Visual Studio 를 이용한 애자일 프로그래밍

@Microsoft

Lean에게서 배우는 고품질 소프트웨어 개발 및 프로젝트 성공

강성재

http://vsts2010.net

Visual Studio Tools Evangelist

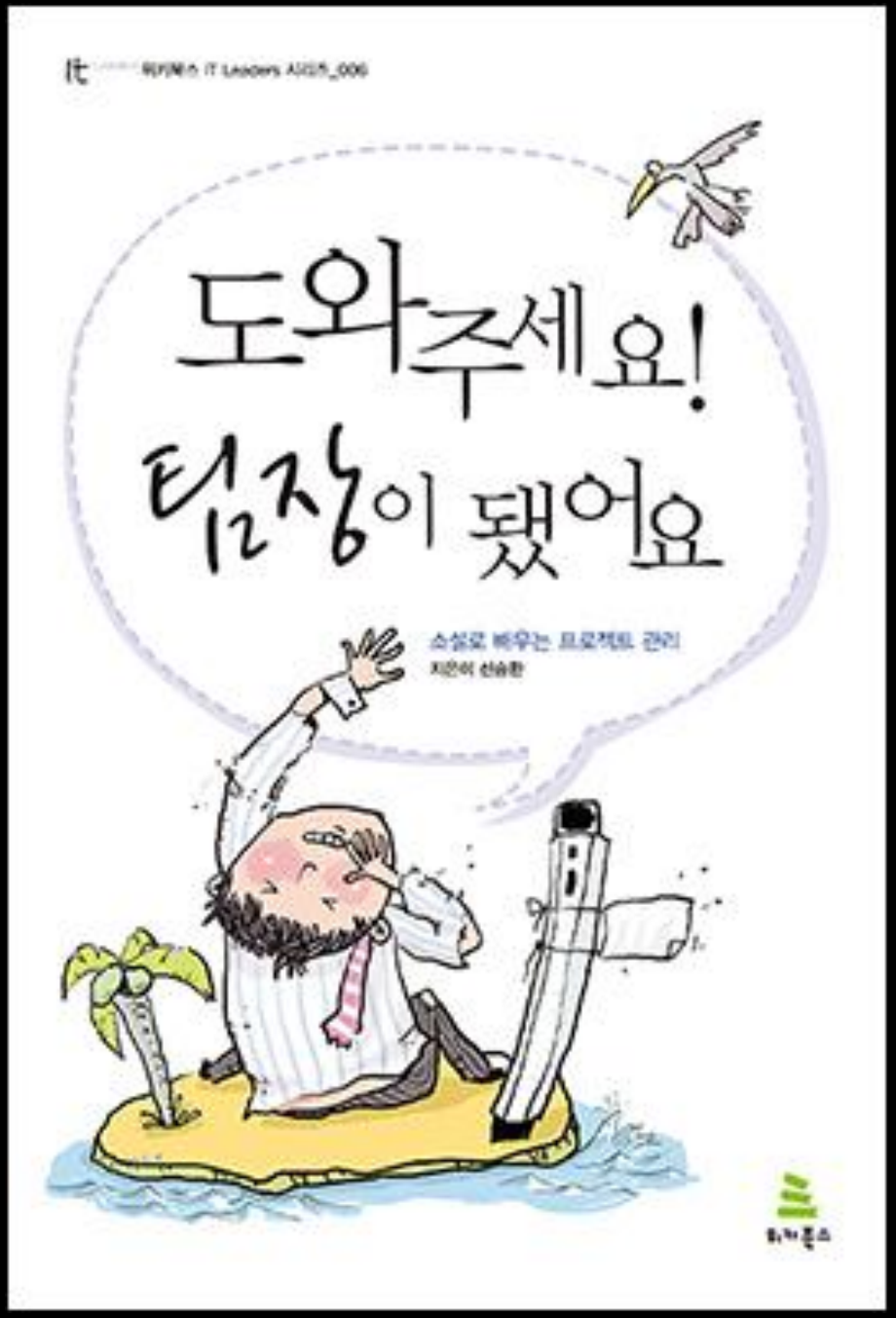
Developer & Platform Evangelism

Microsoft Korea

요구사항 정리

사용자의 요구

- 호환성
- 복잡하고 혼합된 응용프로그램
- 리치 인터넷 응용프로그램
- 데이터 중심이면서 동적
- 확장형 구조


IT Leaders AGILE_006
도와주세요!
팀장이 됐어요
소설로 배우는 프로젝트 관리
지은이 신승환
위키북스

개발 환경 변화의 효과

"안정적이고 효율적인 애플리케이션 운용을 통한 고객 만족 및 IT 투자 효과 극대화"

개발 생산성 향상

- 소스 코드 버전 및 변경 관리
- 이슈 및 버그 원인 추적
- 배포 이력 관리

효과적인 팀 협업

- 익숙한 개발 도구에서의 작업
- 도구 간의 정보 교환 용이

위험 및 이슈의 조기 해결

- 체계적인 개발 일정 관리
- 균등한 작업 분배

신속한 장애 대처

- 팀원 간의 원활한 커뮤니케이션
- 정보 접근 및 전달 시간 단축

효율적인 리소스 관리

- 개발 작업 진행 상황 모니터링
- 리소스 사용 현황 파악

애플리케이션 개발 프로세스

In the beginning...

TOYOTA

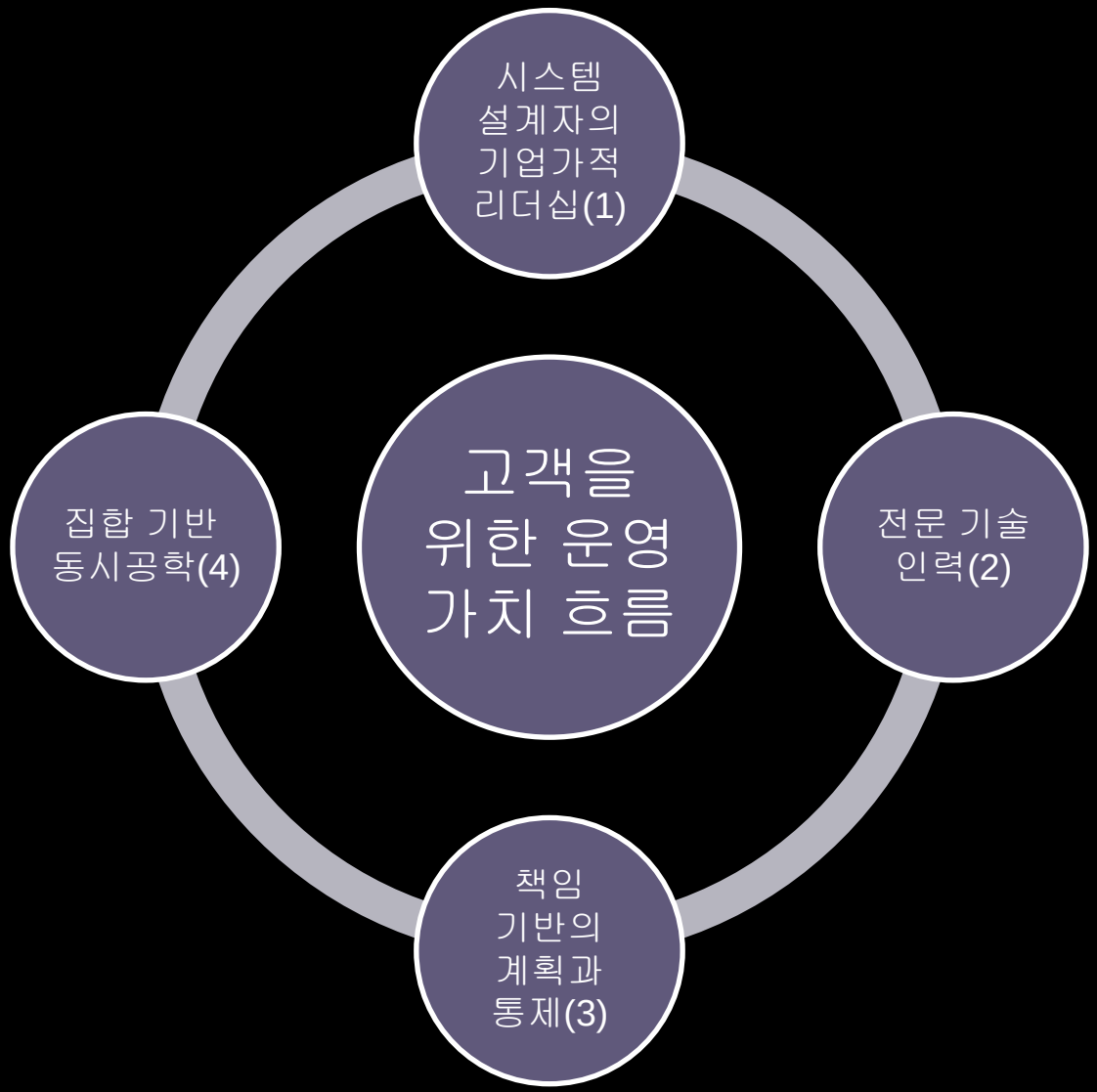
Lean Software Development = Agile Development + Lean Thinking

- 린 소프트웨어 개발은 모든 낭비와 지연까지 제거 하기 위해 노력
- 애자일 소프트웨어 실천을 확장하고, 발전하고, 배가 시킬 수 있는 환경을 확립
- 도구와 기법, 방법들을 조직의 고유하고 특별한 상황과 역량에 맞게 고쳐서 적용할 수 있게 하는 원칙을 제공

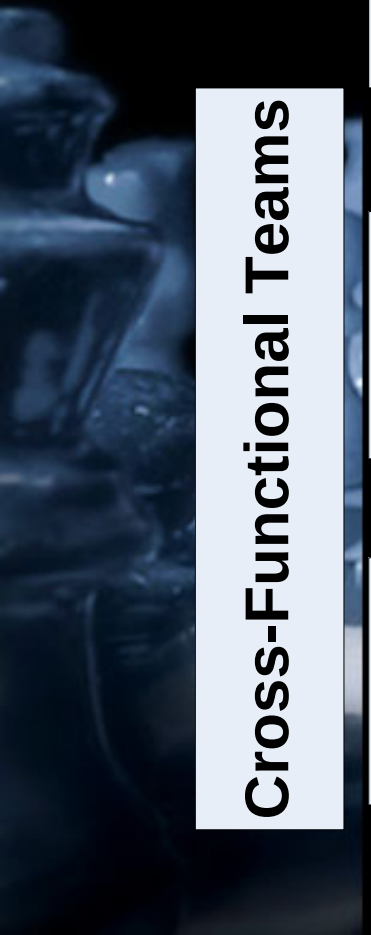
낭비의 정의 (3 M’s, 3무)

- *Muda* 無駄
  - 낭비(waste)
- *Mura* 無斑
  - 고르지 못함, 스트레스(stress)
- *Muri* 無理
  - 무리, 부하(loading)

린 개발 방식(지식기반 공학)



Lean – View



Lean Principles

Cross-Functional Teams

Project Management / SW Development Process

Scrum, TSP/PSP, SDLC (Waterfall, Agile, RUP)

Technical Practices

**Extreme Development, Design Patterns,
Test-Driven-Development, QA/Test, Agile Analysis**

Practices

원칙1: 낭비를 제거하라

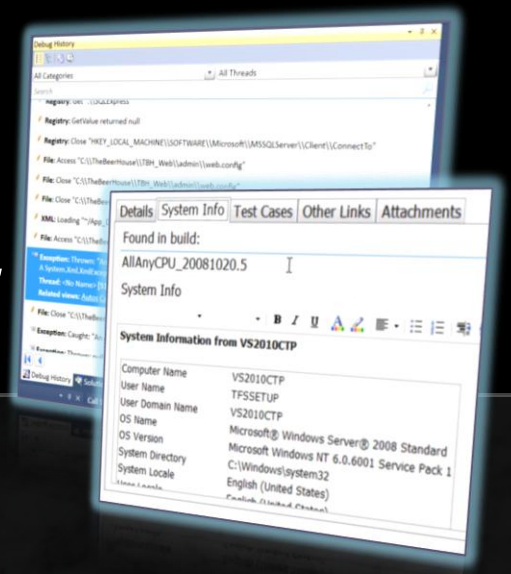
- 낭비의 제거 → 낭비를 인지 → 가치를 더하지 못하는 어떤 것도 낭비 → 가치가 정말 무엇인지 감지 ex) 구글
- 가치의 이해 → 현실에서 낭비를 보는 능력을 개발 → 낭비는 고객에게 적시 적소에 전달하는 것을 방해하는 모든 것
- ex) 제조 분야에서 재고는 낭비, 소프트웨어 개발에서 재고는 미완성
- 작업(Partially done work)
- 소프트웨어 개발의 대표적 낭비 "혼란", 요구사항 혼란, 가외 기능(extra feature)
- 잘못된 통념: 스펙 조기 확정이 낭비를 줄인다.

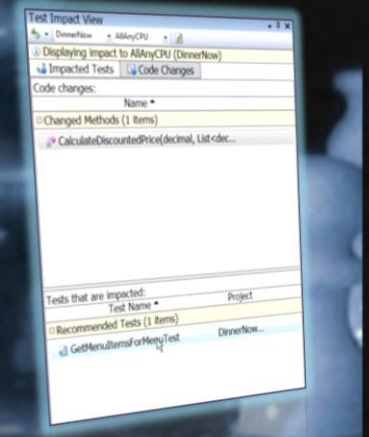
원칙2: 품질을 내재화하라

- 결함 대기열
- 검사(inspection), 테스트 주도 개발(test-driven development)
- 생산성 급등
- 잘못된 통념: 테스팅의 역할은 결함을 발견하는 것이다.

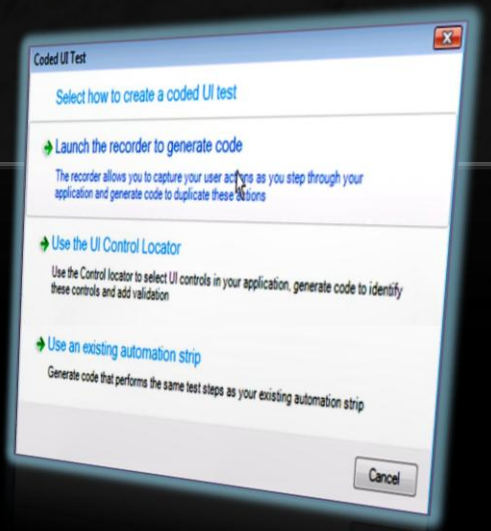
디버깅 시간 감소

- IntelliTrace™
  - *존재하는 버그들에 대하여 "재현 불가능한 버그" 히스토리를 만들어 줍니다.*





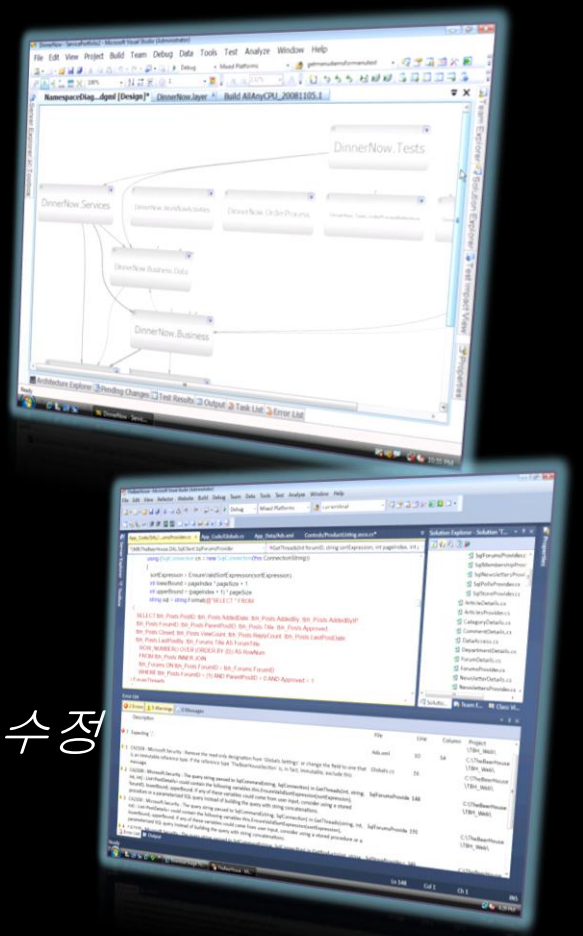
- Test Impact 분석과 코드 커버리지
  - *올바른 테스트 진행*
  - *코드단위의 테스트가 가능하여 코드가 테스트 되었는지에 대한 확인 가능*



- 코딩된 UI 테스트
  - *UI 요소들에 대한 자동 회귀 테스트로 버그들을 미리 찾을 수 있음*

코딩, 디버깅에서의 낭비 제거

- 종속성 다이어그램
  - *코드의 종속성을 비주얼 하고 변경을 이해*

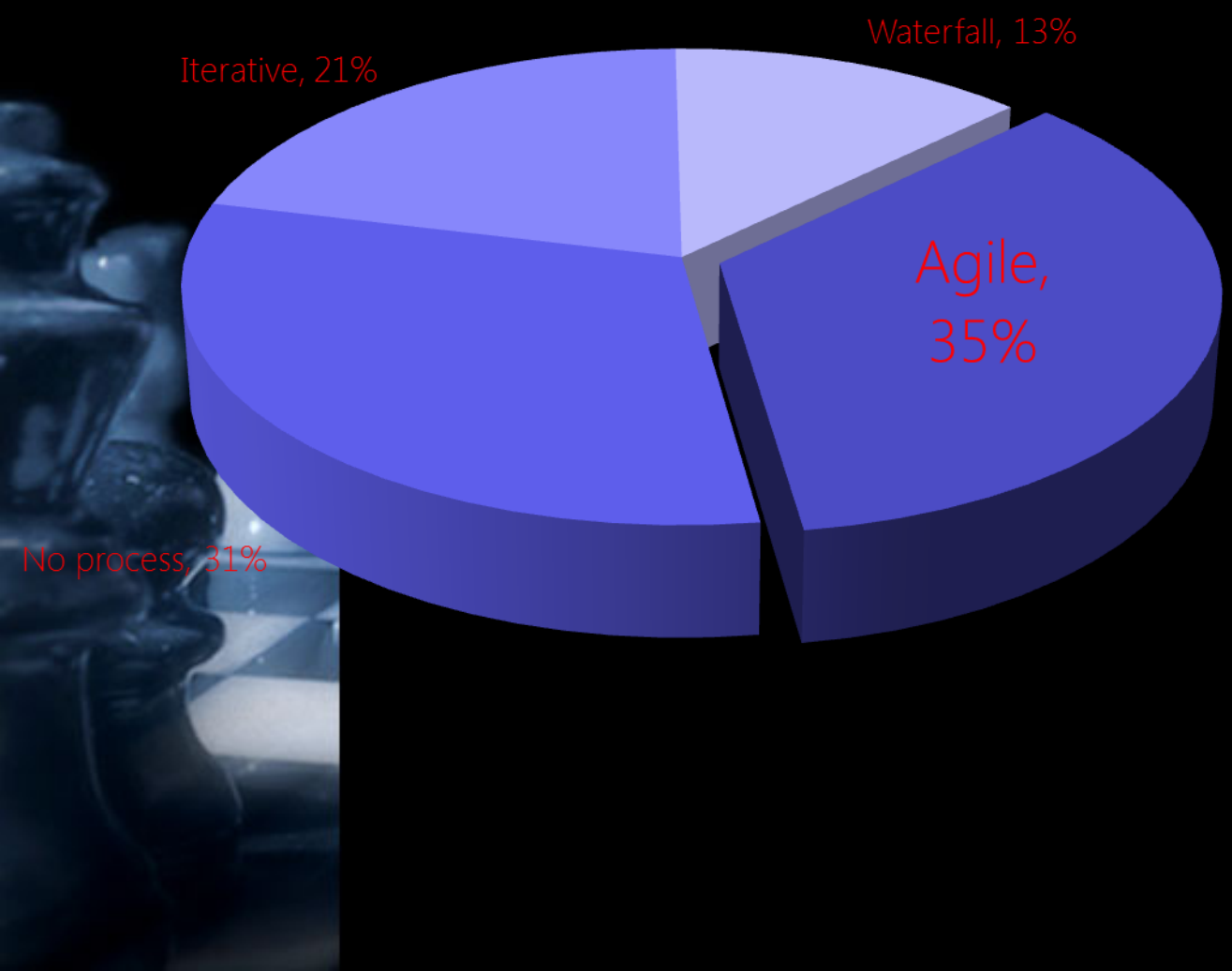


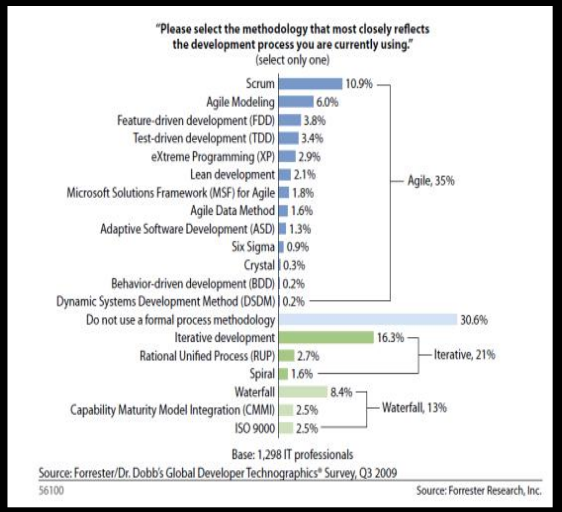
- 코드 분석과 코드 매트릭
  - *잠재적 문제를 그것이 발생하기 전에 발견하고 수정*

원칙3: 지식을 창출하라

- 소프트웨어 개발은 지식을 창출해가는 프로세스
- 폭포수 모델은 '요구사항'이라는 형태로 지식이 코딩과 분리되어 존재
- '표준 프로세스'를 찾는 과정에서 일어나는 최악의 상황은 프로세스를 문서 속에 가둬 버리는 상황이다.
- 잘못된 통념: 예측을 해야 예측이 가능해진다

개발 프로세스





Source: Forrester Reasearch, Inc.

하버드 비즈니스 스쿨의 앨런 맥코맥

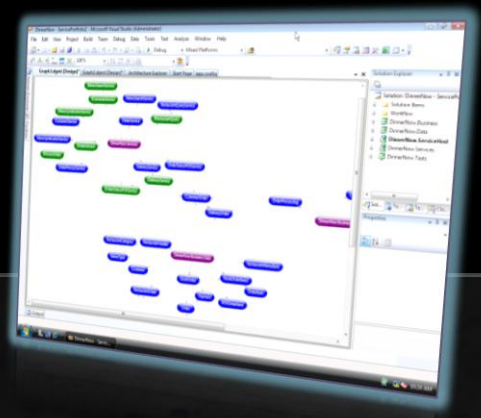
- 성공적인 소프트웨어 개발을 위한 실천법
  1. 고객의 평가와 피드백을 위한 최소 기능 집합의 조기 출시
  2. 일일 빌드와 통합 테스트를 통한 빠른 피드백
  3. 좋은 판단을 할 수 있는 경험과 직관을 갖춘 팀과 리더
  4. 새로운 기능을 쉽게 추가 할 수 있는 모듈화된 아키텍처

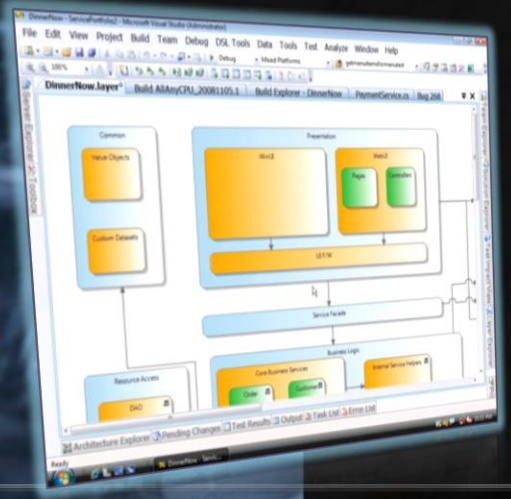
원칙4: 확정을 늦춰라

- 잘못된 통념: 계획을 세우는 것은 확정하는 것이다
- 아이젠하워
  - "나는 전투를 준비함에 있어 계획(Plan)은 언제나 쓸모가 없다는 것을 발견했다. 그러나 계획을 세우는 것(Planning)은 필수불가결하다."
- 계획은 아주 쉽게 변하고, 탄성(elasticity)이 중요하다.

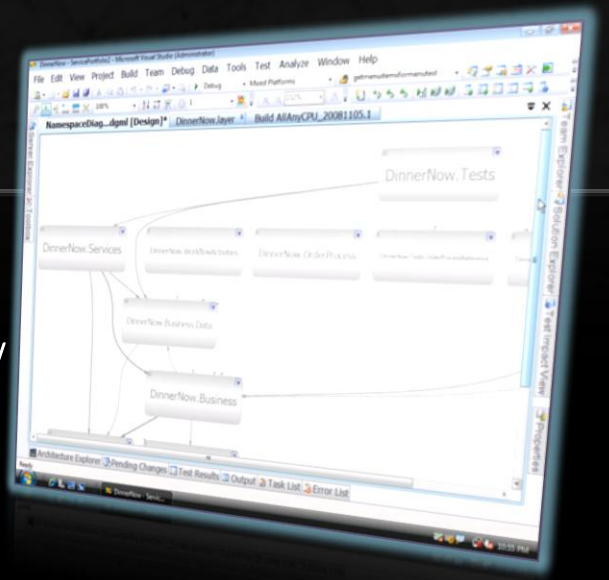
필수요소로서의 아키텍처

- 코드와 연동된 모델링
  - *표준 UML 다이어그램을 이용하여 모델과 코드를 자유롭게 넘나들며 애플리케이션을 표현*





- 아키텍처 검증
  - *코드가 의도한 설계와 맞는지를 확인*



- 아키텍처 Explorer
  - *시스템이 어떻게 작동하는지 이해하고 현재의 코드를 시각화하여 어떻게 시스템과 연결되는지 확인*

원칙5: 빨리 인도하라

- 잘못된 통념:
  서두르는 것은
  낭비를 낳는다

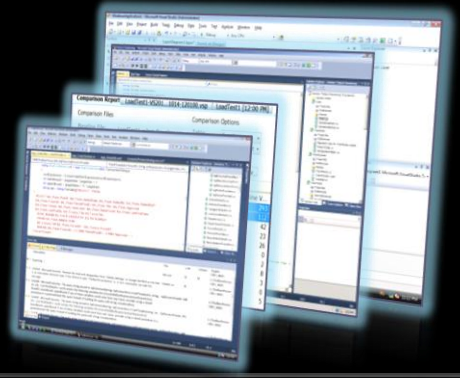
원칙6: 사람을 존중하라

- 잘못된 통념: 유일한 최선의 방법(One best way)이 존재한다

원칙7: 전체를 최적화하라

- 잘못된 통념:
  부분으로 나누어
  최적화하라

통합된 개발과정



- 개발 사이클 전반에 걸친 이슈들을 빨리 해결 - *데 통합된 개발도구와 레퍼지토리 공유를 통해 데이터가 끊김없이 움직일수 있도록 함으로서 비즈니스 요구와 기술적 이슈를 해결하는데 도움을 줌*

Microsoft SharePoint

Windows 7

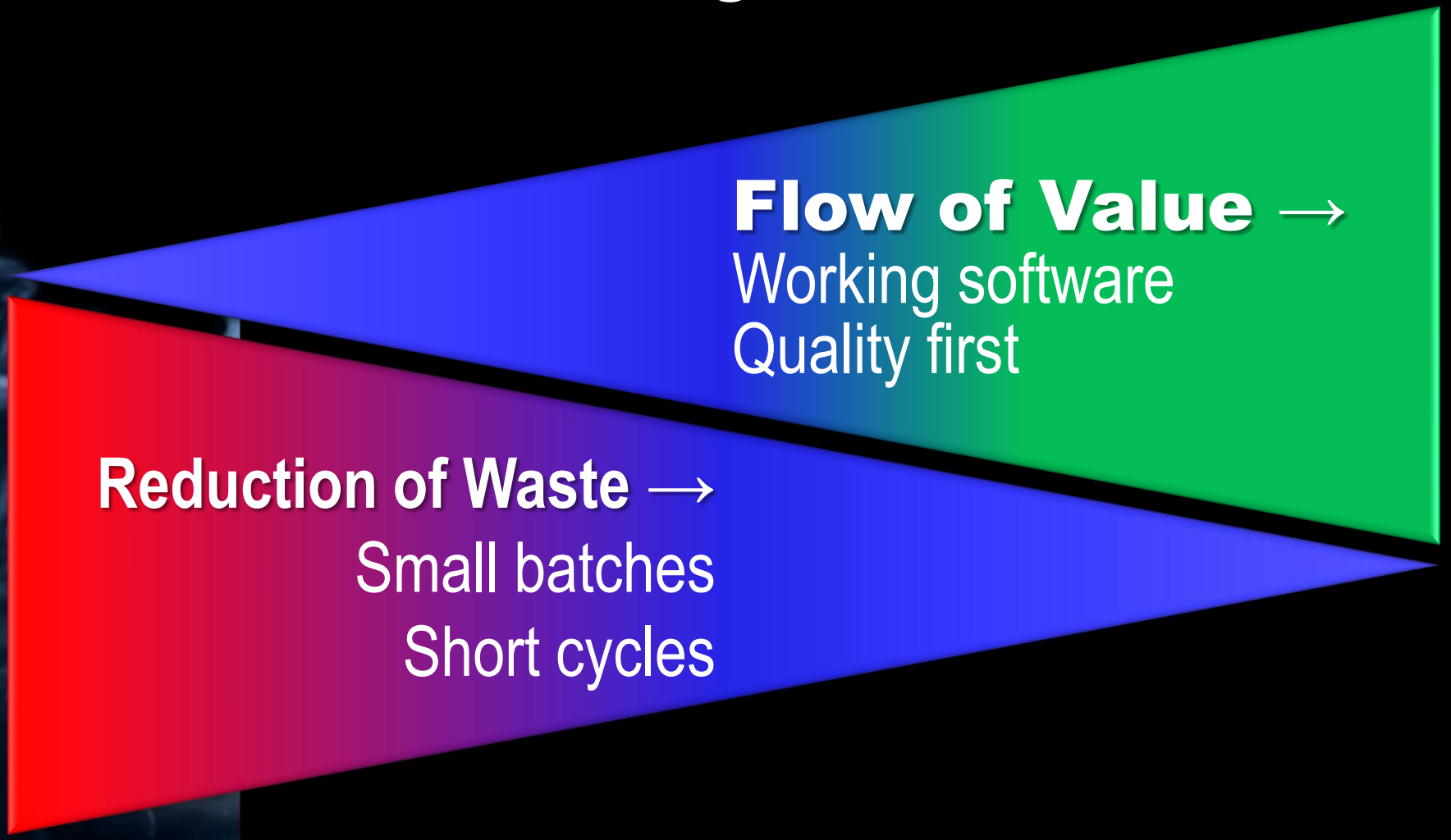
Windows Azure

- 최신 플랫폼 지원
  - *Windows 7, SharePoint, Cloud development, Silverlight, WPF 등*

- Deployment Excellence
  - *팀전체의 세부 정보를 이용하여 프로젝트 완료시점을 파악*
  - *MS 빌드, Click Once 와 클라우드 Deployment 옵션으로 효율적으로 개발한 솔루션을 deploy 할 수 있음*

Lean Application Lifecycle Management

두 개의 원칙



감사합니다.

sungjaek@microsoft.com

{ http://vsts2010.net }