애자일 소프트웨어 개발 방법론
Agile Methodology

김창준 (애자일 컨설팅 대표)
june@agileconsulting.co.kr

애자일 컨설팅은...

국내에서 애자일 소프트웨어 개발 방법론을 전문적으로 컨설팅 하기 시작한 최초의 회사

삼성전자/전기, 삼성SDS, LG전자, LGCNS, 팬택앤큐리텔, SKT, SK Communications, 한미연합사령부, 다음커뮤니케이션, NHN, 엔씨소프트, 한국생명공학연구원 등에 교육, 컨설팅, 코칭 등 제공

프로젝트 성공과 실패

성공적인 프로젝트들을 떠올려보자. 어떤 공통점이 있는가.

실패한 프로젝트들을 떠올려보자. 어떤 패턴이 보이는가.

Agile?

1. able to move quickly with skill and control

2. able to think quickly and intelligently

Agile Methodology

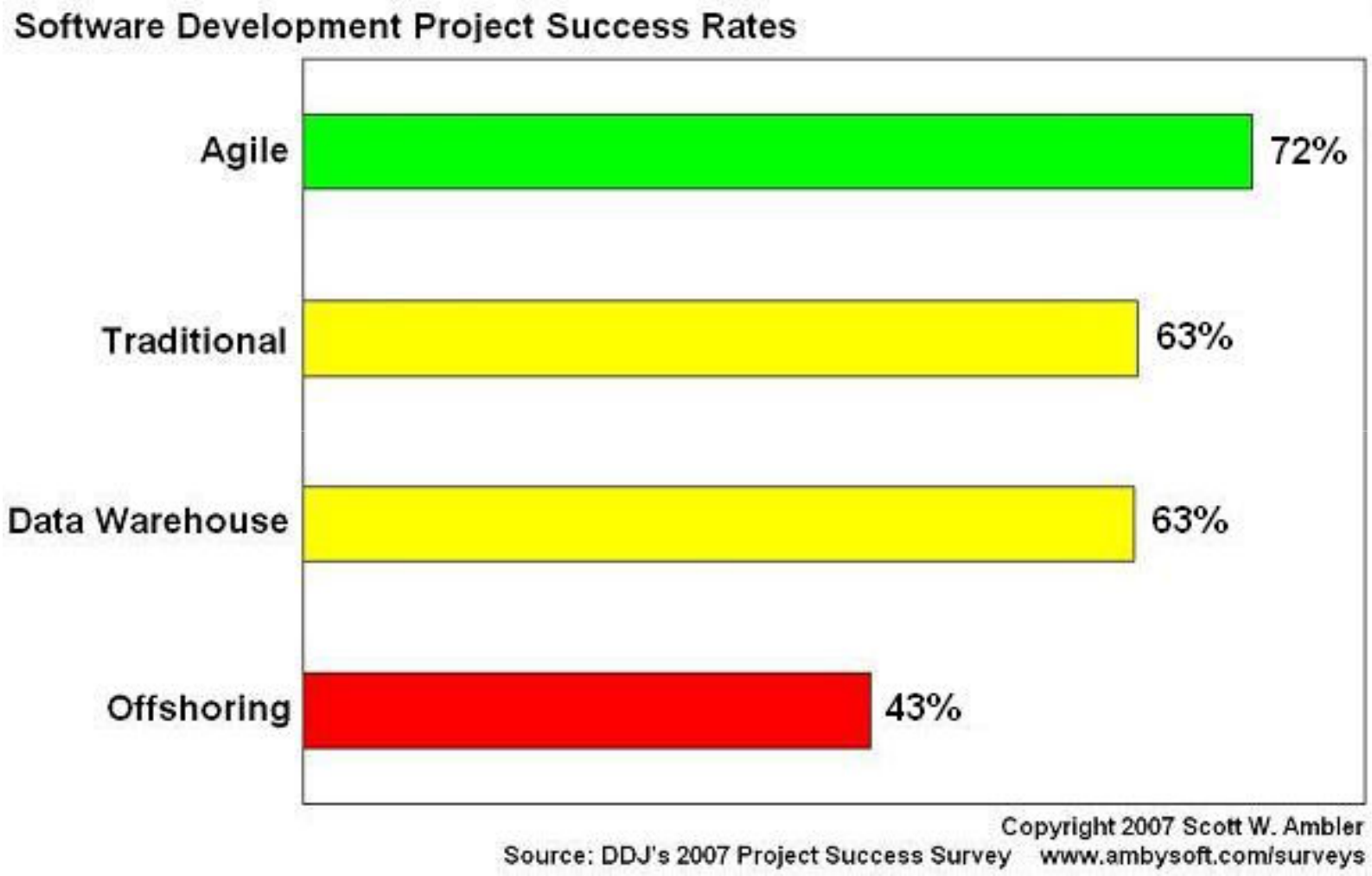
1990년대 중반 이후부터 덩치 크고 무거운 방법론들에 대한 대안들이 대두되기 시작. 변화 대응력, 민첩성 등을 특징으로 함.

2001년 스노우버드에 서로 다른 방법론의 수장들이 모여서 서로의 공통점을 논의. 애자일 방법론이라는 이름을 지음.

현재 전세계적으로 애자일 방법론에 대한 높은 관심을 보이고, 점차 많은 성공사례들이 발표되고 있음.

애자일의 효과

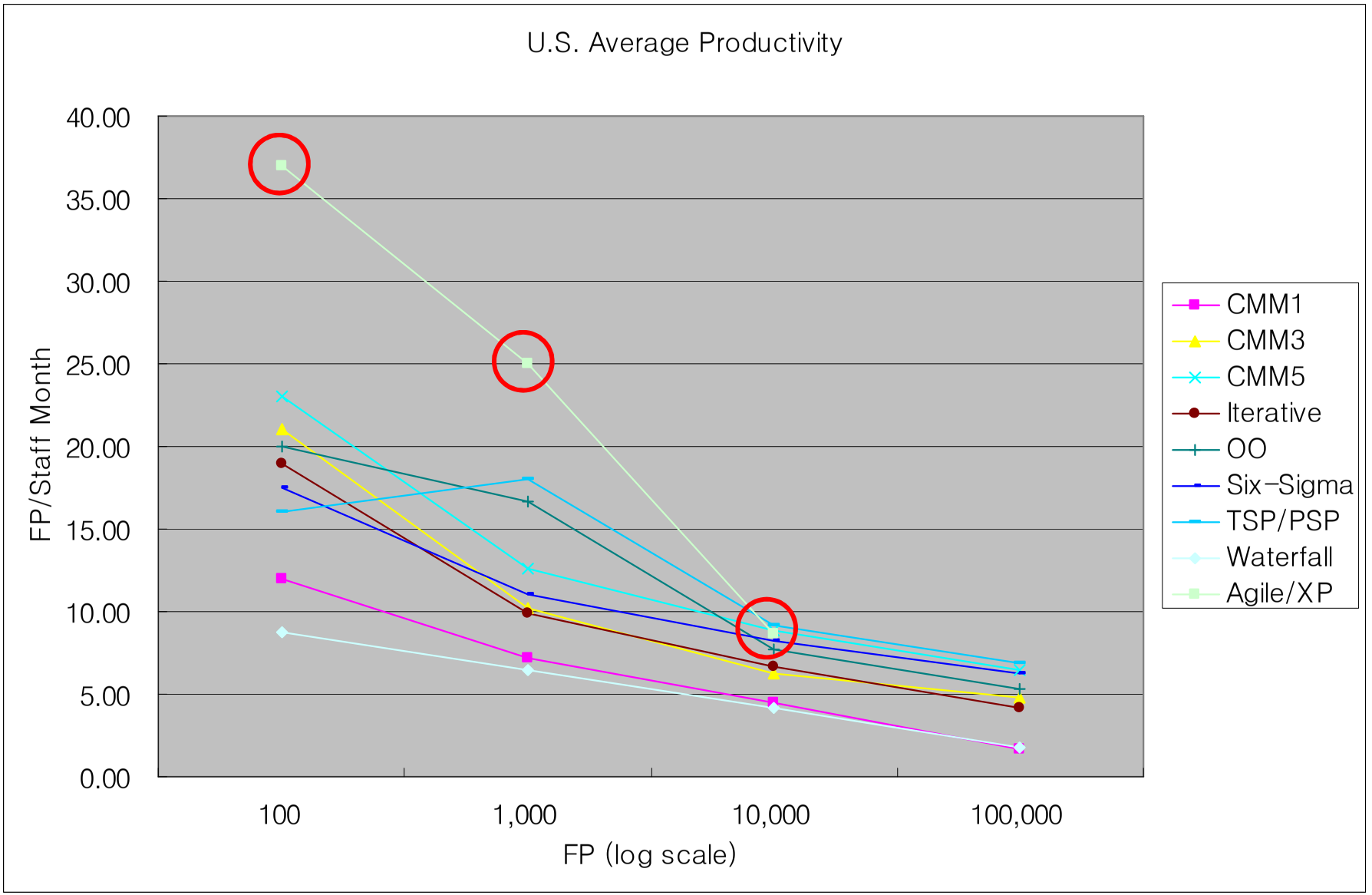
방법론별 성공률 비교


Software Development Project Success Rates
Agile
72%
Traditional
63%
Data Warehouse
63%
Offshoring
43%
Copyright 2007 Scott W. Ambler
Source: DDJ's 2007 Project Success Survey www.ambysoft.com/surveys

프로젝트 성공 원인


THE STANDISH GROUP
Top Ten Reasons for Success
1. User Involvement
2. Executive Management Support
3. Clear Business Objectives
4. Optimizing Scope
5. Agile Process
6. Project Manager Expertise
7. Financial Management
8. Skilled Resources
9. Formal Methodology
10. Standard Tools and Infrastructure
Copyright © 2006 The Standish Group International, Inc..

방법론별 생산성 비교



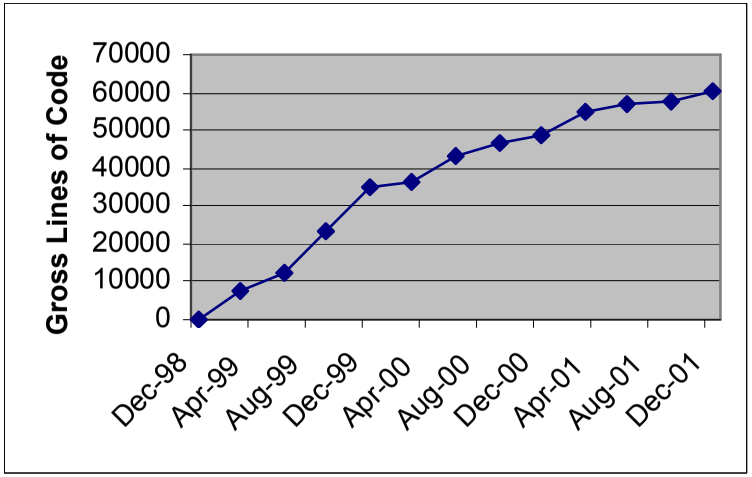
From Applied Software Measurement 3E, Capers Jones

애자일 적용 사례

GMS 프로젝트 (임베디드)

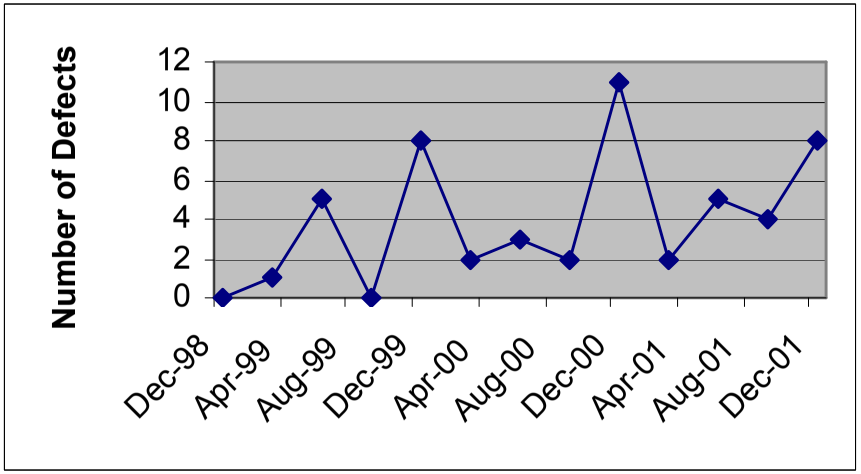
구성 인력 절반 이상이 임베디드 초보

일정 수준 이하로 결함률 유지



0줄에서 6만줄로 코드 증가 했는데...

그 동안 결함률을 일정 수준 이하로 유지



- 생산성: 임베디드 업계 표준에 비해 292% (ESLOC/hr)

- 결함률:
  - 3년간 전체 버그 51개
  - 임베디드 업계 표준 441개(동일 코드 크기 기준)

해외 적용 회사/프로젝트

Hewlett-Packard, IBM, Oracle, Sun Microsystems

Sabre Airline Solutions,

Microsoft, Google, Yahoo,

Nokia, Ericsson, British Telecom,

Philips Research, General Electric Healthcare,

SAP, Siemens,

BBC,

US Federal Reserve

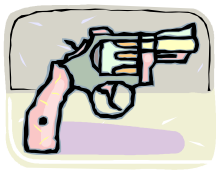
…

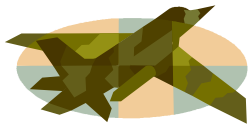
국내 도입 상황

- SI 업계 : 삼성 SDS, LG CNS, 티맥스 등
- 웹 서비스 업계 : NHN, 다음커뮤니케이션, 야후 코리아 등
- 게임 업계 : 엔씨소프트, 네오플, 티쓰리엔터테인먼트, 엔트리브, 웹젠 등
- 임베디드 업계 : 삼성전자, LG전자, 아이디스 등
- 금융 업계 : 퍼스트데이타코리아(신용결제 VAN 사업)
- 연구조직 : 한국생명공학연구원, 서울대/이화여대 산학협동 등

애자일의 핵심 특징



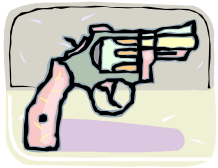


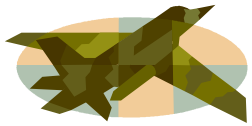


분석 설계 구현 테스트 전개

→ → → → →



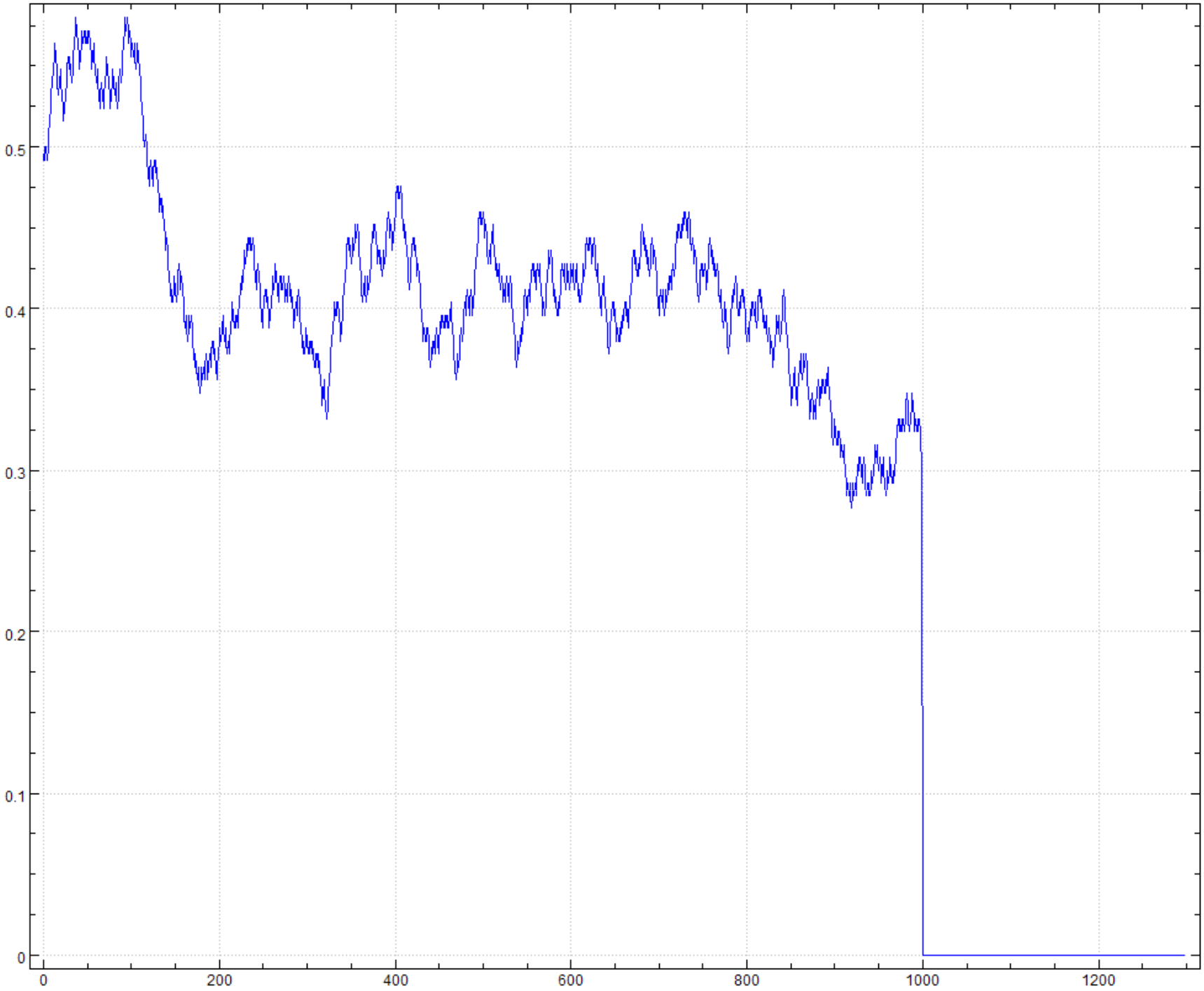




분석 설계 구현 테스트 전개

"여기까지 석 달 걸렸는데
생각한 것보다 조금 더 걸렸고
한 60% 진행됐으니까 ... 흠...
두어달 더 있으면 되겠네요."

설마가 사람 잡는 경우

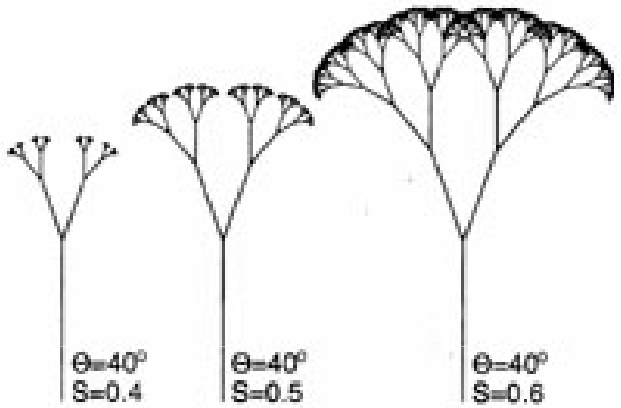

0.5
0.4
0.3
0.2
0.1
0
0
200
400
600
800
1000
1200

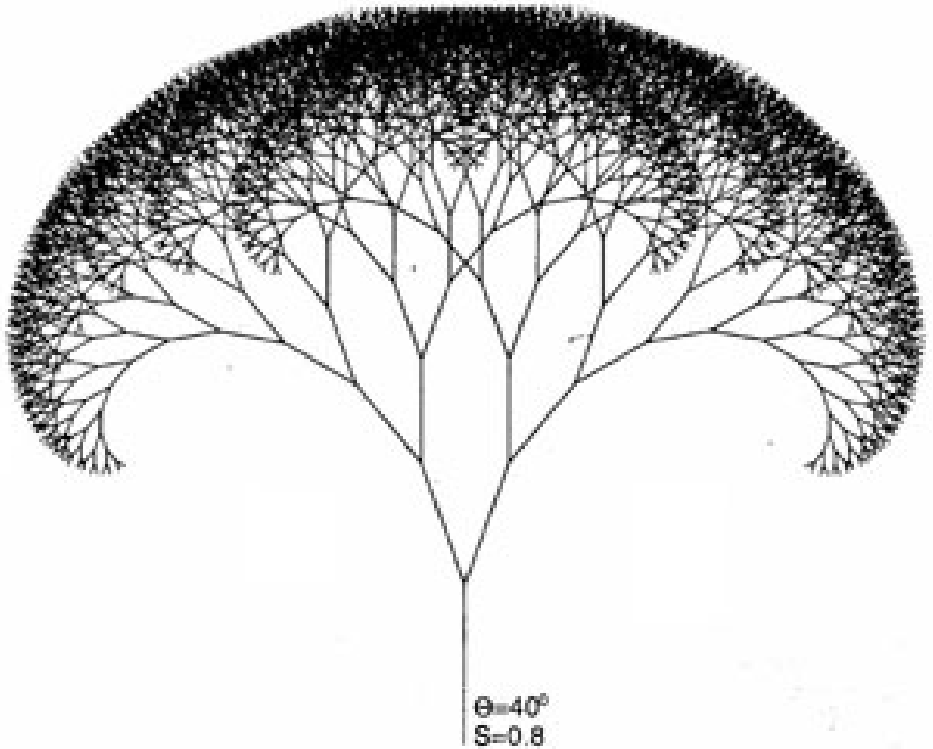
확률/통계의 설명력 :

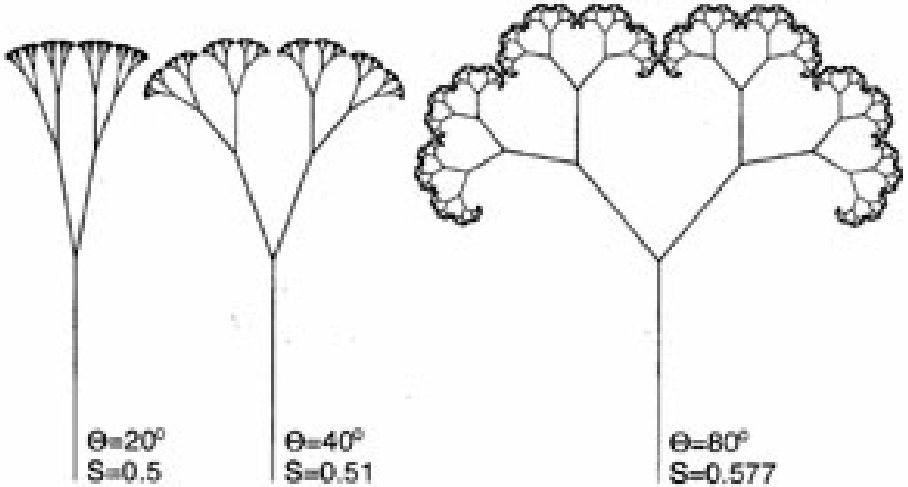
과거와 미래에 연결점이
있을 때

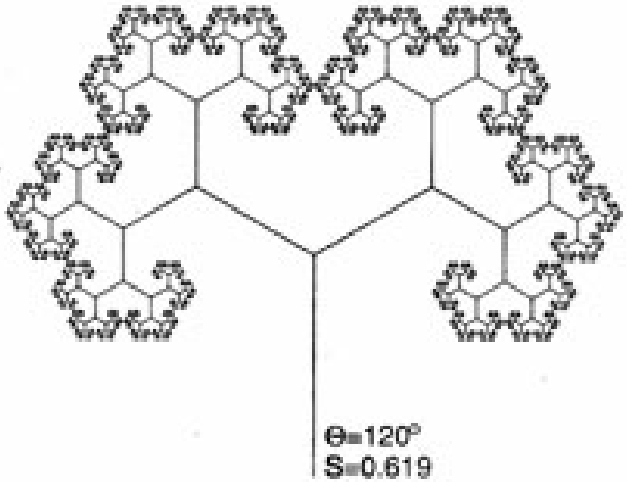
프랙탈

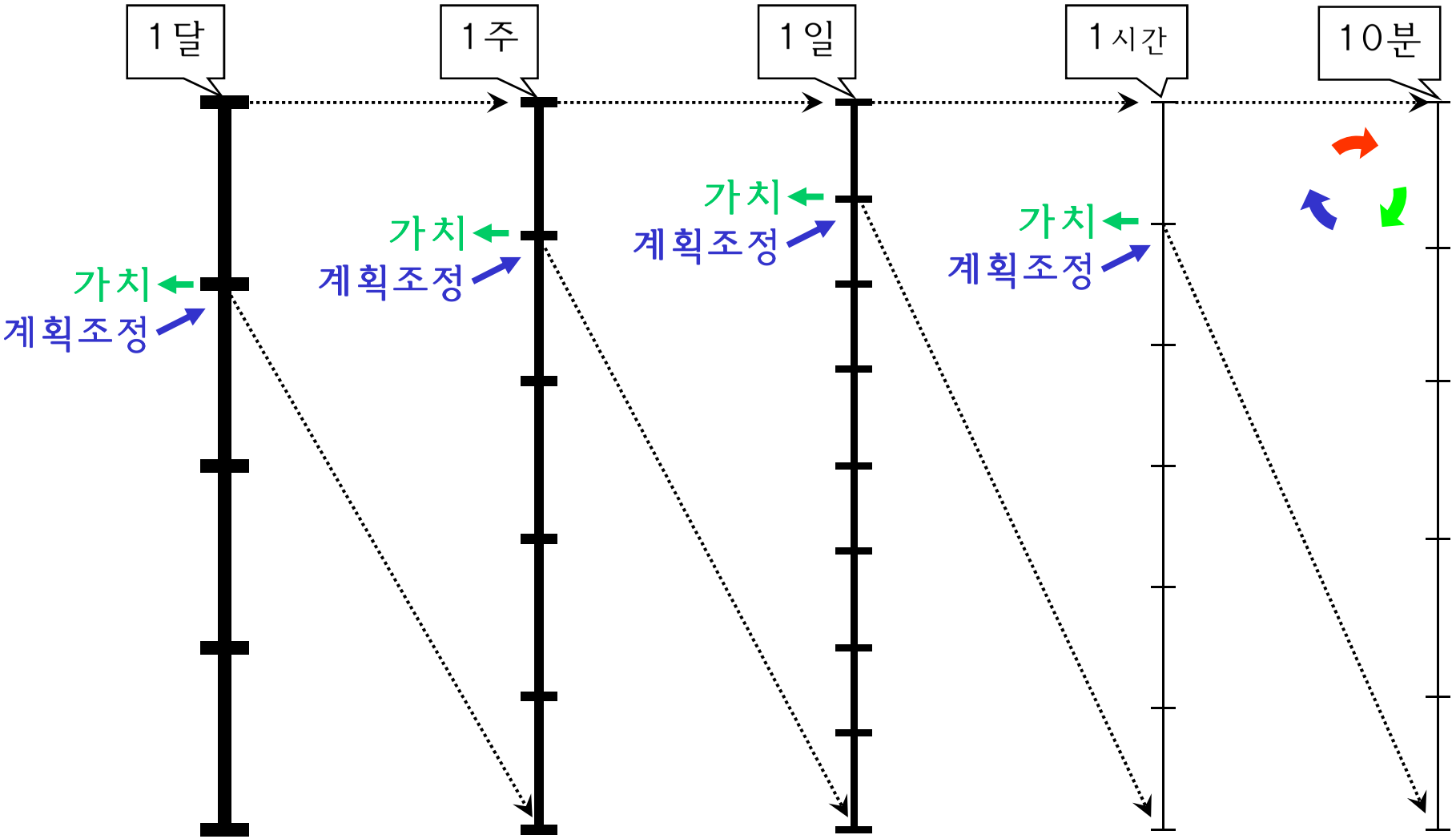
(자기 유사성)

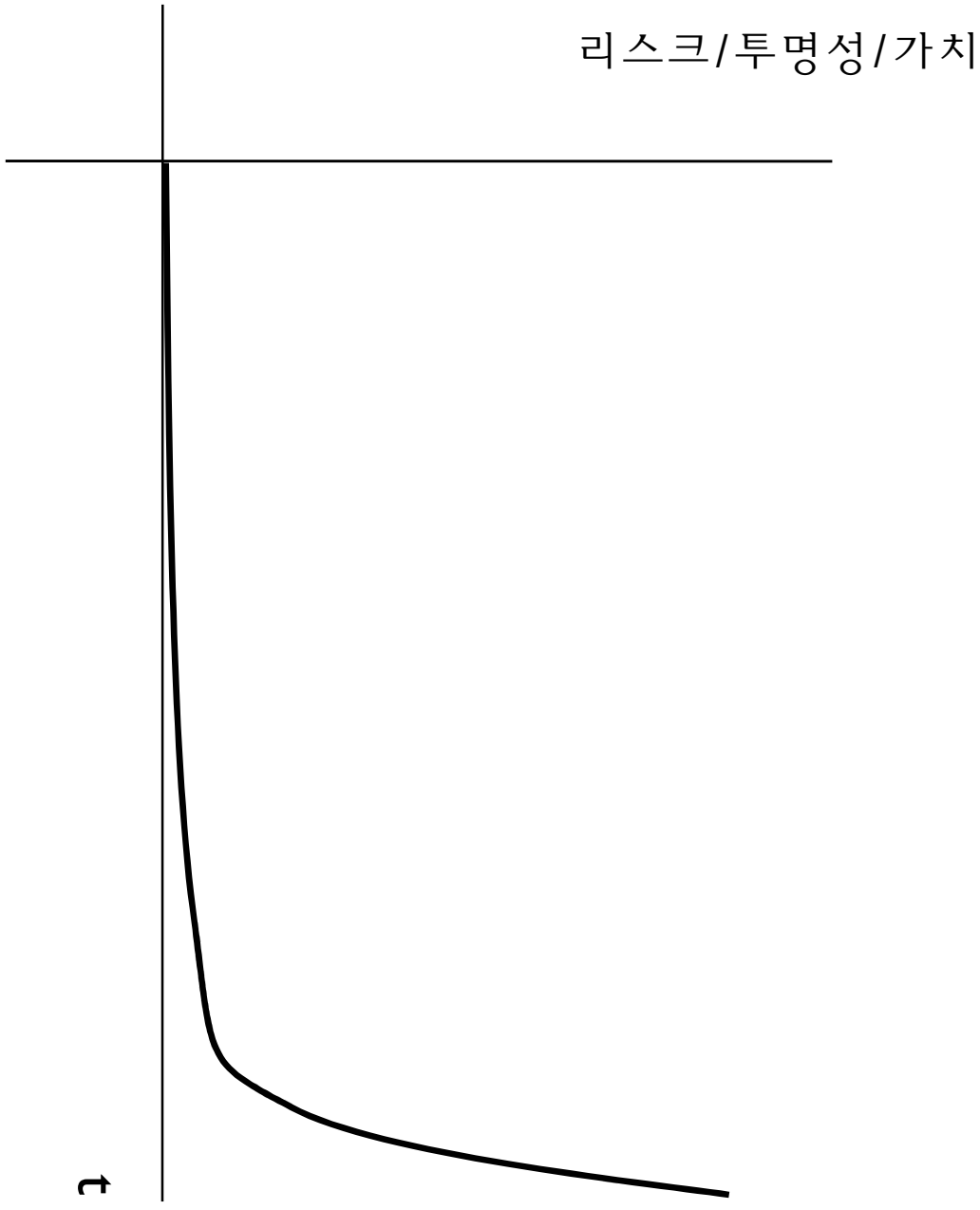

θ=40°
S=0.4
θ=40°
S=0.5
θ=40°
S=0.6

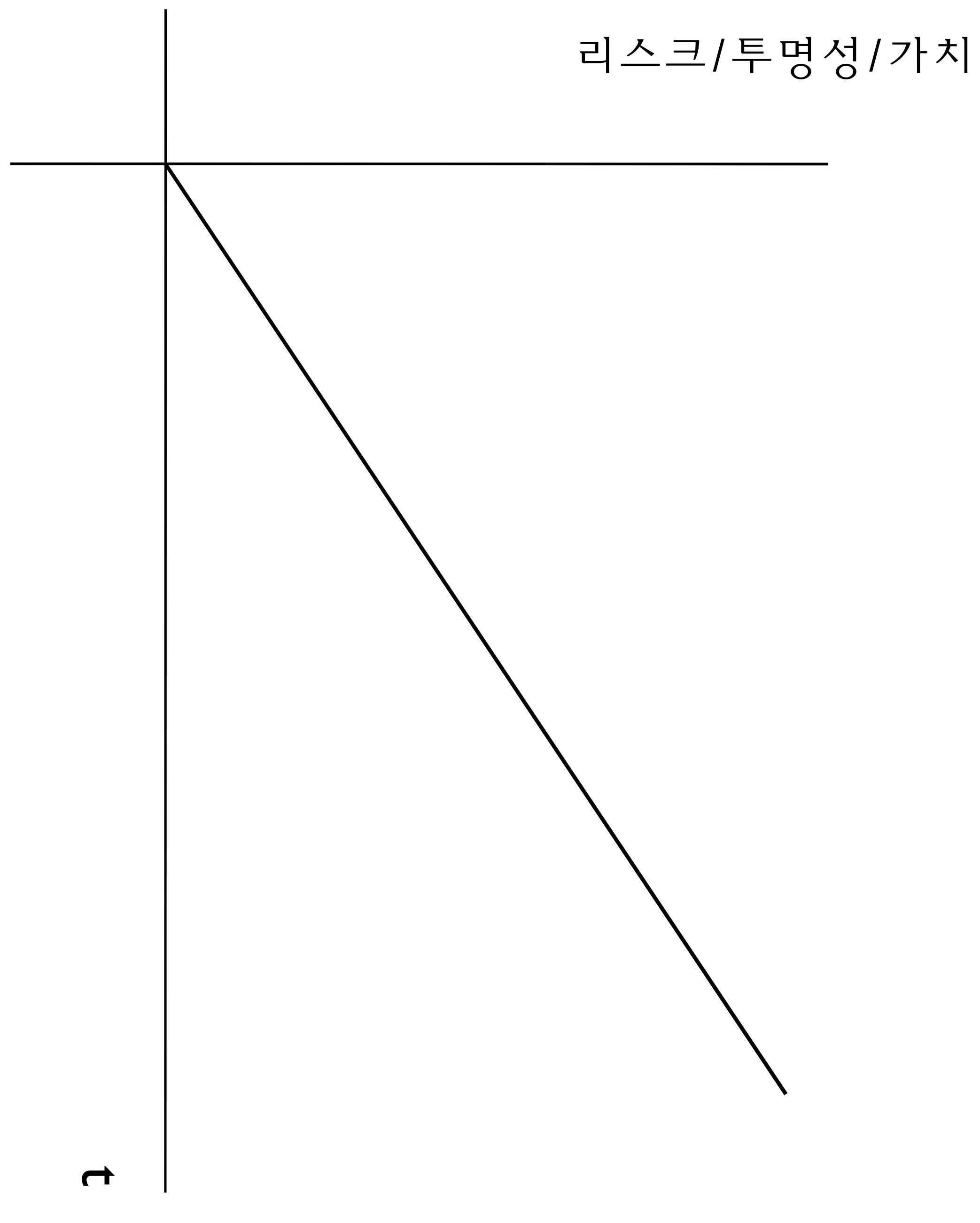

θ=40°
S=0.8


θ=20°
S=0.5
θ=40°
S=0.51
θ=80°
S=0.577

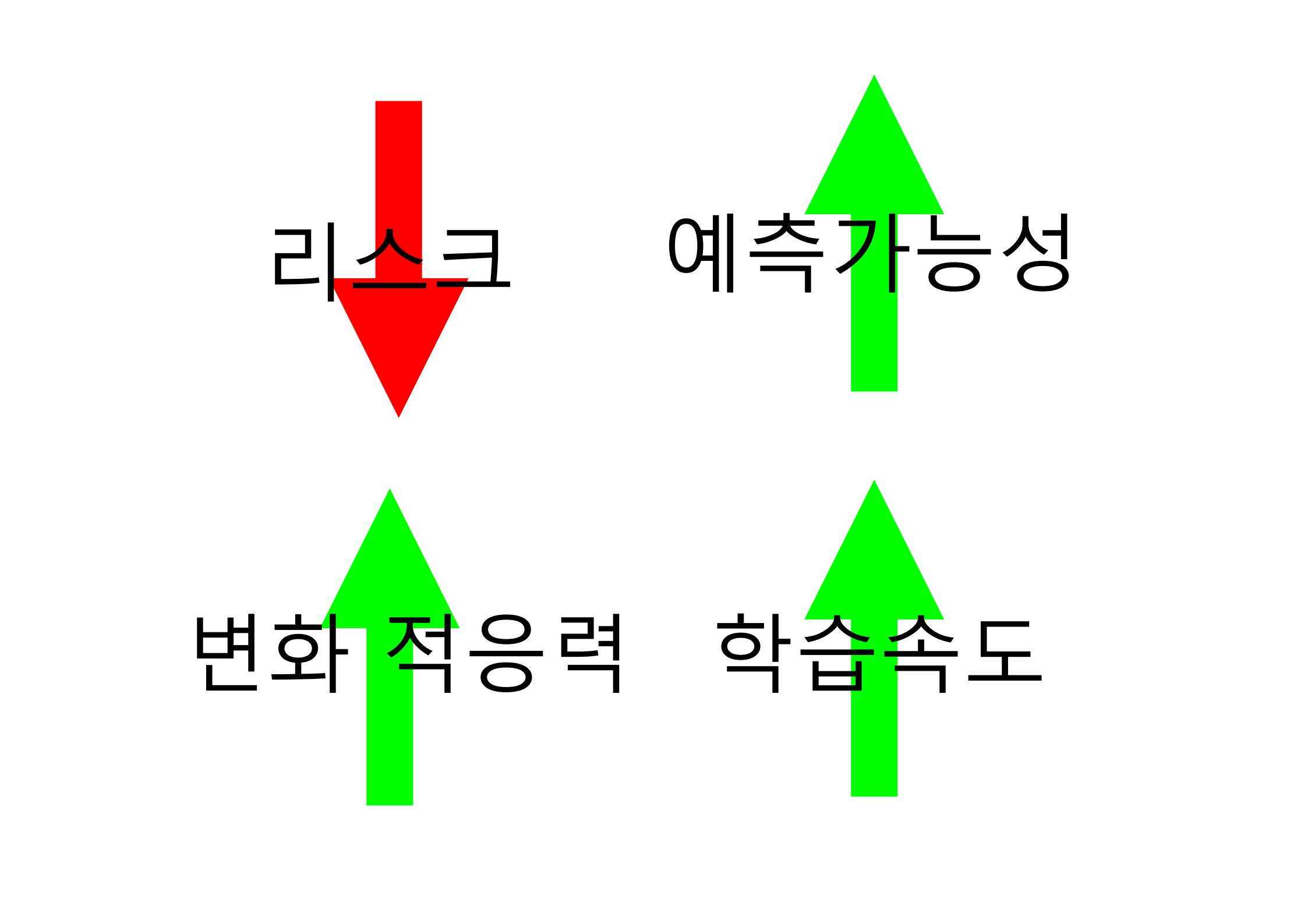

θ=120°
S=0.619


1달
1주
1일
1시간
10분
가치
계획조정
가치
계획조정
가치
계획조정
가치
계획조정


리스크/투명성/가치
t


리스크/투명성/가치
t

리스크/투명성/가치


리스크
예측가능성
변화 적응력
학습속도

연속된 작은 개선의 놀라움

매 시도 당 0.1%의 성공률 개선이 있다고 한다면 1000번의 반복후에는…

성공률 2.7배

개발자	확률
갑갑해 대리	90%
을화통 사원	90%
병[illegible]개 과장	[illegible]0%
정[illegible]리	[illegible]0%
무[illegible]사원	[illegible]0%
기똥찬 사원	90%
경운기 과장	90%

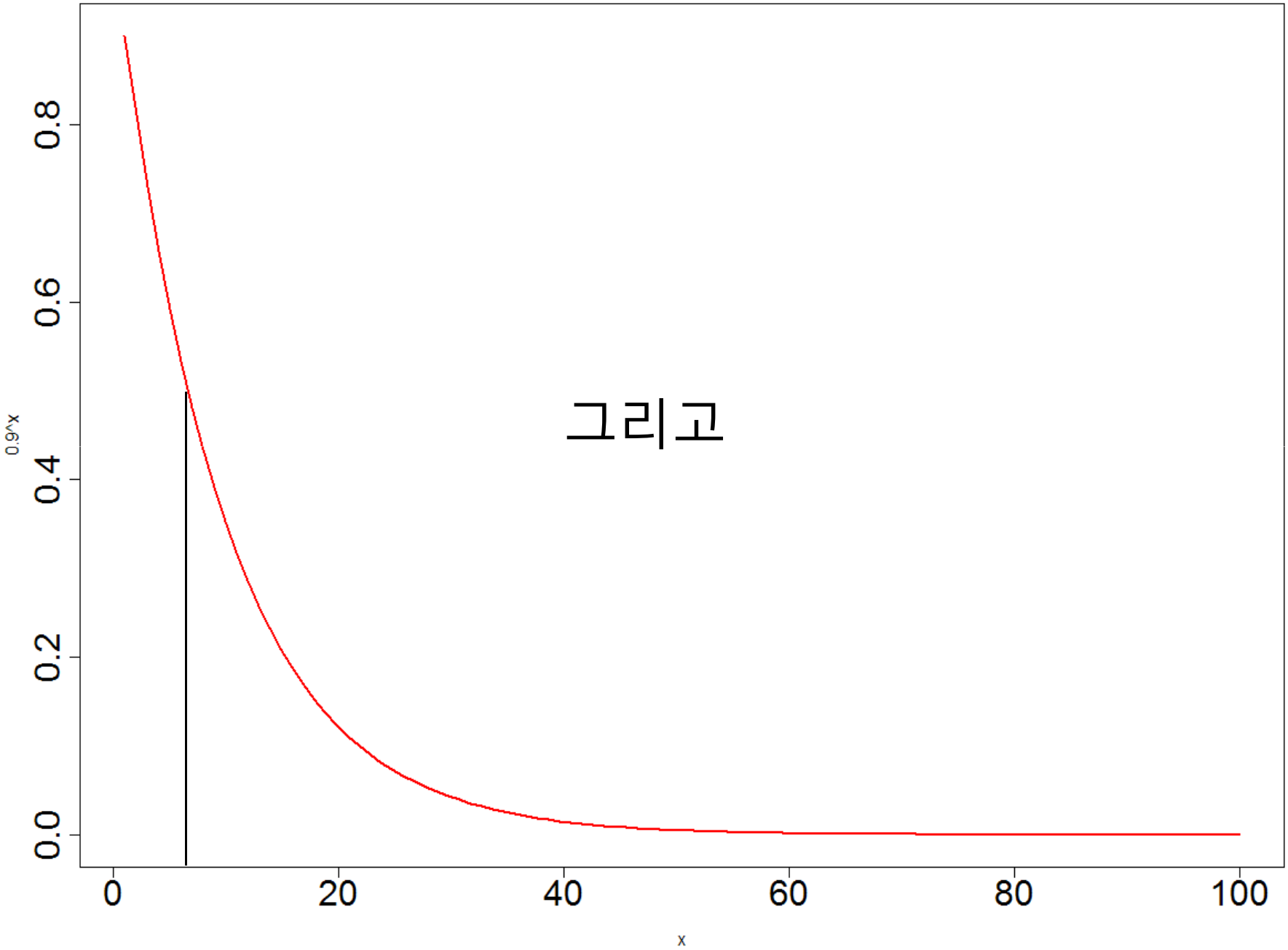
90%

$$\frac{0.9 + 0.9 + 0.9 + 0.9 + 0.9 + 0.9 + 0.9}{7}$$

$$= 0.9$$

$$0.9 \times 0.9 = 0.81$$

$$0.9 \times 0.9 \times 0.9 = 0.729$$



그리고

$$0.9^{7} \fallingdotseq 0.48$$

협 력

버그 발견의 확률론

한 사람이 1시간 동안 핵심적인 버그를 발견할 확률이 0.1 이라고 하고, 사람이 총 7명이라고 하면,

모든 사람이 버그를 발견할 확률은?

0.1^7 = 0.0000001

한 사람이라도 버그를 발견할 확률은?

1-0.9^7 = 0.53

한 사람이 뭔가 새로운 것을 학습했을 때, 그 지식을 모든 사람과 공유할 수 있다면...

图3

애자일은 협력을 통해 집단 학습을 가속화시키고, 결과적으로 프로젝트 생산성을 향상시킨다.

애자일 방법론의 핵심

1)세상이 변하고
2)우리가 더 똑똑해지는

상황 하에서는

협력과 집단 지성을 통해, 서로 학습하면서 큰 프로젝트를 여러 개의 작은 프로젝트인 것처럼 만들어 프랙탈적으로 시도하는 것이 훨씬 유리하다.

해외 정부 지원 사례

Agile-ITEA (Mobile-D)

유럽연합(9개국)에서 임베디드 분야를 위한 애자일 방법론 연구, 도입을 지원

필립스, 노키아 등 참여

역사상 가장 생산적인 팀 중 하나였던 윈도우용 볼랜드 쿼트로 프로 개발팀은 그 생산성이 업계 평균에 비해 최소 20배가 넘었다. 숫자로만 이야기한다면, 같은 숫자의 사람들로 20년 걸리는 일을 이 사람들은 1년이면 할 수 있었다는 말이 된다.

SACWIS

주 범위 자동화 아동 복지 정보 시스템

(Statewide Automated Child Welfare Information System)

동일 요구사항 하에 모든 주에서
시스템을 구현해야 함

플로리다주 대 미네소타주

플로리다

1990년 프로젝트 시작

초기의 경비 예상 $32M

1998년 완료로 계획

현재 109명이 작업 중

현재까지 $170M 사용(2002년)

$230M로 추정 비용 정정

2005년으로 완료 시점 정정

IBM 컨설턴트 투입

미네소타

1999년 프로젝트 시작

완료 계획:

99년 9월 Phase 1

00년 중반 Phase II

예상 비용 $1.1M

2000년초 완료

8명

총 $1.1M 소요

극과 극?

같은 일을 하는 시스템인데...

1990년 프로젝트 시작
초기의 경비 예상 $32M
1998년 완료로 계획
현재 109명이 작업 중
현재까지 $170M 사용(2002년)
$230M로 추정 비용 정정
2005년으로 완료 시점 정정
IBM 컨설턴트 투입

1999년 프로젝트 시작
예상 비용 $1.1M
완료 계획:
99년 9월 Phase 1
00년 중반 Phase II
2000년 전반기 완료
8명
총 $1.1M 소요

왜?

성공하는 프로젝트의 특징

무슨 특징이 있었나?

어떤 조건과 상황이 그런 성공을 가능하게 했을까?

성공적인 팀의 7가지 특징

전세계에 성공적인 팀들을 조사했더니 다음 7가지 공통점이 있었다. 특히 처음 세가지는 거의 모두에게 해당했다.

Frequent Delivery

지난 6개월간 실행되고, 테스트 했고, 사용 가능한
코드를 사용자 커뮤니티에 최소 2회 이상 제공했는가?

Reflective Improvement

지난 3개월간 최소 한번은 모두 모여서
작업습관에 대해 고민하고 토론해 봤는가?

무엇이 우리의 속도를 높이고 또 무엇이 속도를
떨어뜨리며 뭐를 개선할 수 있는지 토론해 봤는가?

Osmotic Communication

당신의 질문이 답을 가진 사람의 눈 혹은 귀에
도달하는데 30초 미만이 걸리는가?

최소 며칠에 한번은 다른 팀원의 대화에서 뭔가
나에게 유용한 것을 엿듣는가?

Personal Safety

50% 이상 추정이 잘못되었다고 상사에게 말할 수 있는가?

매력적인 이직 제안이 있었다고 말할 수 있나?

팀 회의에서 스케쥴에 대해 상사에게 반대 표현을 할 수 있나?

서로의 설계에 대해 오랜 논쟁 후에 기분좋게 disagree할 수 있나?

Focus

각자 자신이 해야 할 최고 우선순위 일 2개가
무엇인지 모든 사람이 알고 있나?

최소 이틀 연속, 매회 2시간 이상 인터럽트 없는
시간을 보장 받을 수 있나?

Easy Access to Expert Users

시스템의 사용에 대해 질문이 생긴 시점부터 전문 사용자가 답할 때까지 평균 3일 이하가 걸리는가?

몇 시간 이내에도 가능한가?

Technical Environment

with

Automated Tests, Configuration Management, and Frequent Integration

물리적으로 옆에 있지 않으면서 시스템 테스트를 종료할 때까지 자동 실행할 수 있는가?

모든 개발자가 자기 코드를 CMS에 넣는가? 주 당 최소 2회는 통합 하는가?

eXtreme Programming
익스트림 프로그래밍

XP란?

사회적 변화의 메커니즘
개발 스타일
개선으로 가는 길
인간성과 생산성을 화해하려는 시도
소프트웨어 개발 규율(discipline)

개선

지금 상황과 상관없이 여러분은 언제나 더 나아질 수 있다.

더 나아지는 일은 언제나 스스로부터 시작할 수 있다.

더 나아지는 일은 언제나 오늘부터 시작할 수 있다.

XP의 차별성

짧은 개발 주기
점진적 계획 접근방법
비즈니스 쪽의 변화 요구를 포용
자동화된 테스트에 의존
커뮤니케이션시 구두 전달, 테스트, 소스 코드에 의존
진화적 설계
개인들의 긴밀한 협력에 의존
단기적, 장기적 모두 이득이 되는 실천방법에 의존

가치

의사소통

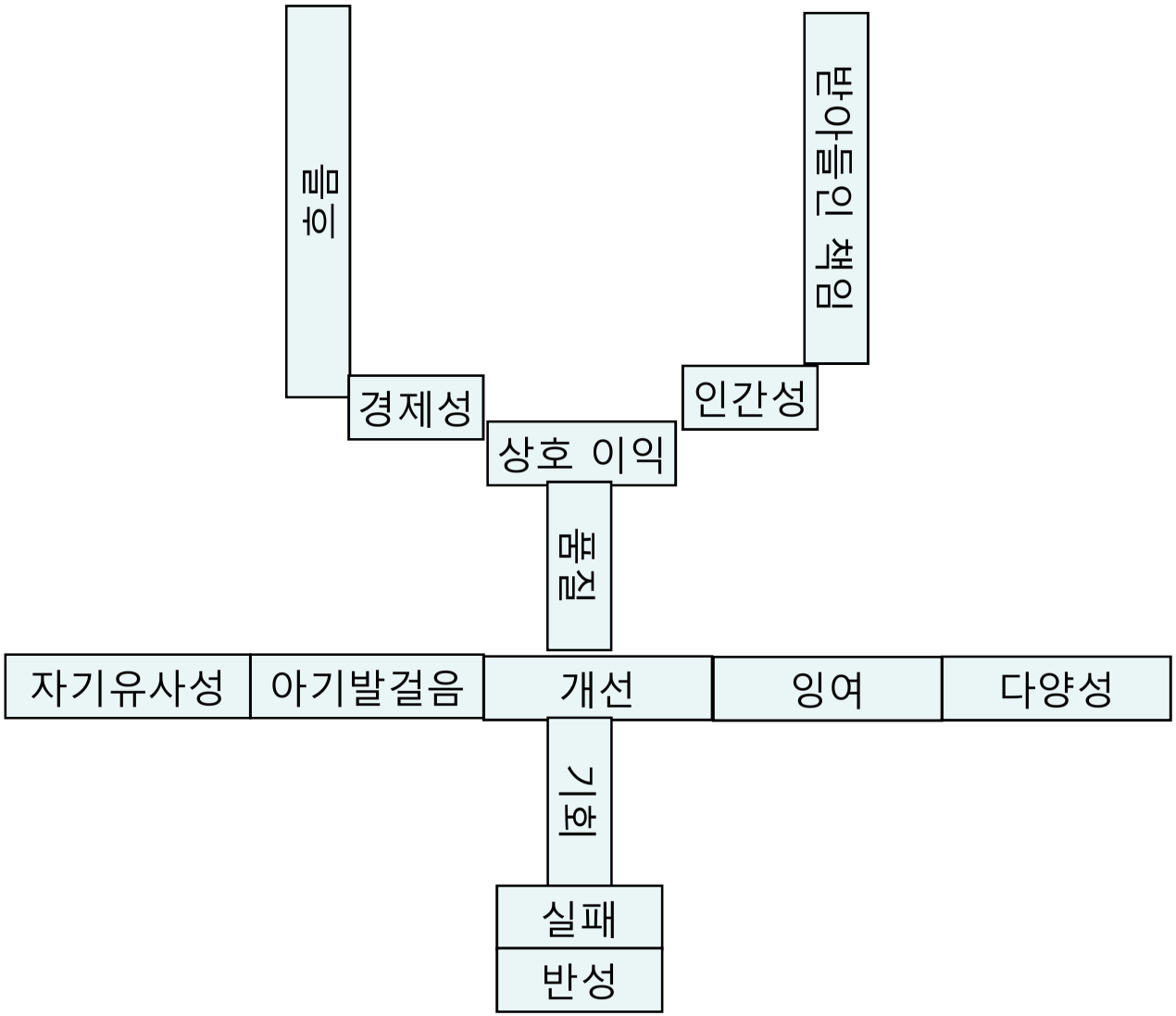
단순성

피드백

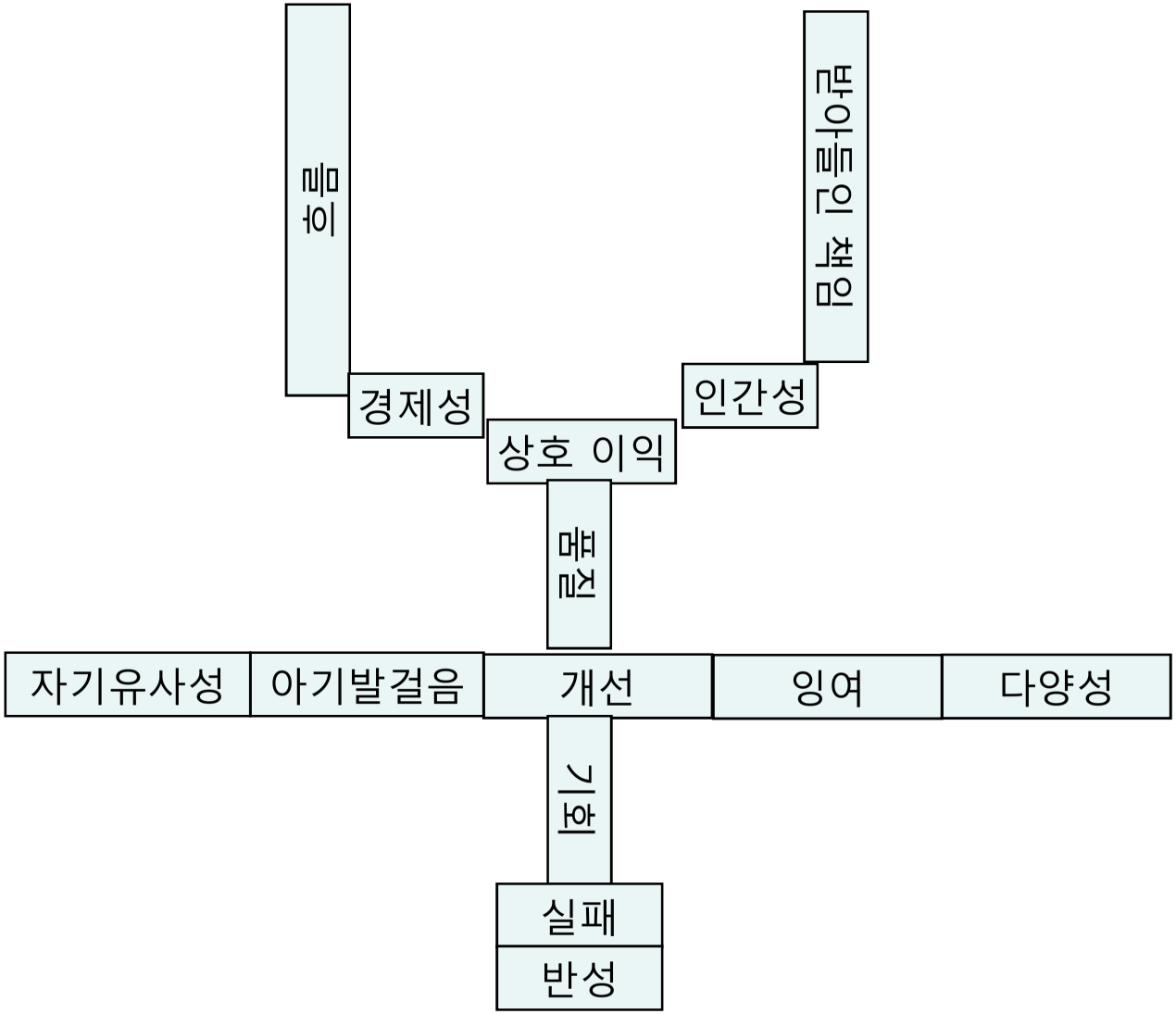
용기

존중

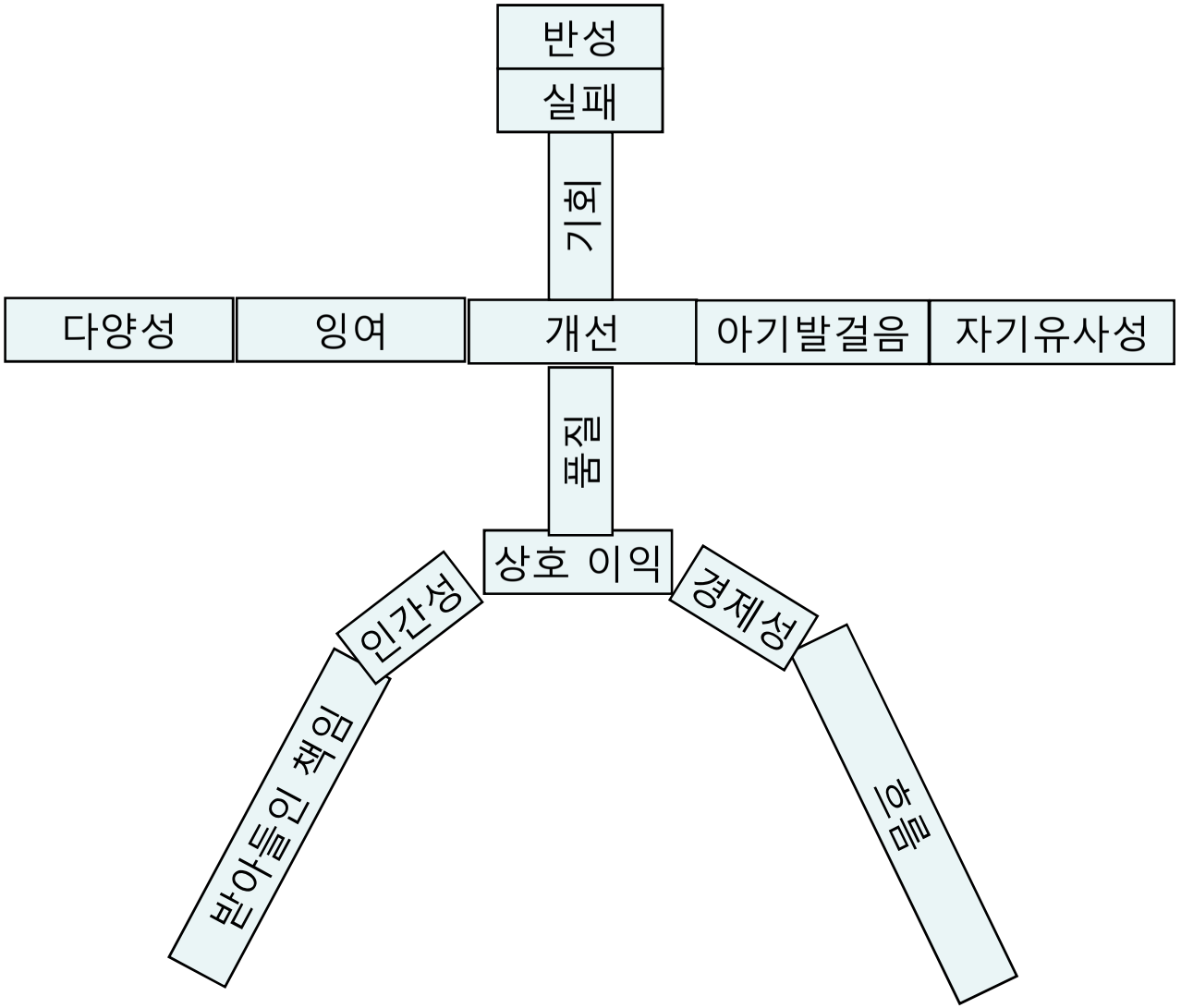
원칙



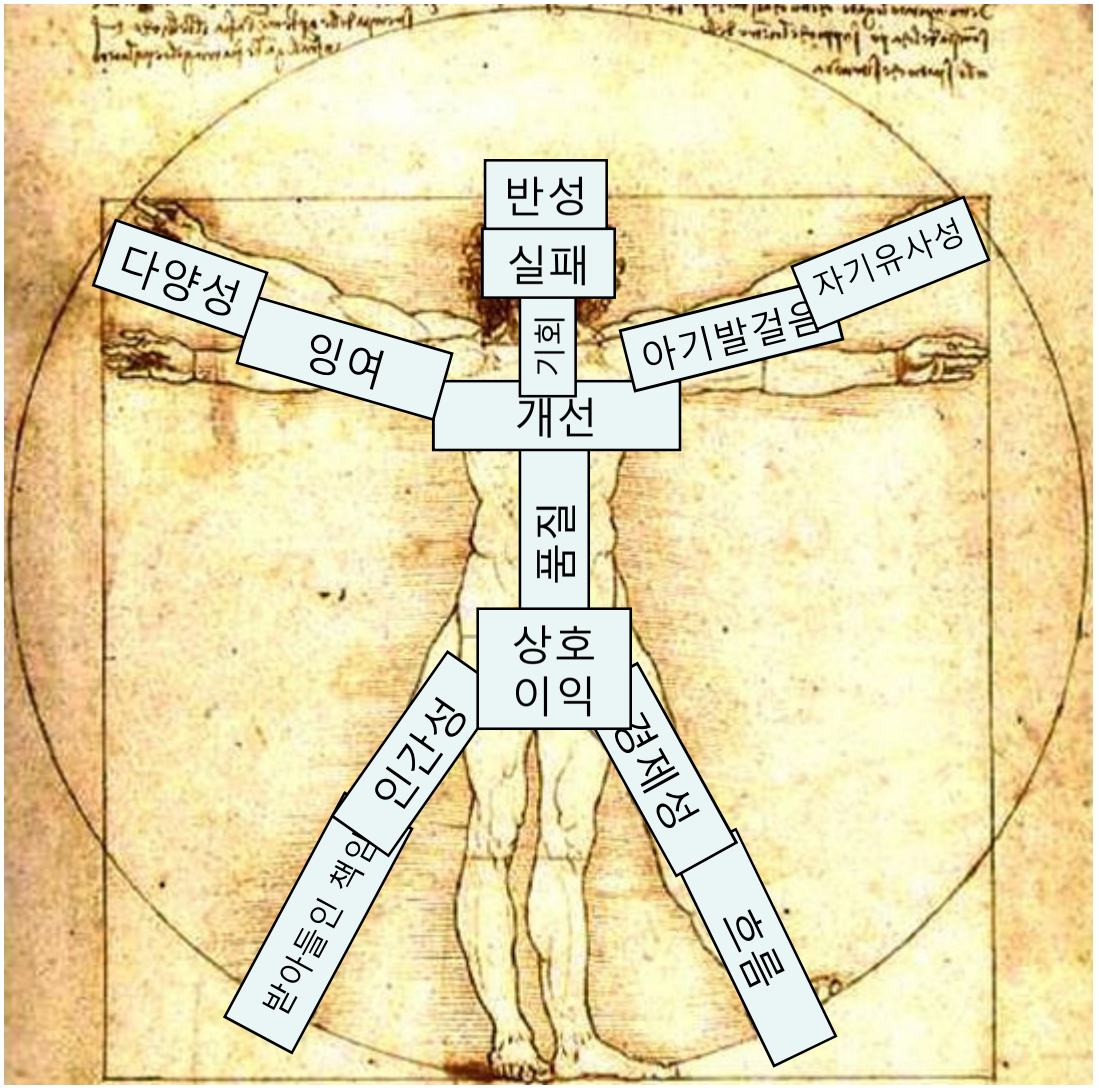
원칙



원칙

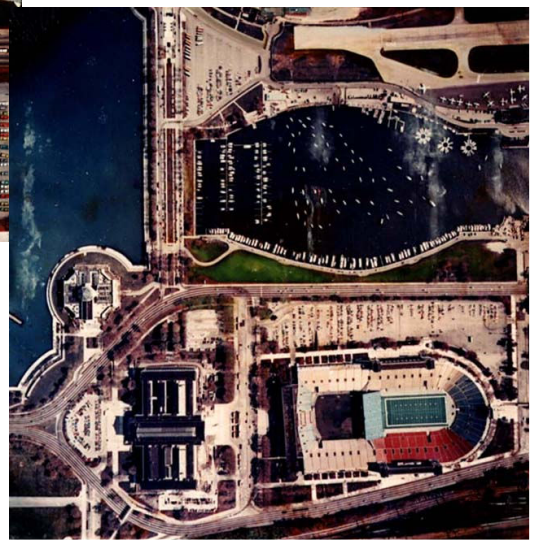


원칙



실천 방법

10의 제곱수들



초

짝 프로그래밍

분

테스트 우선 프로그래밍

10분 빌드

시

점진적 설계

지속적 통합

반일

정보를 제공하는 작업공간

일

함께 앉기

전체팀

활기찬 작업

코드 공유

코드와 테스트

매일 배치

근본 원인 분석

진짜 고객 참여

점진적 배치

단일 코드 기반

주

일주일별 주기

스토리

여유

사용별 지불

월

분기별 주기

팀 지속성

팀 크기 줄이기

범위 협상 계약

초

짝 프로그래밍

분

테스트 우선 프로그래밍

10분 빌드

시

점진적 설계

지속적 통합

반일

정보를 제공하는 작업공간

일

함께 앉기

전체팀

활기찬 작업

코드 공유

코드와 테스트

진짜 고객 참여

매일 배치

점진적 배치

근본 원인 분석

단일 코드 기반

주

일주일별 주기

스토리

여유

사용별 지불

월

분기별 주기

팀 지속성

팀 크기 줄이기

범위 협상 계약

적용 사례

Analysis – Design – Coding – Testing - Maintenance

Analysis – Design – Coding – Testing – Maintenance

Epic

User Story (prioritized)

User Research/Analysis/Model

Task Analysis/Model

Contextual Inquiry

Requirements Workshop

Analysis – Design – Coding – Testing – Maintenance

CRC Card

Whiteboard

Quick Design Session

Spike Solution

Light-weight UML

Refactoring

Simple System

Analysis – Design – Coding – Testing – Maintenance

Pair Programming

Test Driven Development

Continuous Integration

Collective Code Ownership

Analysis – Design – Coding – Testing – Maintenance

Four Quadrants

Exploratory Testing

Computer Aided Testing

Acceptance Testing

Usability Testing

Stress Testing

Bug Hunt

Analysis – Design – Coding – Testing – Maintenance

Simple Configuration Management

Bug Test

Root Cause Analysis

…

Analysis – Design – Coding – Testing – Maintenance

Learning

Pair Work

Project Retrospective/Review

Interation Retrospective/Review

Daily Scrum Meeting

Self-controlled Measurement