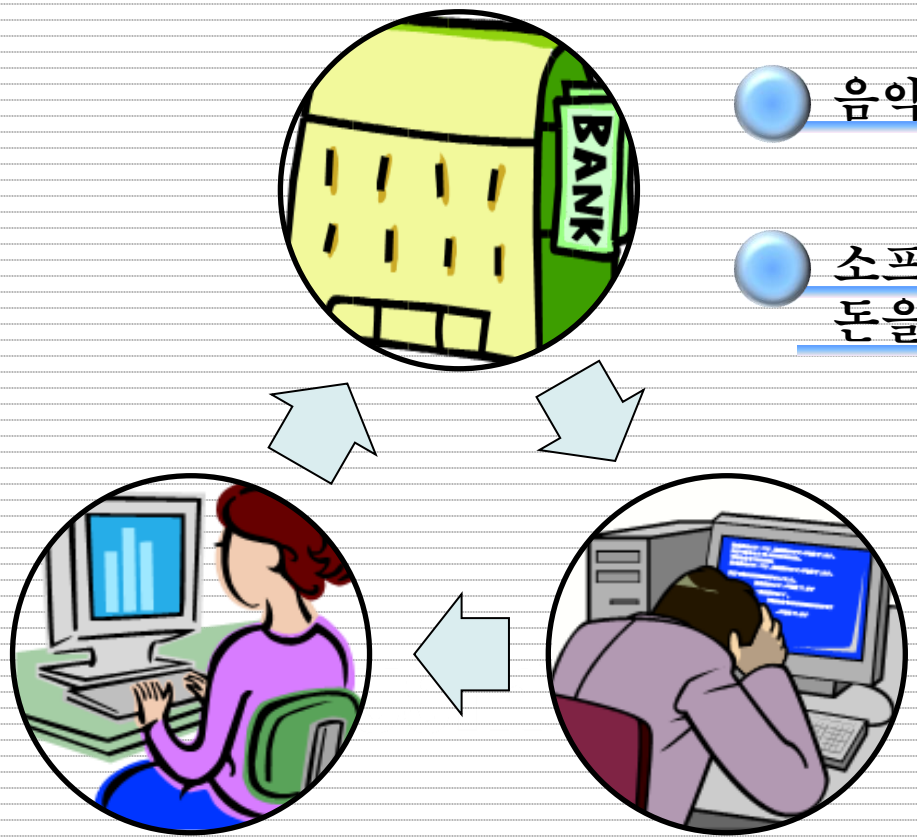
- 실리콘 IC에 비해 소프트웨어의 시장 가격이 너무나도 낮음.

Reason

- 컴포넌트를 일일이 각각의 고객을 위해 개조할 수 없다.
- 구매자들의 무단 복제 경향.

Software Industry vs Music Industry

Solution of music companies



- 음악이 방송에 수신될 때마다 로열티를 받음.
- 소프트웨어 컴포넌트를 사용할 때마다 돈을 받도록 제안.

Generalization of Programming

Background

- 산업적 프로세스 조절, 고급 공업기술 도구, 교체 가능 프로그램들의 조합.
- 누구나 자신을 소프트웨어 엔지니어로 광고할 수 있는 상황.

Question

- 누가 중요한 시스템을 구축하기에 적합한가?
- 적절한 기준이 마련되어 있는가?

Silver Bullet

Claim of OO Advocates

- 패러다임의 전환.
- 더 줄어든 가격으로 생산성의 “14-to-1 improvement” 를 이뤄낼 것.

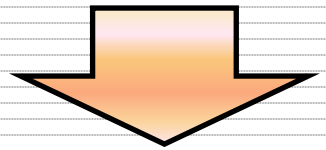
What makes them skeptical

- 소프트웨어 개발의 생산성은 분야의 성숙에 비해 뒤떨어지고 있음.
- 하드웨어 공학부분에 의존한 업적들이 많음.

Who is productive developer?

The Reason No one knows how productive programmer is

- 미국 회사들의 10% 이하만이 일관적으로
  그들의 프로그래머들의 생산성을 측정하고 있음.
- 기업들은 아직 유용한 측정의 표준 단위를 정하지 못함.
- 같은 증명서를 가진 개발자 사이에서도 개인차가 존재함.



어떤 누구도 자신을 소프트웨어 엔지니어로 광고할 수 있다.

Computer Science Need Laboratory

Problem

- 컴퓨터 과학은 실험을 필요로 함.
- 소프트웨어 공학을 위한 실험실의 부재.

Solution

- 정부의 역할의 중요성.
- 실제 소프트웨어 개발 조직과 실험하는 협력관계의 예 : NASA.

Conclusion

Reference : A Developing World

1. Change's in Software Market
2. Growth in India's Software Engineering
3. What have attracted multinational firms

1 – Change's in Software Market

American's Market Govern

- 매년 100개 국가 이상이 생산해내는 코드보다 많은 코드를 생산.
- 미국의 공급업체가 세계 소프트웨어 시장의 70% 이상을 차지.

Division in software development

- 개발 도상국들의 노동력을 바탕으로 한 시장 진출.
- 시스템 설계 분야(서양 소프트웨어 엔지니어)와
  구현 분야(동양 프로그래머)로의 개발분야 분리.

2 – Growth in India's Software Engineering

India's software export

- 지난 5년간 38% 성장, 지난해 60%의 성장률 : 세계의 4배 수준.
- 1997년도에 수출 10억달러 달성.

Support of India government

- 관세와 규제 철폐.
- 소프트웨어 기술 단지에 보조금을 지급하고 소프트웨어 수출사에 대해 5년간의 세금 면제 제공.

3 – What have attracted multinational firms

Labor

- 소프트웨어 파트 당 125달러 : 미국은 925달러.
- “body shopping” 의 증가 : 인도 소프트웨어 수출의 절반 이상 차지.

Growing Trust in Project Management

- 은행 시스템의 개발을 봄베이에서 한 Citicorp와 벵갈로르에서 자사의 이리듐 위성 네트워크를 위한 소프트웨어를 제작하는 Motorola의 사례.
- 미국의 프로그램 코드 작성의 주도권이 사라지고 있음.

Progress Toward Professionalism