

3
2
4
4
4
4
2
2
4
5
3
3
0
4
4
3
5
4

Software Development Process Misconceptions

Nathalie Gaertner, ngaertner@rational.com

Agenda

- 소프트웨어 개발 프로세스란?
- 소프트웨어 개발 프로세스의 장점과 요건
- RUP
- 잘못된 인식들
- 결론

프로세스가 가져야 할 조건

- 내용
 - ✓ 어떻게 하면 목적을 달성할 수 있는가?
 - ✓ 무엇을 하면 되는가? 에 대한 가이드 라인
- 형태
 - ✓ Knowledge Database (종이 또는 전자문서 형태)
- 범위
 - ✓ 코딩, 프로젝트 관리, **요구 사항 관리**, 테스트 이슈

소프트웨어 개발 프로세스란?

- 프로세스 : 사람을 업무에 할당해서 무언가를 만들도록 이끄는 일련의 동작이나 운영
- 소프트웨어 개발 프로세스 : 매일 매일 해야 하는 소프트웨어 명세 작성, 구현, 배포 및 유지보수를 수행할 수 있는 가이드

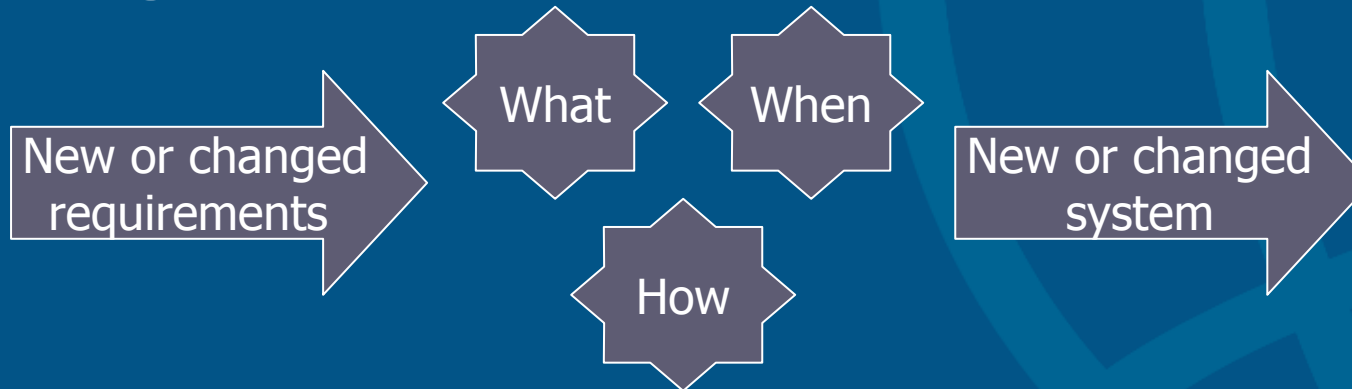


그림 1. SW Engineering Process (Who has to do WHAT, HOW, and WHEN)

소프트웨어 개발에서 프로세스의 장점

- 예측 가능한 소프트웨어 개발
 - ✓ 개발팀 및 PM이 완료된 작업, 진행중인 작업, 앞으로 발생할 작업에 대해서 잘 알게 됨
- 소프트웨어 개발의 반복과 측정 가능
 - ✓ 프로젝트의 방향성, 어떻게 완료되었는지 또는 얼마나 효율적이었는지 측정할 방법이 필요
- 일관된 비전 제시
 - ✓ 다른 팀원이 무엇을 하는지에 대한 이해를 증진시켜 의사 소통을 원활하게 만듦

소프트웨어 개발에서 프로세스의 장점

- 로드맵 수립 가능
 - ✓ 개발 과정에 대한 전체적인 로드맵 수립이 가능해짐

32444245304354

훌륭한 프로세스의 요건

- 이해 용이성
 - ✓ 프로세스에 의해 제공되는 정보가 이해 가능하다.
 - ✓ 명확하고 모호하지 않다.
 - ✓ 사용하는 사람들이 프로세스가 의도하는 바와 목적을 알 수 있다.
- 접근성
 - ✓ 상시로 접근 가능함
 - ✓ 가이드 중에서 원하는 유형을 빨리 찾을 수 있음
 - ✓ 어디에서 찾아야 하는지 금방 알 수 있음

훌륭한 프로세스의 요건

- 사용 용이성
 - ✓ 어떻게 적용해야 하는가에 대한 설명이 있다.
 - ✓ 적당히 상세하다.
 - ✓ 잘 정의되어 있으며 형식이 갖춰져 있다.
- 커스터마이징
 - ✓ 기술, 복잡도, 팀 구성원에 따라 맞춤이 가능하다.
 - ✓ 프로젝트의 유형에 따라 맞춤이 가능하다.
 - ✓ 기존 정보의 변경이 가능하다.
 - ✓ 새로운 정보의 추가가 가능하다.

32444245304354

RUP Part I

- Rational Unified Process (RUP)
 - ✓ 소프트웨어 엔지니어링 프로세스
 - ✓ 웹 기반
 - ✓ 검색 가능한 지식 데이터베이스
 - ✓ Rational Tool과의 연계
 - ✓ Online Mentor 기능
 - ✓ 여러 번의 반복 (Iteration) 을 거치며 각각의 반복은 요구사항 분석, 분석 & 설계, 구현, 및 테스트 & 평가 과정을 포함
 - ✓ 작업자, 단계별 산출물, 작업 흐름, 업무 활동 등의 정의

RUP Part II

- 작업자 (worker)
 - ✓ 개인이나 그룹의 행위나 책임
 - ✓ 역할을 의미
 - ✓ 산출물의 소유자
- 업무 활동 (activity)
 - ✓ 작업자의 역할에 따라 수행해야 하는 단위 업무
- 산출물 (artifact)
 - ✓ 프로세스에 의해 생성되고 수정되며, 사용되는 정보의 단위

RUP Part II

- 작업 흐름 (workflow)
 - ✓ 업무 활동의 순서와 작업자 간의 상호 작용의 기술

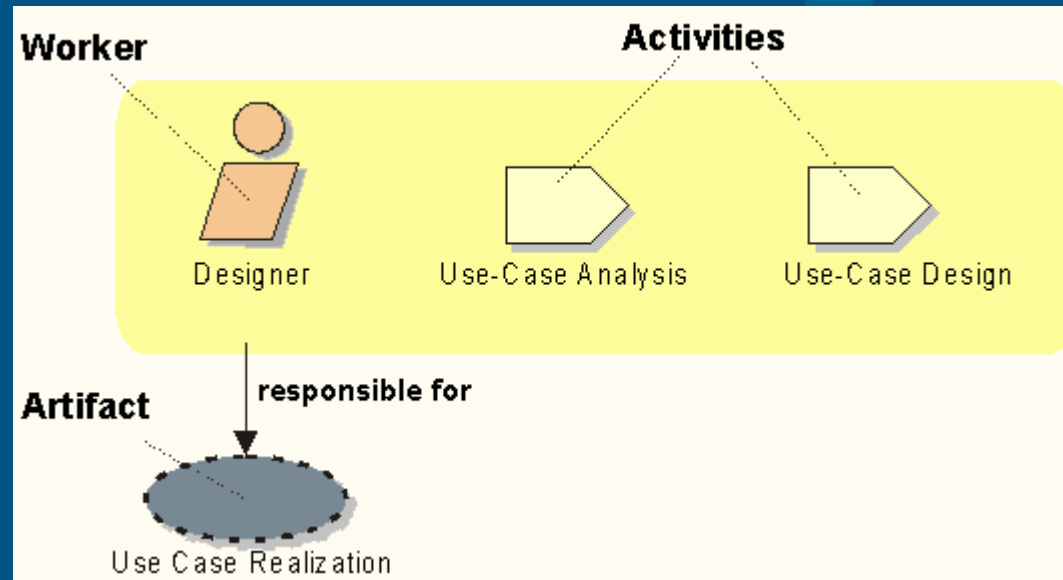


그림 2. 작업자, 작업 활동, 산출물의 관계

잘못된 인식들 – 계획 없는 작업

- 과거
 - ✓ 소규모의 프로젝트
 - ✓ 적은 인원
 - ✓ 기술적 복잡도가 낮음
- 현재
 - ✓ 중, 대규모의 프로젝트
 - ✓ 많은 인원
 - ✓ 다양한 기술이 뒤섞여 사용됨

32444245304354

잘못된 인식들 – 계획 없는 작업

- Visible한 개발 환경의 개선
 - ✓ 프로그래밍 언어는 점차 개정되고 있음
 - ✓ IDE와 같은 디버깅 툴 및 개발 보조 도구의 발달
 - ✓ 전반적으로 프로그래밍 기술의 발달
- Invisible한 개발 환경의 개선
 - ✓ 팀 단위의 소프트웨어 개발 방법
 - ✓ 개발팀 내 인원이 서로 다른 역할을 가짐
 - ✓ 의사 소통을 개선할 필요 발생
 - ✓ 공동의 목표를 달성하기 위한 비전의 필요성

잘못된 인식들 – 계획 없는 작업

32444245304354

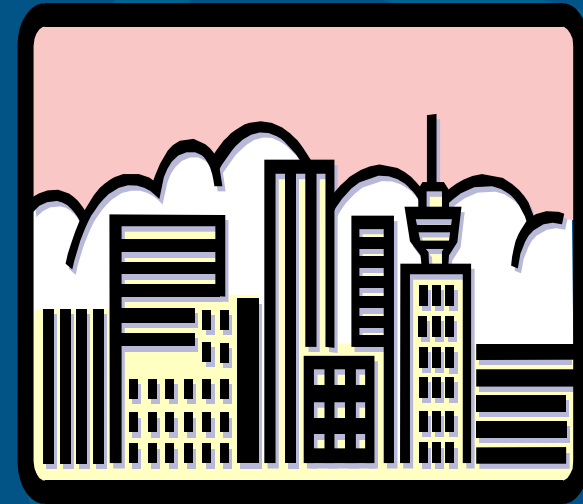


그림 3. 복잡도의 변경은 프로세스의 변경을 의미한다.

잘못된 인식들 - 학습 곡선

■ 학습 곡선

- ✓ 새로운 지식을 습득하기 위해 들인 노력 및 성취의 정도를 나타낸 그래프 (Learning Curve)
- ✓ 매우 완만한 곡선으로 시작해 일정 수준에 도달한 이후에는 급격한 경사 (성과) 를 보이는 것이 일반적임

■ 변화에 저항

- ✓ 학습 곡선은 어느 분야나 존재함
- ✓ 프로세스 도입을 주저하는 심리에는 학습 곡선에 대한 두려움이 내재된 것
- ✓ 학습 곡선은 있지만 그에 따른 ROI도 존재함

잘못된 인식들 – 프로세스는 매력적이지 않다.

- 완전성 vs 단순성
 - ✓ 완전하기만 하다면 복잡하다는 의견이 지배적임
 - ✓ 단순하기만 하다면 효율성에 의문을 제기함
- 적절히 조화
 - ✓ 프로세스는 완전해야 하지만 팀과 프로젝트의 크기에 맞게 조정되어야 함
 - ✓ 프로세스는 완전해야 하지만 간단해 보여야 하며, 이해, 인식, 접근이 용이해야 함

잘못된 인식들 – 프로세스는 매력적이지 않다.

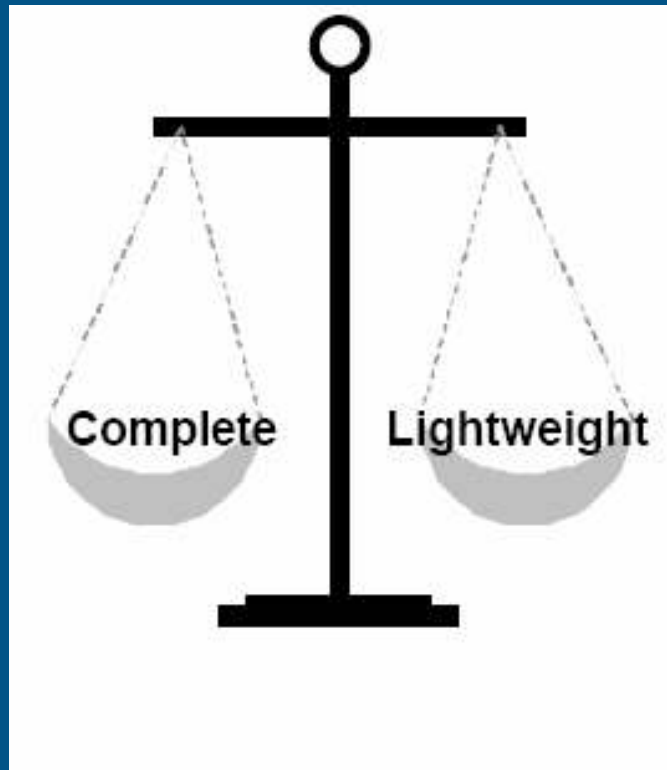


그림 4. 완전성인가? 단순성인가?

잘못된 인식들 – Iteration은 끊임없는 변경 사항을 양산 한다.

- Iteration의 기본 개념
 - ✓ 모든 문제에 대해서 한 번에 대응할 수 없음
 - ✓ 시작은 작은 부분부터 하며, 점차 다른 부분을 추가해 결국 원하는 결과를 얻을 때까지 반복
- Iteration 프로세스의 정의
 - ✓ 기본 계획 (Base-lined Plan) 을 가진 각각의 연속적인 작업
 - ✓ 각각의 작업에서 달성해야 하는 목표를 정의하고 있는 평가 측정 기준
 - ✓ **목표를 완료하면 반복 과정이 종료**
 - ✓ 코딩, 요구 사항 관리, 계획, 디자인 및 테스트 등

잘못된 인식들 – Iteration은 끊임없는 변경 사항을 양산 한다.

■ 문제

- ✓ Iteration의 각 단계를 거치면서 관련자들의 요구 사항을 계속 받아 들여 개발 과정이 완료되지 않음

■ 해결

- ✓ 계획 수립과 활동에 대한 통제를 내포
- ✓ 언제 시작하고, 끝내는지, 각 단계의 목표를 알 수 있음
- ✓ 추가적인 요구 사항에 대해서 관리될 수 있는 만큼만 허용 (적정성, 우선순위, 변경의 결과를 평가)
- ✓ 일련의 평가 후 허용/거부/연기

잘못된 인식들 – Iteration은 끊임없는 변경 사항을 양산 한다.



그림 5. 절대 끝나지 않는 Iteration –
Iteration 개발 방법의 잘못된 인식

잘못된 인식들 – 방법론자에 의해 고안된 이론이다.

■ 인식

- ✓ 방법론자에 의해서 고안된 이론
- ✓ 아무런 실제 프로젝트 경험이 없고 제품을 사용해 보지 않은 한 명이 만들어낸, 단지 “이론”에 불과

■ 사실

- ✓ 업계 전반의 성공 사례를 수집해서 고안
- ✓ 성공한 프로젝트에서 사용된 유용한 사례
- ✓ 다수의 프로젝트에서 재사용되는 사례

32444245304354

잘못된 인식들 – 방법론자에 의해 고안된 이론이다.



그림 6. 단지 한 사람 머리에서 나온 훌륭한
아이디어인가?

잘못된 인식들 – 프로세스는 단지 서류 작업만을 의미한다.

- 인식
 - ✓ 프로세스는 수많은 “문서”만 양산 한다.
- 사실
 - ✓ 내부 개발의 경우 공식적인 종이 문서는 거의 유지하지 않음
 - ✓ 문서를 유지하더라도 일부 (계획, 요구 사항) 이며 전자문서의 형태
 - ✓ 프로젝트의 성격에 따라 공식적인 문서 요구가 있을 수 있음 (강제 사항 또는 써드파티와의 작업)
 - ✓ 적절한 가이드를 제공하며 공식/비공식 프로젝트도 지원

잘못된 인식들 – 프로세스는 단지 서류 작업만을 의미한다.

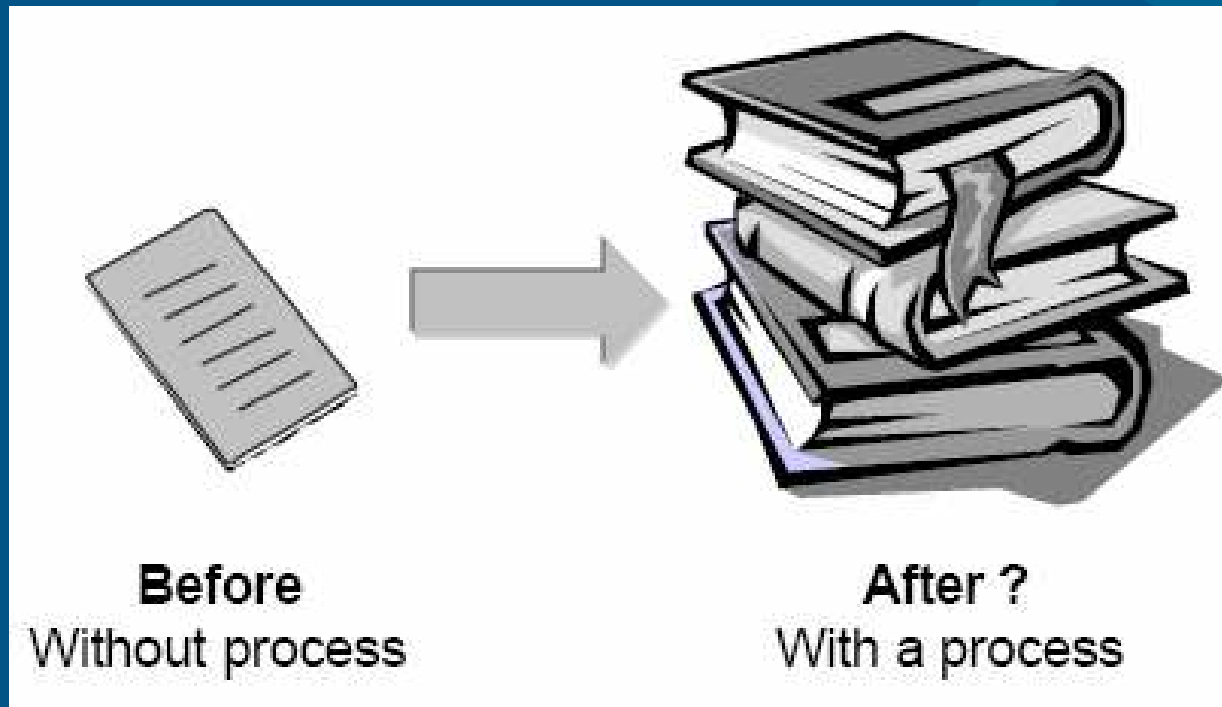


그림 7. 문서 작업만 강요한다는 인식

잘못된 인식들 – 프로세스는 창의력 을 말살한다.

■ 인식

- ✓ 프로세스는 강제적이며 유연하지 않다.
- ✓ 열정과 창의성을 감소시켜 자유를 억압한다.

■ 사실

- ✓ 프로세스가 창의성의 촉매 역할을 함[단, 자발적인 의사 소통 및 자신의 역할 대로 수행하는 경우]
- ✓ 프로세스는 요리책이 아니라 규칙 모음에 가깝다.
- ✓ 소프트웨어 팀의 인원은 프로세스가 정의한 틀 안에서 움직여야 하지만, 어디로 가야 할 지 결정할 자유는 있다.
- ✓ 프로세스는 개발 과정을 방해하지 않음

잘못된 인식들 – 프로세스는 창의력을 말살한다.

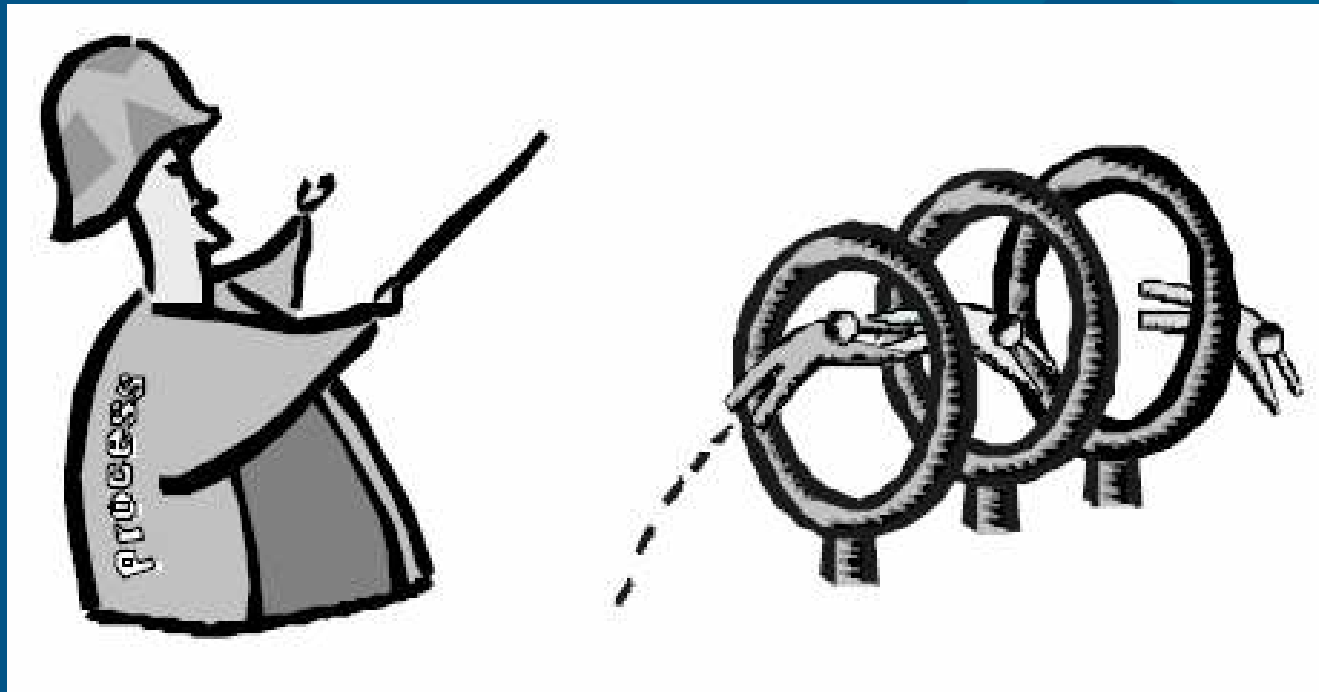


그림 8. 프로세스가 강압적인가?

잘못된 인식들 – 내 프로젝트에는 안 된다!

■ 상황

- ✓ 모두 프로세스의 장점에는 동의
- ✓ 프로세스로 유발되는 단점과 오해에 대해서도 동의
- ✓ 프로젝트에서 시간적인 압박으로 프로세스를 적용할 시간이 부족함

■ 사실

- ✓ 시간적인 압박은 반드시 부딪히게 되는 요소
- ✓ 프로그래머가 새로운 기술을 연구하는데 시간을 배정하는 것은 일반적이므로 어떻게 하면 팀 단위의 소프트웨어 개발을 잘 수행할 것인지에 대해서도 **시간을 배정 해야** 하는 것은 당연한 사실

잘못된 인식들 – 내 프로젝트에는 안 된다!

■ 상황

- ✓ 프로젝트의 규모가 크고 복잡하다.
- ✓ 중요한 프로젝트에는 도입하기가 어렵다.
- ✓ 프로젝트의 규모가 너무도 작아서 프로세스 도입의 장점이 없다.

■ 사실 (대안)

- ✓ 실제 이러한 주장은 변화를 거부하는 심리와 일맥상통한다.
- ✓ 경험을 쌓기 위해서 소규모로 점차 시작해 나간다.
- ✓ 프로세스의 일부를 도입해 본다.

32444245304354

잘못된 인식들 – 내 프로젝트에는 안 된다!

32444245304354



그림 9. 시간 압박 때문에 이번 프로젝트에서는 안 된다.

결론

- 소프트웨어 엔지니어링 프로세스의 장점
 - ✓ 프로젝트를 빠르고 안전하게 한다.
 - ✓ 프로젝트의 주요 요소들의 예측을 가능케 한다.
 - ✓ 프로젝트의 위험을 감소시킨다.
 - ✓ 팀 공통의 비전과 문화를 만든다.
- 결론
 - ✓ 소프트웨어 엔지니어링 프로세스에 대한 잘못된 인식이 존재한다.
 - ✓ 따라서, 프로세스 도입에 거부감이 존재함
 - ✓ 잘못된 인식을 계도하고 실제 의도와 장점을 부각하려는 노력이 필요

Reference

- Methods & Tools News Letter Fall 2002
- Software Development Process Misconceptions
by Nathalie Gaertner (www.rational.com)
- RUP의 기본 개념
- Jongseo Kyun (Rational Software Korea Ltd.)

32444245304354

마지막 페이지



감사합니다.