라이브 게임 소프트웨어 유지보수 방법론

㈜넥슨
컴뱃암즈팀
박경호2(khpark@nexon.co.kr)

발표자 소개

업계 10 년

- MMORPG, RTS, FPS, 미들웨어
- 신규, 상용화, 라이브
- 창업, 벤처기업, 코스닥 기업, 대기업 계열사, 외국계 대기업

넥슨 컴뱃암즈팀에서 2 년

- 컴뱃암즈는 2006 년 개발 시작
- 현재 북미(2008), 유럽(2009), 브라질(2010)에서 상용 서비스 중
- 프로그래밍파트장을 맡았던 지난 1 년 경험의 공유

목차

우선,

용어 정의

넥슨은

서비스 회사다

온라인 게임 서비스 회사다, 퍼블리셔다

그런데 소프트웨어 개발도 한다

온라인 게임 소프트웨어 개발 회사다, IT 업체다

여기서 질문,

왜 서비스 회사에서

소프트웨어 개발도 하는 걸까?

사서 쓰면(퍼블리싱) 될 것 같은데?

몹시 빠른 속도로

서비스가 바뀌기 때문

매달

(여담입니다)

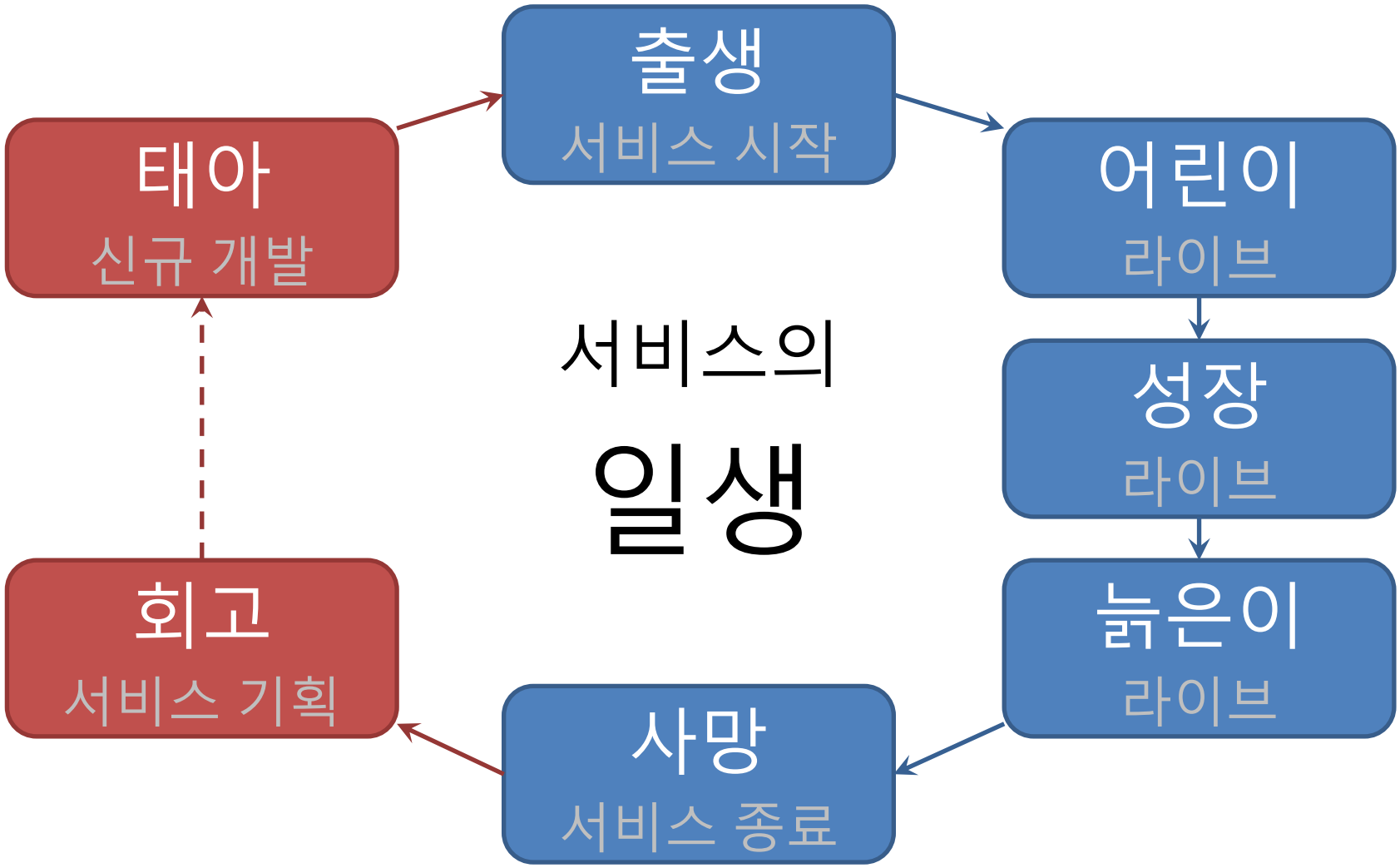
그럼 신규 개발은 왜?

원활한 라이브 전환을 위해,

역시 퍼블리셔가 자체 개발하면 유리

그런데,

라이브 전환에 문제가 없는 좋은 구조라면?


출생
서비스 시작
어린이
라이브
성장
라이브
늙은이
라이브
사망
서비스 종료
회고
서비스 기획
태아
신규 개발
서비스의
일생

‘라이브 소프트웨어’ 란

스펙이 성장하는 소프트웨어

근육 vs. 지방 – 업계의 현실은?

그럼 신규 개발에서 스펙이 계속 바뀌는 경우는?

성장이 아니라 출생의 반복일 뿐

‘라이브 게임 소프트웨어’ 란

스펙이 ‘게임’ 인 라이브 소프트웨어

라이브 소프트웨어의 ‘유지보수’ 란

매우 짧은 주기의 빈번한 패치

버그 수정, 장애 대응 – 좁은 의미

‘유지보수 방법’ 이란

짧은 주기에 적응하는 방법

라이브 유지보수

업무 항목

라이브 유지보수 업무의
1 + 5

스펙 변경
- 추가, 제거, 수정

지식 관리

서비스

버그 관리

툴 제작

리팩토링

스펙 변경

기존 컨텐츠 개선

신규 컨텐츠 추가

제거는?

지식 관리

기본

- 개발 업무 지식
- 외부(퍼블리셔) 배포 지식
- ‘지식인’ 서비스

확장

- 파트 내부 엔지니어 세미나
- 팀내 툴 사용법 교육
- 외부 세미나(NDC, KGC!)

서비스

라이브 서비스 관리
- 점차 퍼블리셔에 인계

고객 대응 업무 지원
- 점차 퍼블리셔에 인계

빌드 제작 및 관리
- 각 국가 버전별

팀 운영 지원
- 팀 테스트용 공용 서버(개발 서버) 관리
- 팀 업무용 공용 서버 관리, 개인 PC 사용 교육 및 장애 상담, 보안 관리

버그 관리

버그 관리

- 이슈 트래킹 시스템(ITS) 관리
- 기타 각종 버그 리포트 확인 및 ITS 에 등재
- 리포팅된 버그 재현 및 1 차 원인 분석
- 담당자에게 버그 배정
- 버그 우선순위 관리
- 버그 관련 외부 커뮤니케이션

버그 직접 수정

툴 제작

개발 툴

엔지니어링 툴, 게임 디자인 툴, 아트 툴

서비스 툴

서버 관리, 게임 운영(샵, 이벤트 등등)

통계 툴

로그 분석

리팩토링

팀내 협업 프로세스

코드 구조

애플리케이션 성능

라이브 유지보수

항목별 방법론

라이브 유지보수

스펙 변경

소프트웨어 유지보수의
일반적인 경우

v1.x

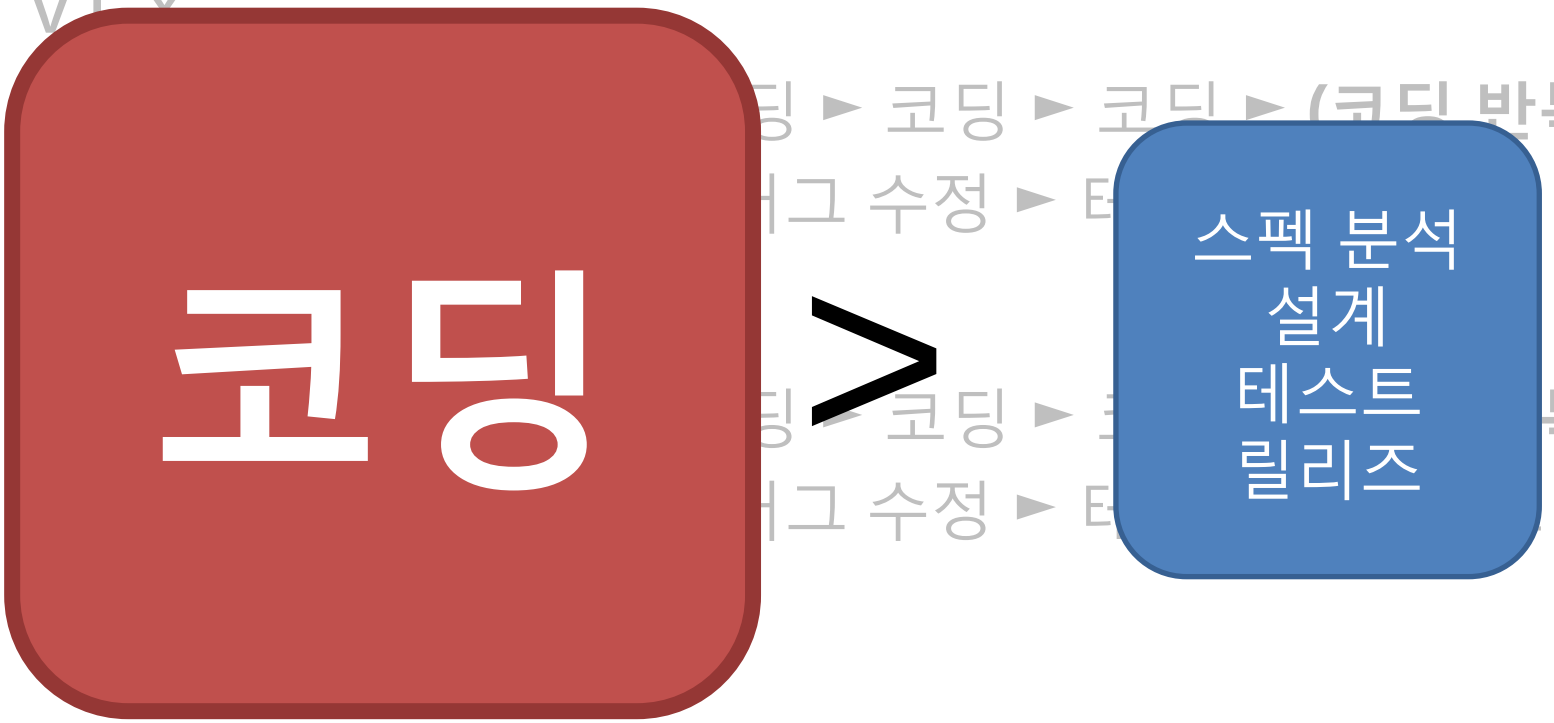
스펙 분석 ► 설계 ► 코딩 ► 코딩 ► 코딩 ► **(코딩 반복)**
► 테스트 ► 릴리즈 ► 버그 수정 ► 테스트 ► 릴리즈

v2.x

스펙 분석 ► 설계 ► 코딩 ► 코딩 ► 코딩 ► **(코딩 반복)**
► 테스트 ► 릴리즈 ► 버그 수정 ► 테스트 ► 릴리즈

소프트웨어 유지보수의

일반적인 경우



소프트웨어 유지보수의

일반적인 경우

스펙 변경이 예측 가능하고 빈번하지 않으므로,

프로그래머들은 대개 **코딩** 업무에 집중한다

스펙 분석, **설계**, **테스트**, **릴리즈** 업무는

겸사겸사 가능하다

그런데

라이브 유지보수의 경우

v1.0

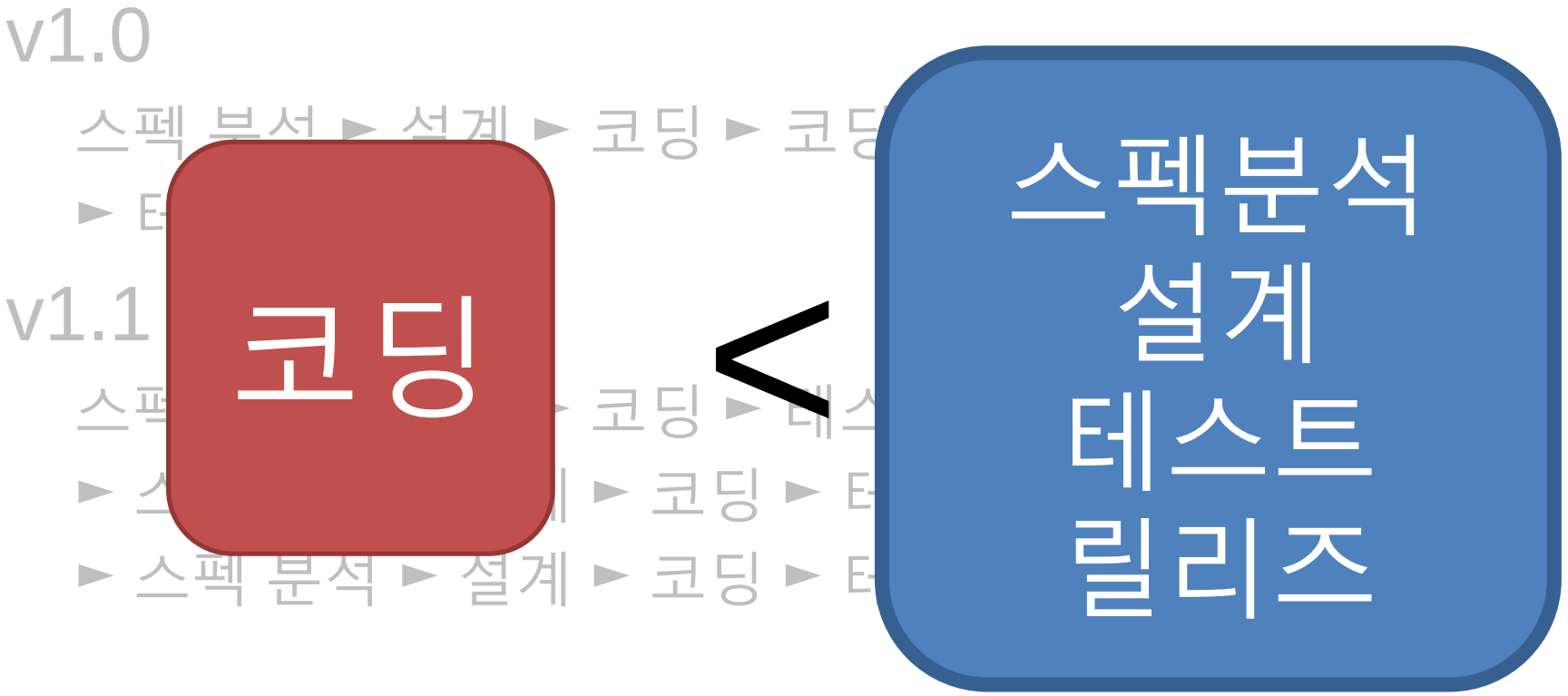
스펙 분석 ► 설계 ► 코딩 ► 코딩 ► 코딩 ► **(코딩 반복)** ► 테스트 ► 릴리즈

v1.1 ~ v1.xx

스펙 분석 ► 설계 ► 코딩 ► 테스트 ► 릴리즈
► 스펙 분석 ► 설계 ► 코딩 ► 테스트 ► 릴리즈
► 스펙 분석 ► 설계 ► 코딩 ► 테스트 ► 릴리즈

그런데

라이브 유지보수의 경우



그런데

라이브 유지보수의 경우

프로그래머들이 태아 시절 하던대로 하다가는,

태도과 행동의 인지부조화 상태에 빠진다

거참 일도 얼마 안하는데 이상하게 바쁘네..? ——a

출생 → 성장

키워드

혼자 많이 → 함께 오래

영웅 → 바보

효율 → 안전

강점 극대 → 약점 보완

EDGE → DETAIL

출생 → 성장

엔지니어 선호도

코딩 → 리팩토링

즐거움 자부심

범용 기능 → 전문 기능

이직에 도움 승진에 도움

긴 개발 기간 → 큰 보상

시간 여유 지갑 여유

라이브 유지보수

지식 관리

출생 → 성장

담당자 불변 → 담당자 변경

위험 감수 → 위험 회피

지식 소비자 적다 → 지식 소비자 많다

지식 가치 낮다　　지식 가치 높다

지식 관리

지식 관리의 핵심은 문서화, 검색 편의

문서화를 업무 프로세스에 반영

지식 체계 수립

한 곳에서 관리

지식 관리의 성패는 지식의 품질

품질 평가

기준 이상 유지

지식 보관 장소

- 보관 장소는 중요하지 않다
  - 파일 시스템
  - 서브버전
  - 지라
  - 위키

지식 보관 장소

서브버전과 지라 연동

- 지라에 서브버전 플러그인 설치
- 메타 데이터 관리는 지라로 극복
- 버전 관리는 서브버전으로 극복

위키

- 편집 기능 활용성 적음
- 지식 분산 가능성

지식 관리 시나리오

생산

- 지라에서 업무 배정
- 서브버전에 지식 생성
- 체크인할 때 지라 번호 등록 – 자동으로 지라에 등록됨
- 필요하면 지라에 별도 메타 데이터(코멘트, 링크 등) 등록

활용

- 지라에서 검색하면 서브버전의 지식도 동시에 검색됨
- 활용 후 지라에 간단한 코멘트
- 지식 내용 개선하여 서브버전에 업데이트

라이브 유지보수

서비스

자동화

라이브 서비스 관리
자동화

CS 업무(DB) 지원
자동화

빌드 제작 및 관리
자동화

팀 운영 지원
자동화?

라이브 유지보수

버그 관리

버그 관리

소스 코드 관리 시스템(SCM)과 이슈 트래킹 시스템(ITS)을 잘 사용하고 있다면,

별다른 이슈 없어요

라이브 유지보수

툴 제작

툴 제작

개발 툴

빌드 자동화

이건 별도 세션에서..

서비스 툴

설명 필요 없을 듯

통계 툴

다양할수록 훌륭한 팀

라이브 유지보수

리팩토링

리팩토링 키워드

후임자 배려

후임자들이 점점 무능력하거나 게을러지는 것 같으면,

괜히 엄한 사람 잡지말고, 리팩토링

게임 이론

이건 다음 기회에

그 때까지 살아있으면..

라이브 유지보수

항목별 업무량 비율

우선순위

P1: 지식 관리

P2: 서비스

P3: 버그 관리

P4: 스펙 변경

P5: 툴 제작

P6: 리팩토링

우선순위 vs. 업무량 비율

우선순위와 업무량 비율은 별개

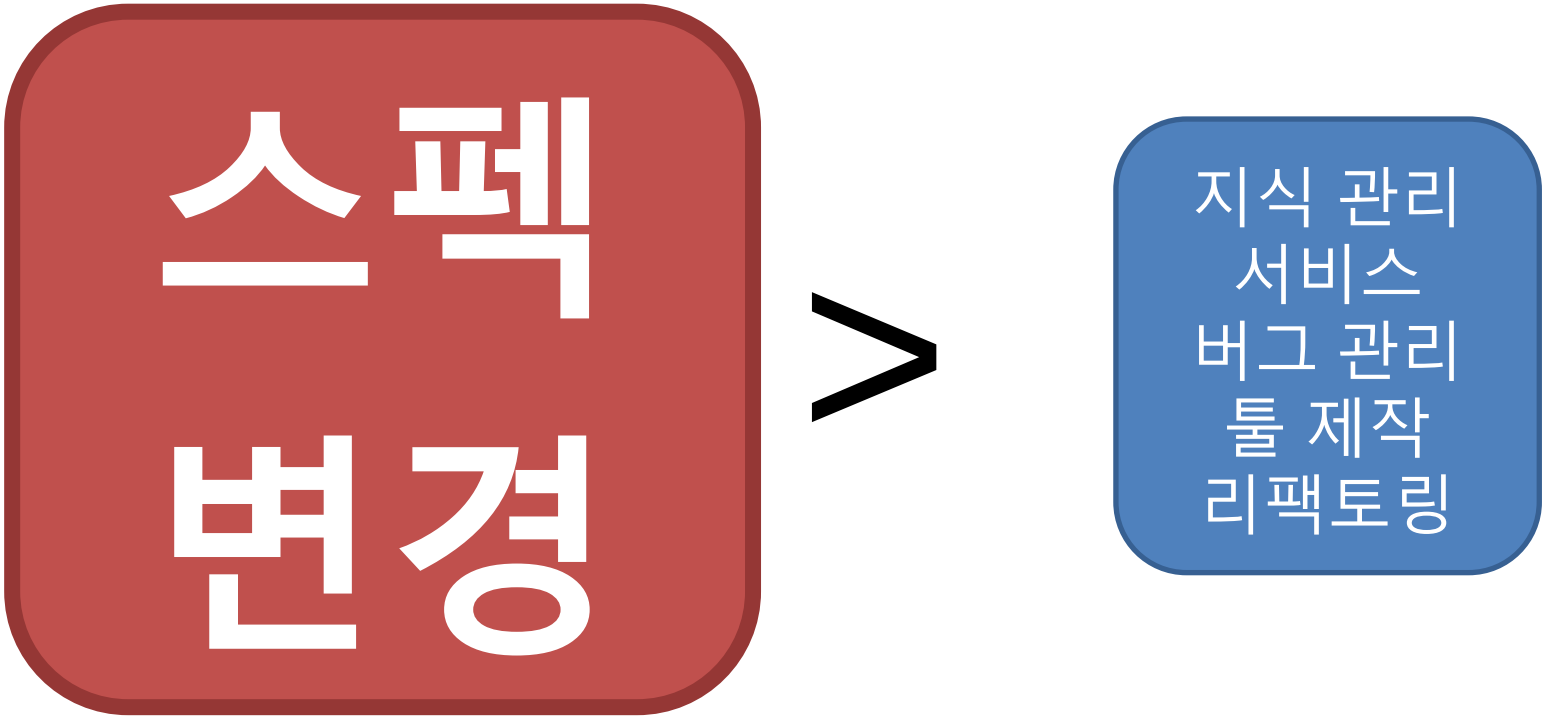
우선순위는 고정되어 있는 반면,

‘업무량 비율’ 은 살아있다!

이것이 진정한 라이브의 세계

라이브 팀의 항목별 업무량 비율

훌륭한 예



라이브 팀의 항목별 업무량 비율

훌륭한 예

라이브 팀의 항목별 업무량 비율

훌륭한 예

라이브 팀의 항목별 업무량 비율

따라하다가는

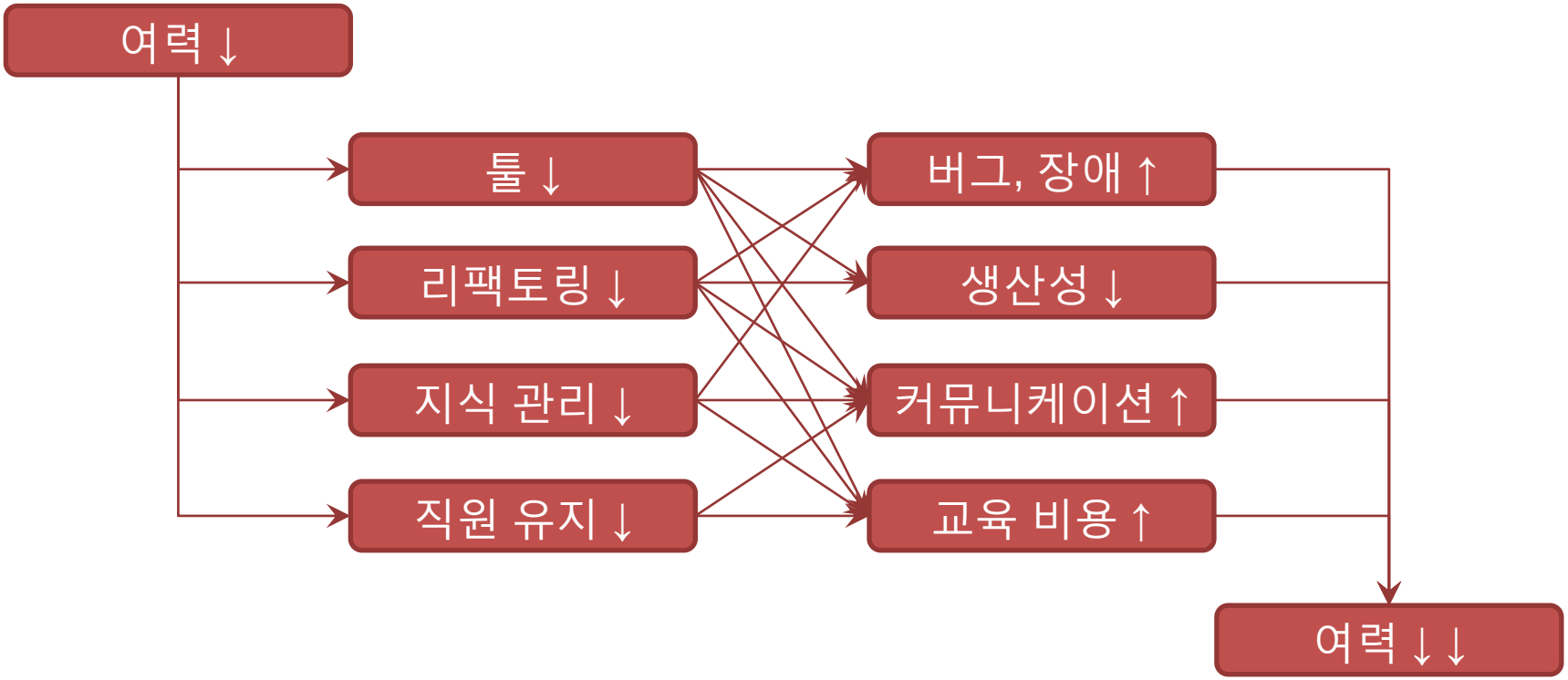
스펙 변경 > ?

현실적으로

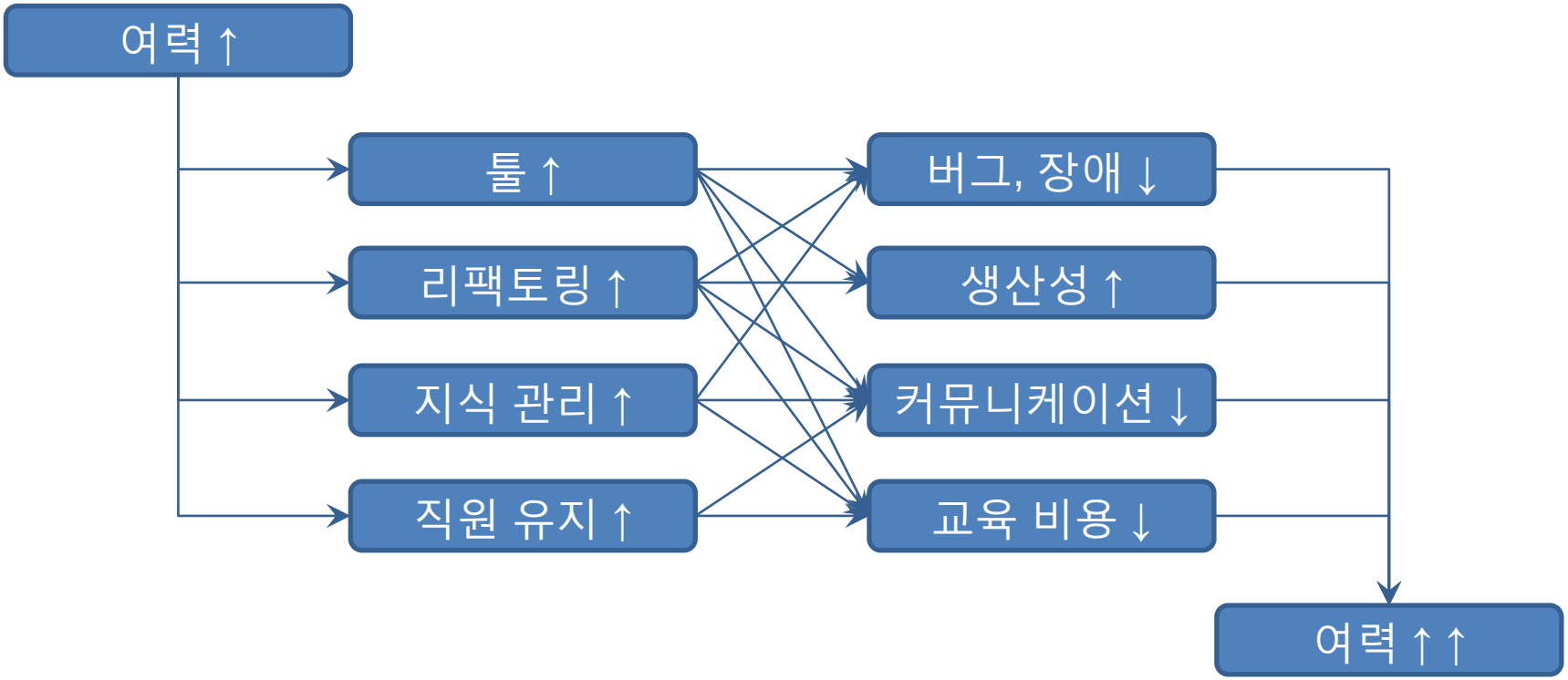
좋은 예

스펙 변경 < 지식 관리
서비스
버그 관리
툴 제작
리팩토링

나쁜 예의 악순환



그런데 반대의 경우



악순환 선순환 반전 키워드

메타 업무 능력

자신의 여력을 판단할 수 있는 능력

판단 결과,

여력이 없으면 만들어야

악순환 선순환 반전 키워드

메타 업무 능력 + 용기

자신의 여력을 판단할 수 있는 능력

판단 결과,

여력이 없으면 만들어야

업무 비율 계속 조정

용기가 필요하다, 왜?

라이브 유지보수

컴뱃암즈의 경우

지식 관리

파일 시스템, 위키, 셰어포인트, 서브버전, 지라, 메일, 개인별 기억 등등

대략 난감

아직 기획 단계

다음 기회에

서비스

라이브 서비스 관리
자동화

CS 업무(DB) 지원
자동화

빌드 제작 및 관리
자동화

팀 운영 지원
자동화?

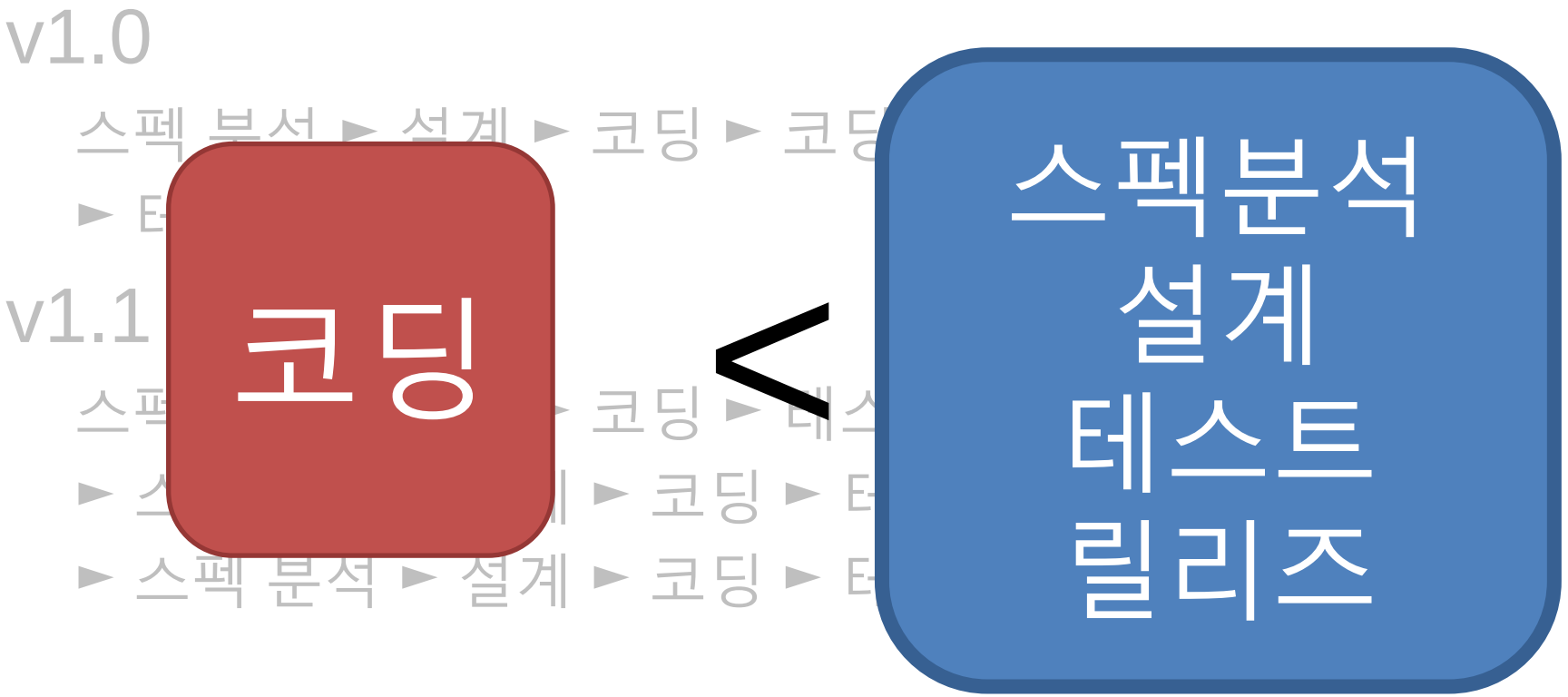
버그 관리

소스 코드 관리 시스템(SCM)과 이슈 트래킹 시스템(ITS)을 잘 사용하고 있으니,

별다른 이슈 없어요

퍼블리셔와도 협업 중

스펙 변경



툴 제작

개발 툴

빌드 자동화

이건 별도 세션에서..
그밖에 게임 디자인 툴 등등

서비스 툴

완료

통계 툴

기대 중

리팩토링

이건 내년에

오늘 반응 좋으면..

맛보기

2011 년 상반기 컴뱃암즈
소스코드 리팩토링 계획

- DB

2011-06-01(v1.2.1.ndc2011)

컴뱃암즈팀
박경호2(khpark@nexon.co.kr)

질문, 다른 의견,

하고싶은 이야기

있으신 분?

못다한 이야기는

박경호2(khpark@nexon.co.kr)