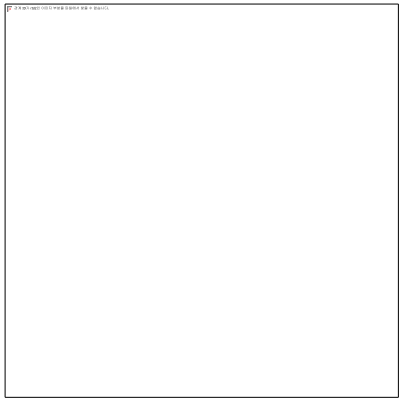
SI이노베이션

IT퍼스트 시대의 개발 패러다임

소속 Mamezou
이름 정도현

자기소개



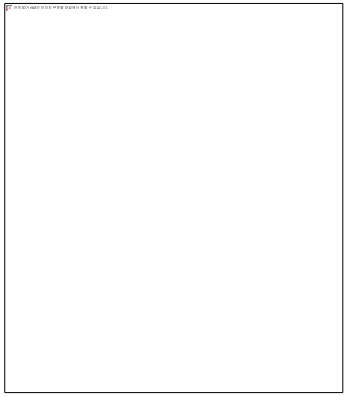
- 정도현
- 일본 豆蔵(Mamezou)
- IT컨설턴트/아키텍트
- 블로그 moreagile.net운영
- InfoQ Japan 필진, Qcon Tokyo 2015 운영위원
- 나는프로그래머다 MC(정개발)
- Facebook ServerSideArchitectGroup 운영진
- 자바, C++을 주로 사용하며 ruby, scala, python, javascript를 업무와 개인용도로 사용. 최근에는 Kotlin과 Go에 관심이 많음.

마메조(まめ蔵)

이미지 출처 http://iconosquare.com/p/876074776160816057_254934017

근무처소개 – Mamezou

- 주식회사 Mamezou(豆蔵)
- 2000년5월 설립
- 사명은 콩(자바빈즈)을 다루는 기능인이라는 뜻
- 소프트웨어 엔지니어링 기술을 핵심경쟁력으로 삼는 컨설팅 회사
- 일본 국내 최고수준의 소프트웨어엔지니어 집단
- InfoQ 일본 파트너
- QCon Tokyo 주관사



프로그래머를 위한 팟케스트 나는 프로그래머다

- iamprogrammer.io
- 임백준(임작가),
  김호광(데니스), 정도현(정개발)
- '개발자들을 위한 유쾌한 팟캐스트'를 모토로 언어, 패러다임, 개발환경등을 주요 주제로 하여 매주 60분 분량 방송
- 이번주 금요일 넥슨 아레나에서 공개방송 및 컨퍼런스 개최!
  http://onoffmix.com/event/55258

Part1

Project vs Product

소프트웨어 개발을 어떤 관점으로 바라 볼 것인가?

비즈니스 환경의 변화

기업에 있어서 IT의 위상변화

- 단순히 서류로 진행하던 작업에 대한 자동화 수준
- 산업의 고도화로 인해 점차 핵심분야로 이동
- 핵심비즈니스 영역에 포함되거나 아예 핵심비즈니스 자체가 됨

산업 구조의 혁신

- 정부 – 전자정부, 전자투표, 데이터활용
- 교통, 운수 - 우버, 카카오택시
- 미디어, 방송 – 넥플릭스, 가디언, 일본경제신문
- 에너지 – 홈그리드
- 자동차 – 자동운전
- 제약,의료 – 신약개발
- 금융, 보험업 – 핀테크
- 유통, 소매업 – 빅데이터
- 제조업 – 팩토리 5.0
- 교육 – 칸아카데미, 코세라

소프트웨어 개발 환경의 변화

소프트웨어 개발의 혁신

- 적은인원으로 대규모 개발이 가능해짐
  - 오픈소스활용
  - 자동화 툴의 발전
  - 클라우드컴퓨팅의 등장
- DevOps와 Agile개발의 등장
  - 빠른 구현에서 배포에 이르는 사이클을 안정적으로 컨트롤 하는 기술과 프로세스가 등장

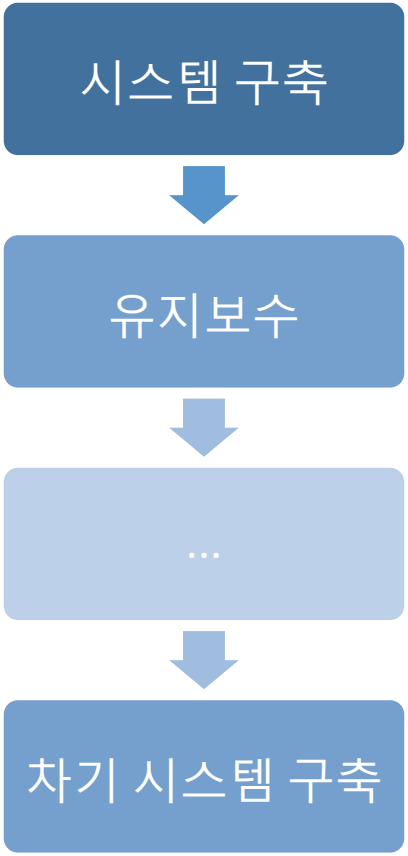
프로젝트란?

- 프로젝트는 계획된 시간 안에 목표로 하는 범위를 완수해야 하는 작업
- 프로젝트는 다음과 같은 특징을 갖는 일회적 작업
  - 시작, 종료일이 정해져 있다.
  - 목적 혹은 수행할 작업의 범위가 명확히 명시되어 있다.
  - 예산이 미리 책정되어 있다.
  - 일반적으로 프로젝트가 종료되면 해산하는 임시 조직이 수행한다.
  - 해결할 업무가 있다.
  - 일련의 예상되는 효과, 생산품, 결과, 산출물이 정해져 있다.
  - 잘 정의된 역할 할당에 따라 선정된 인력들이 업무를 수행한다.

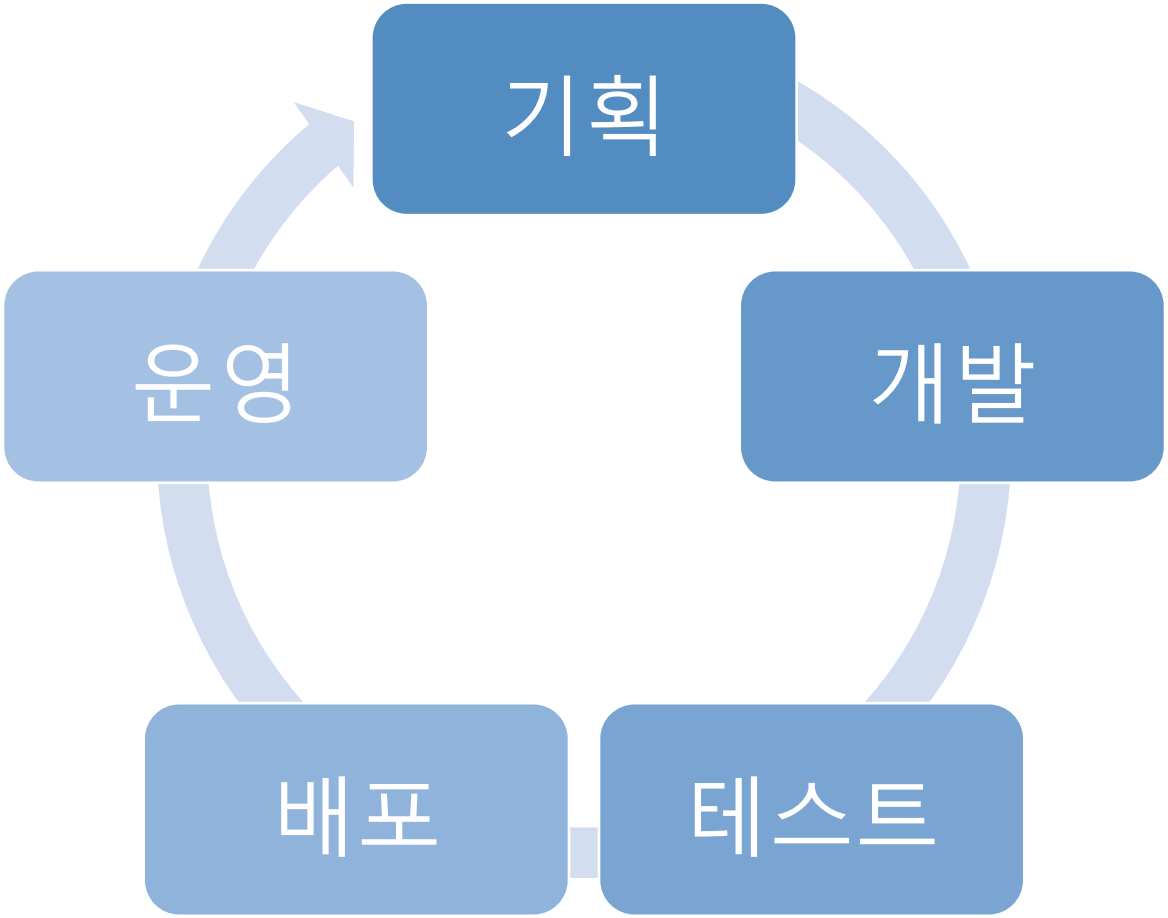
프로덕트로서 IT시스템

- IT시스템을 코어 비즈니스의 일부 또는 그 자체로서 인식
- 한시적을 발생하는 프로젝트의 개념이 아닌 지속적으로 변화를 관리해나가는 프로덕트로서 접근해야 함

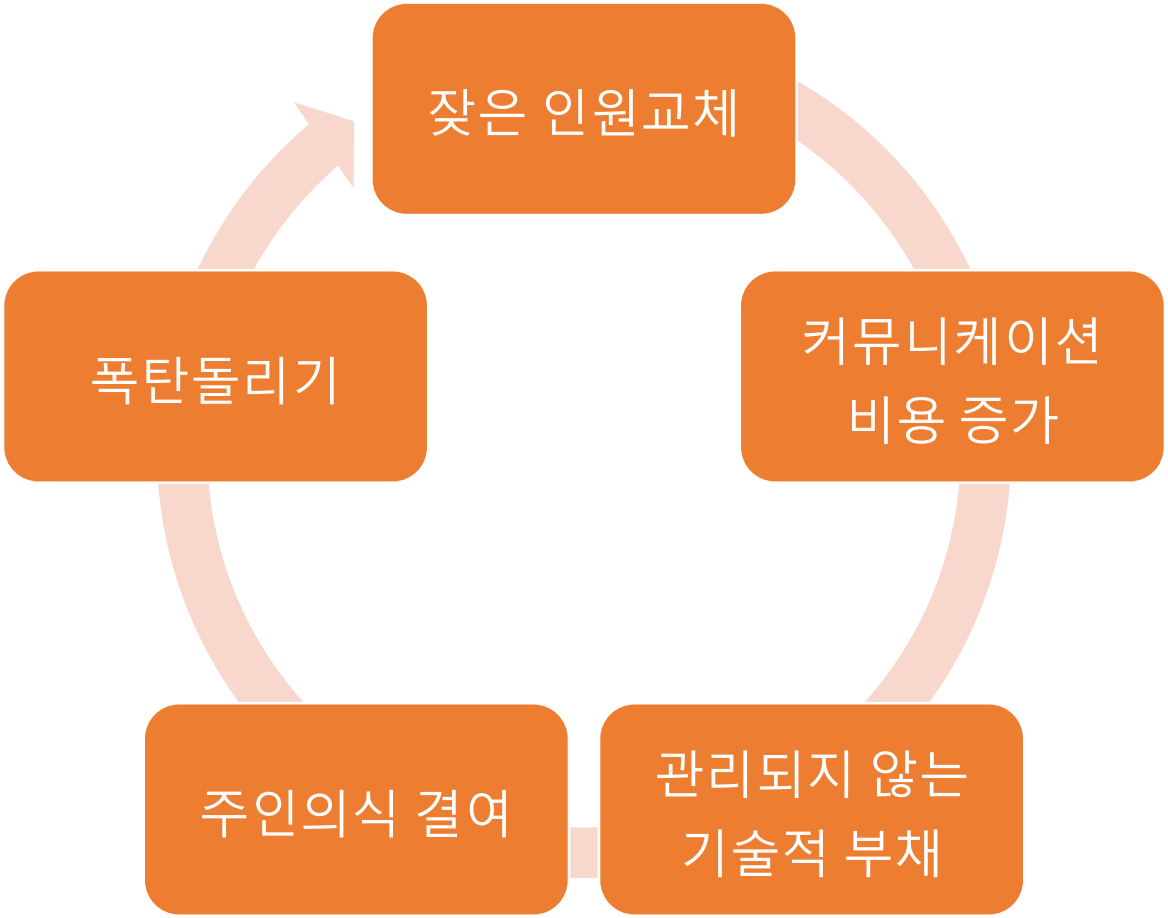
프로젝트 프로세스



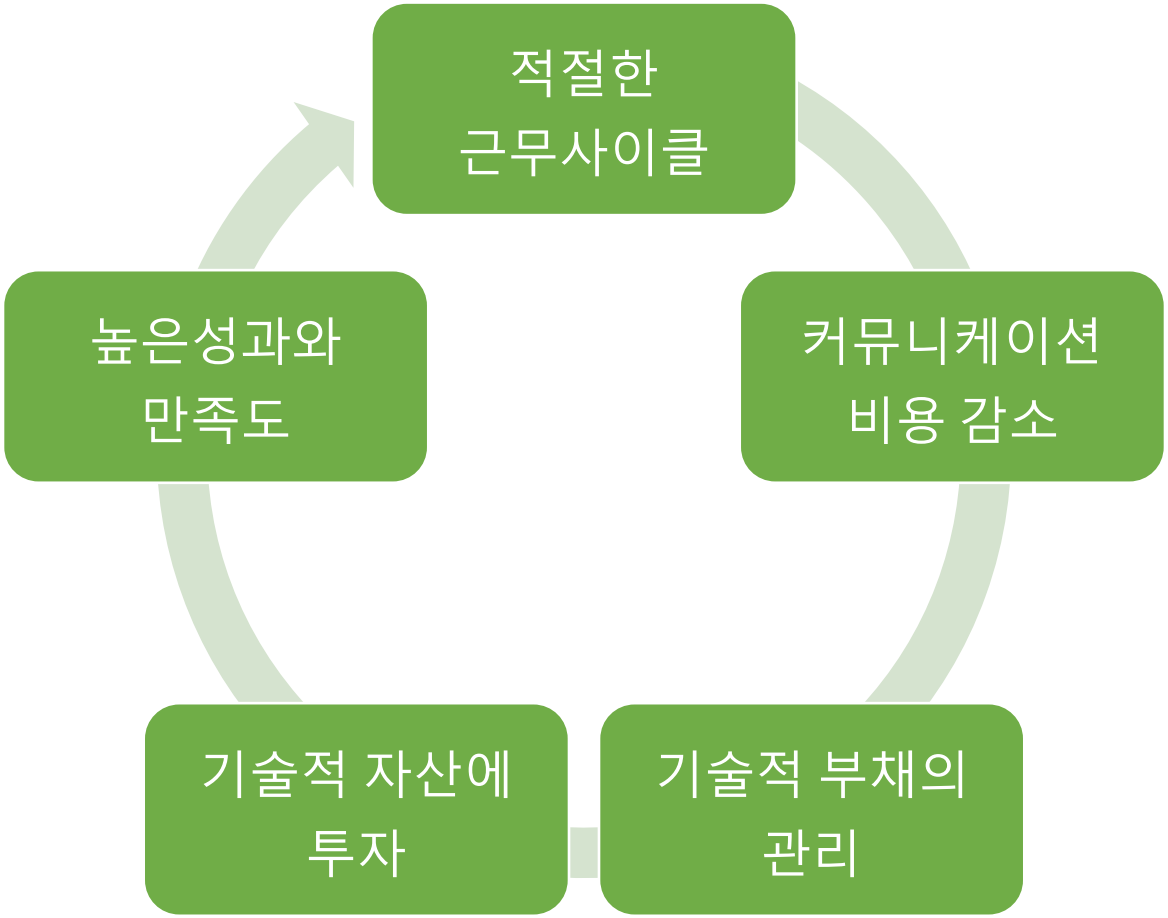
프로덕트 사이클



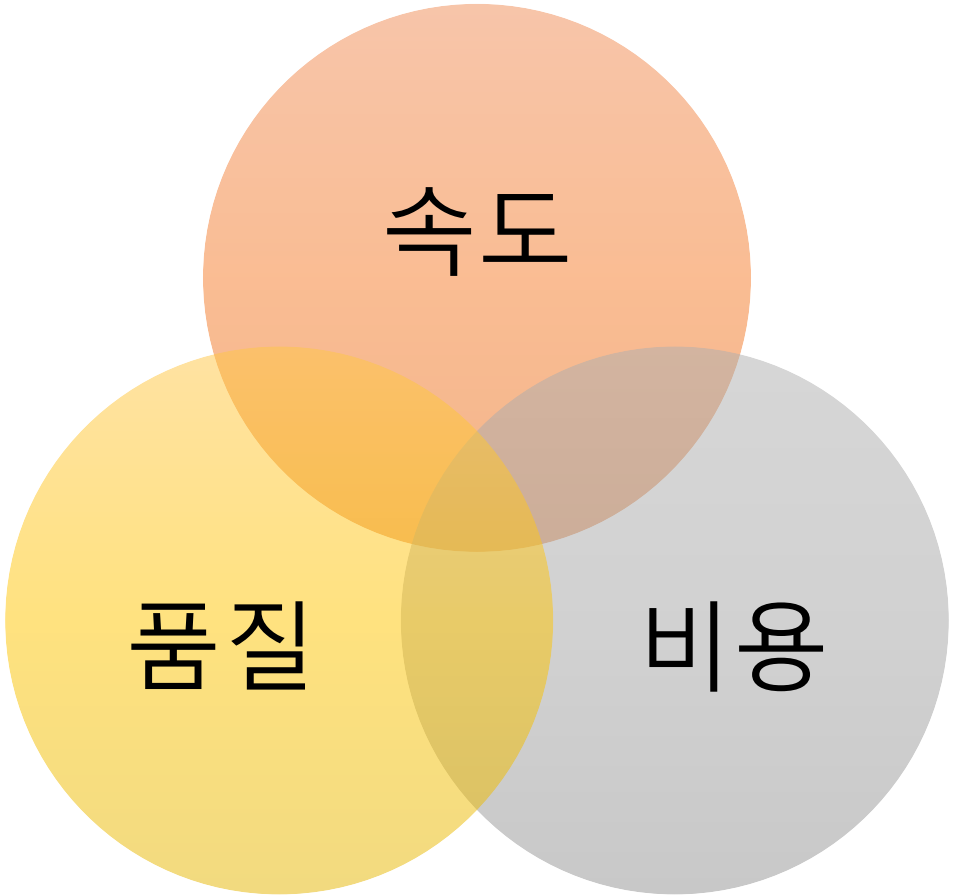
개발의 악순환



개발의 선순환



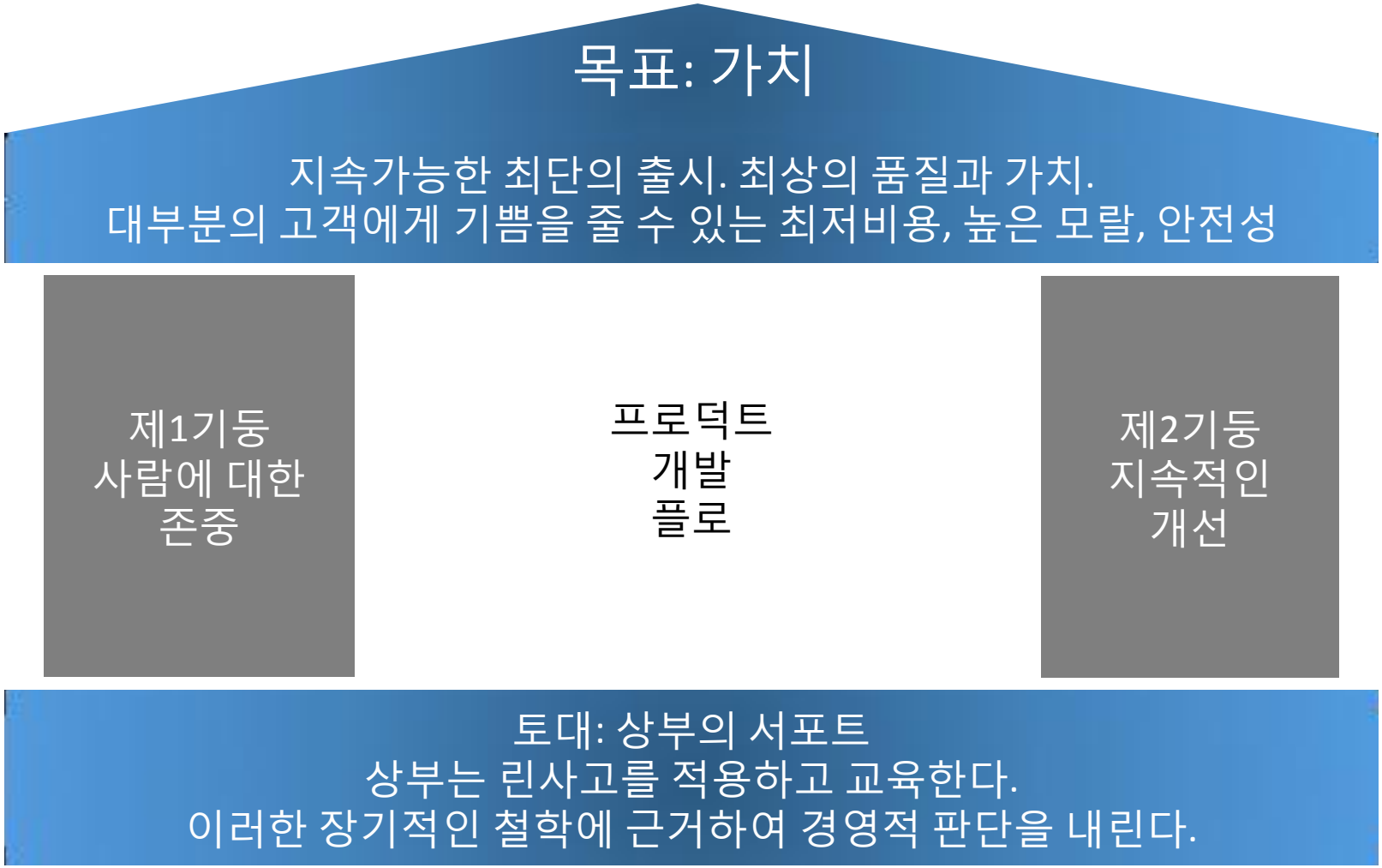
사내개발로의 전환이 필요한 경우는?



Part2

사내개발을 시작하기 위한 기반 요소들

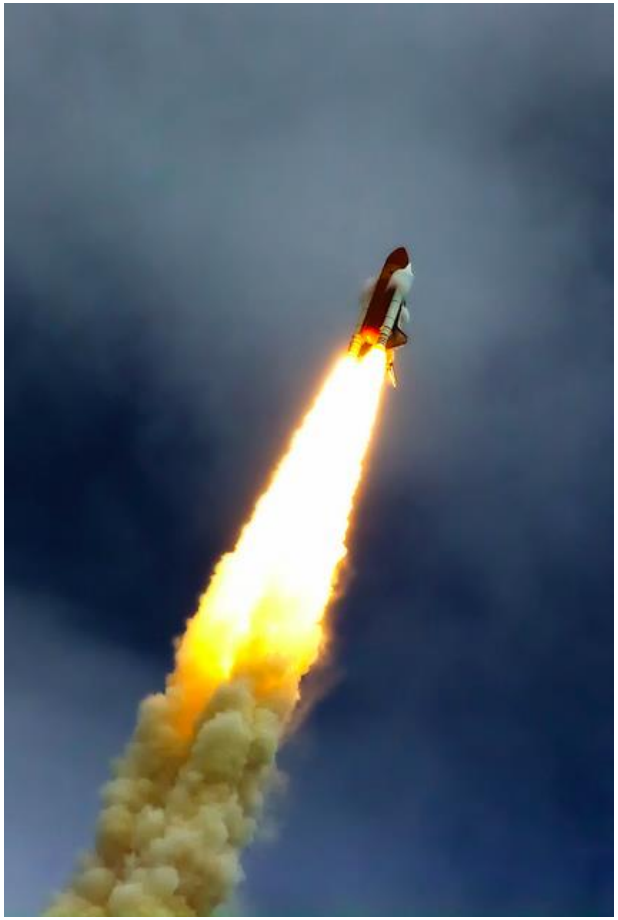
The House of Lean



출전 Leffingwell, “Agile Software Requirements”, 2010

문제는 기술이 아니다!!!

- 하던대로에서 벗어나기
  - 중력에서 탈출하기 위해서는 초속 11.19km의 속력이 필요함



문제는 기술이 아니다

- 상향식과 하향식 접근이 모두 필요
  - 실무자와 경영진이 성공에 대한 이미지를 공유하는것이 핵심
- 조직의 형태
  - 기존 개발부서의 체제변경 - 사내 벤처형태
  - 채용에서 평가, 보상체계 까지 조직 운영을 완전히 재구성
- 점진적 접근
  - 프론트엔드부터
  - 새로 만드는 부분부터
- 끊임없는 개선
  - 개선을 위한 팀을 항시운영
  - 사내의 각 부문과 끊임없는 정보공유

소프트웨어는 그것을 만든 조직의 모습을 투영한다 – 콘웨이의 법칙

출전 Exploring Scrum: The Fundamentals.

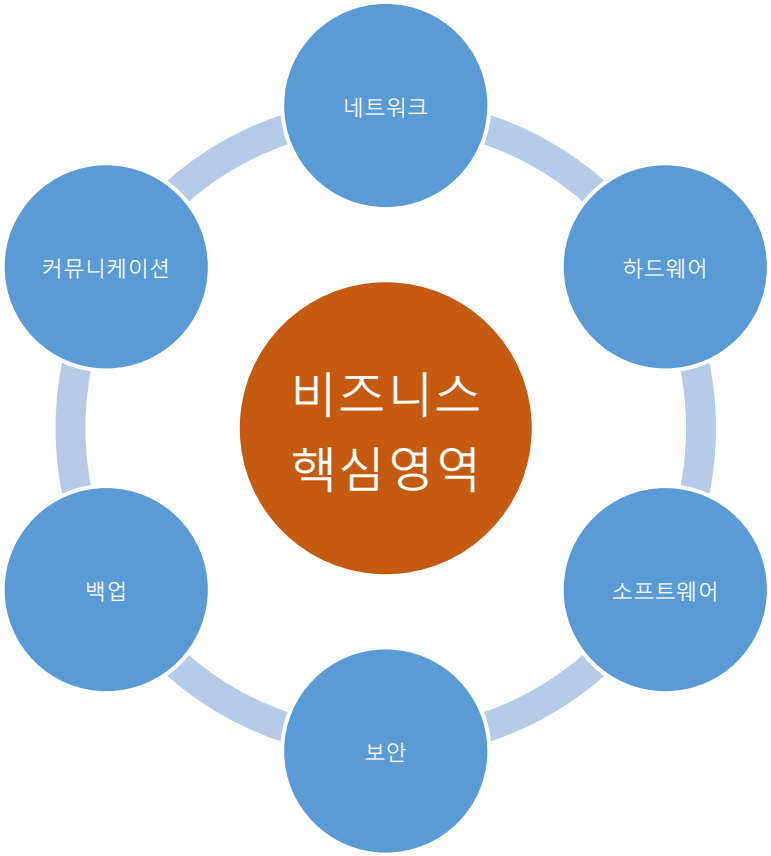
기본에 충실할것

- 목표를 명확히
  - 범위를 한정
  - 우선순위를 정한다
  - 품질은 타협의 대상이 아님
- 반복개발
  - 소프트웨어 공학에 기반
  - 알맞는 개발프로세스의 체화
  - 적절한 툴 도입(CI,TEST,SCM)
  - 리듬을 만들어나가는것이 중요
- 커뮤니케이션
  - 회의는 짧고 효율적으로, 아젠다는 충실하게
  - 적절한 툴 도입 (Slack, Github/Gitlab)
  - 타성에 젖지 않도록
  - 목표에 대한 이미지 공유

맨토링

- 외부 전문가에 의한 자문
- 경험부족을 커버. 단, 만능이 아님!
  - 성공에 대한 경험 – 패턴
  - 실패에 대한 경험 – 안티패턴
- 소프트스킬
  - 커뮤니케이션,팀빌딩 전략, 개인의 성장을 조직의 성장으로연결
- 하드스킬
  - 프로세스, 기술
- 문제를 올바르게 파악하는데 도움일 줄 수 있음

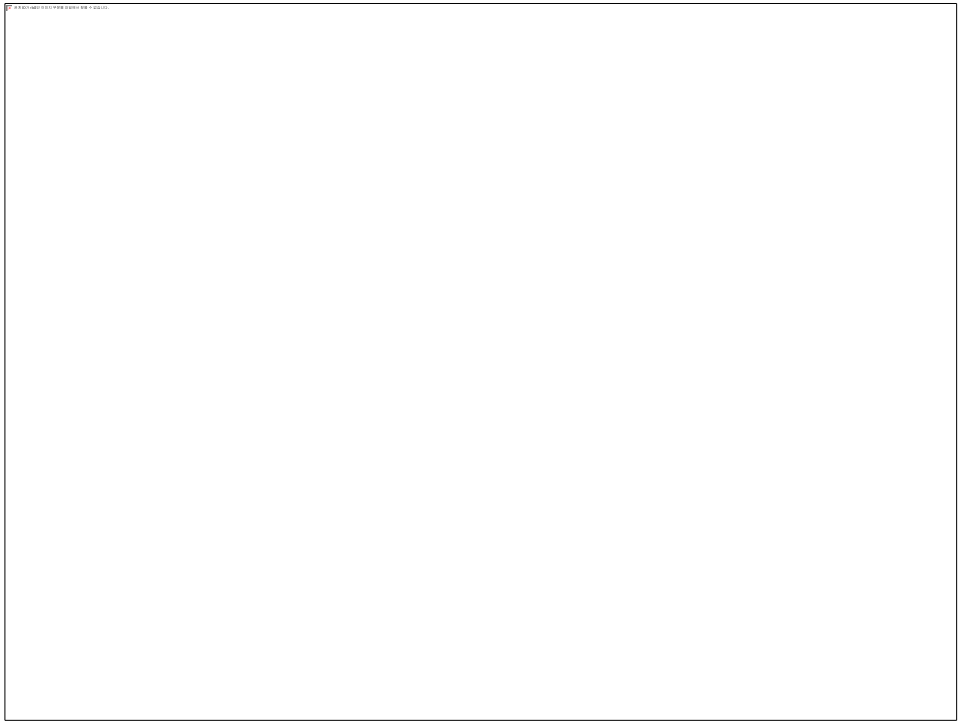
핵심에 집중하라



Part3

구체적인 전략수립

합숙워크샵



- ~~부어라-마셔라~~ 비전의 공유

출저 https://speakerdeck.com/yosukesuzuki/nikkei-web-development-2015

The Twelve-Factor App

- Adam Wiggins가 Martin Fowler의 책, Patterns of Enterprise Application Architecture과 Refactoring에서 영감을 받아 작성
- 응용프로그램의 시간이 지남에 따른 유기적인 성장, 애플리케이션의 코드베이스에서 작업하는 개발자들 간의 협업, 소프트웨어가 낡는 것에 의한 비용을 피하는 법에 집중하여 애플리케이션 개발에 대한 이상적인 방법을 찾고자 하였음.
- 응용프로그램과 서비스제작에 필요한 실천적이고 구체적인 소프트웨어 엔지니어링 기반을 제안

The Twelve-Factor App

- **I. 코드베이스**

버전 관리되는 하나의 코드베이스와 다양한 배포

- **II. 종속성**

명시적으로 선언되고 분리된 종속성

- **III. 설정**

환경(environment)에 저장된 설정

- **IV. 백엔드 서비스**

백엔드 서비스를 연결된 리소스로 취급

- **V. 빌드, 릴리즈, 실행**

철저하게 분리된 빌드와 실행 단계

- **VI. 프로세스**

애플리케이션을 하나 혹은 여러개의
무상태(stateless) 프로세스로 실행

- **VII. 포트 바인딩**

포트 바인딩을 사용해서 서비스를 공개함

- **VIII. 동시성(Concurrency)**

프로세스 모델을 사용한 확장

- **IX. 폐기 가능(Disposability)**

빠른 시작과 그레이스풀 셧다운(graceful shutdown)을 통한 안정성 극대화

- **X. dev/prod 일치**

development, staging, production 환경을 최대한 비슷하게 유지

- **XI. 로그**

로그를 이벤트 스트림으로 취급

- **XII. Admin 프로세스**

admin/maintenance 작업을 일회성 프로세스로 실행

출전 http://12factor.net/ko/

자체개발의 토대가 되는 요소들

오픈소스

- 이미 오픈소스 사용은 선택이 아닌 필수
- 오픈소스 소프트웨어는 공짜 소프트웨어가 아니다
- 올바로 선택하고 활용하기 위해서는 일정수준의 역량이 필요
- 개발 역량을 기르는데에도 도움이 됨
- 역시 금전적으로도 많은 이득이 있음

클라우드 컴퓨팅

- 네트워크, 하드웨어, 소프트웨어 설치,보안등은 비즈니스 핵심 영역과 아무런 상관이 없음
- 하지만 이러한 인프라가 제대로 받쳐주지 않으면 안정적인 서비스가 불가능함
- 비기능 요구사항의 상당부분은 클라우드 컴퓨팅을 통해 상당부분 해결 가능
- 사실상 오픈소스와 클라우드 컴퓨팅은 한몸임

사람이 핵심이다

브랜딩전략

- 한겨울에 연못에서 잉어를 잡는법
  - 커뮤니티, 개발자블로그, 강연, 방송등을 이용하여 개발자 모집을 위한 블랜딩을 진행
  - 잡는다고 끝이 아님
- 프로그래머는 무엇으로 사는가?
  - 개발자에게 돈보다 큰 가치를 지니는것
  - 회사가 배움의 터전이 될 수 있어야 함
  - 교육,승진,보상,해고에 회사의가치관을 녹여 낼 수 있어야 함

개발자 커뮤니티

- 성공과 실패에 대한 경험을 체계적으로
- 남이 풀어놓은 문제를 또 풀지 않는다
- 나의 삽질도 남에게는 소중한 경험
- 오픈소스를 사용하는 경우 개발자 커뮤니티에서의 활동은 선택이 아닌 필수사항임

정리

- 기존의 방식으로 문제가 잘 해결되지 않는경우 패러다임의 전환이 필요하다
- 자체개발 성공의 핵심은 경영진과 실무진의 성공에 대한 비전 공유
- 자체개발의 시작은 조직개편이 우선 – 소프트웨어는 그것을 만드는 조직의 모습을 닮는다
- 핵심에 집중하라
- 사람에 집중하라

감사합니다.

Q&A