대용량 아키텍쳐 설계

서비스 지향 아키텍쳐

#목표

"백앤드 아키텍쳐에 많은 영향을 준 서비스 지향 아키텍쳐에 대해서 알아본다."

SOA는 XML 기반의 웹서비스로 오역되어 있지만, SOA는 아키텍쳐 스타일 뿐이다. 근래에 유행하는 REST API나 마이크로 서비스 아키텍쳐들은 SOA에 근간을 두고 있다.

#1

기본 개념의 이해

오늘 SOA를 보는 이유?

- SOA는 2000년대 초반에 부각된 아키텍쳐 스타일
- 세부 구현 기술은 변했으나, 현대의 분산 시스템의 아키텍쳐 사상으로 그대로 반영되고 있음
- SOA의 본질적인 개념을 파악하여 대용량 분산 시스템 구현에 활용

목 차

#1

SOA의 기본 개념

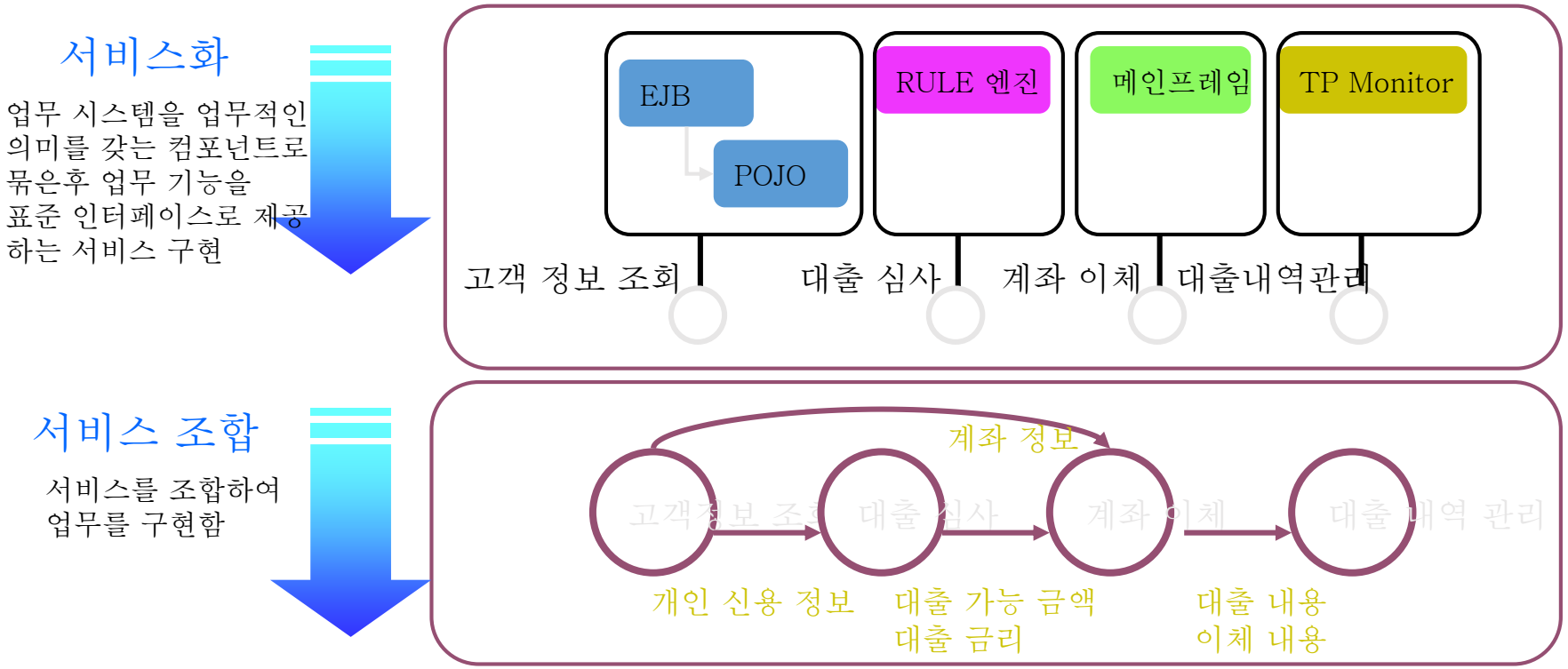
엔터프라이즈 시스템의 발전

- IT 시스템의 패러다임의 변화

Approach	Timeframe	Development Model	Business Motivations
Mainframe	1960s – 1980s	Procedural (COBOL)	Automated Business
Client-server	1980s – 1990s	Database(SQL), Fat Client (Visual Basic)	Computing On the Desktop
N-Tier model and the Web	1990s – 2000s	Component or Object-Oriented (EJB, COM)	Internet and e-Business
Service Oriented	2000s – for the time being	Service-Oriented	Business Agility

WHAT IS SOA?

- SOA (Service Oriented Architecture)란?
  - 기존의 애플리케이션의 기능들을 비지니스적인 의미를 가지는 기능 단위로 묶어서 표준화된 호출 인터페이스를 통해서 서비스로 구현하고,
    이 서비스들을 기업의 업무에 따라 조합하여 애플리케이션을 구성하는 소프트웨어 개발 아키텍쳐



WHY SOA?

- SOA가 주목 받는 이유
  - 웹서비스의 등장 (CORBA,DCOM등의 기술적 복잡도를 낮춤) → 기술적인 대안 등장
  - 점점 확장되어가는 독립된 업무 시스템 →통합에 대한 요구
  - 기업의 비즈니스 속도가 빨라져 감에 따라 IT 시스템의 업무 변화에 대한 민첩한 대응력이 필요하게 됨 → 민첩성에 대한 요구

#2

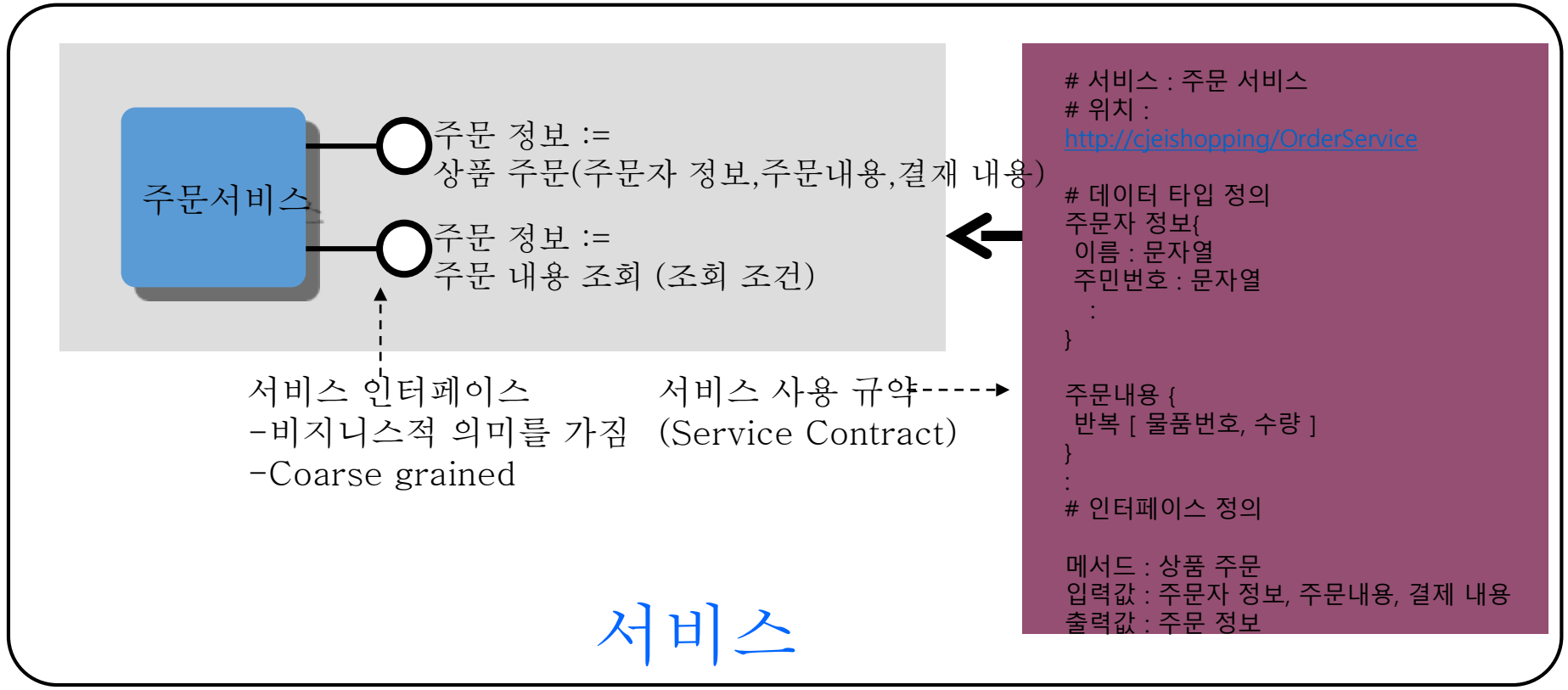
서비스

서비스의 정의

- 서비스란?
  - 플랫폼에 종속되지 않는 표준 인터페이스를 통해서 **기업의 업무를** 표현한 Loosely Coupled 하고 상호 조합 가능한 소프트웨어 컴포넌트
  - 서비스의 예시
    - 임직원정보 서비스
    - 계좌이체 서비스
    - 상품 주문 서비스
  - 서비스로 적절하지 않은것
    - JNDI Lookup
    - SMTP 이메일 클라이언트

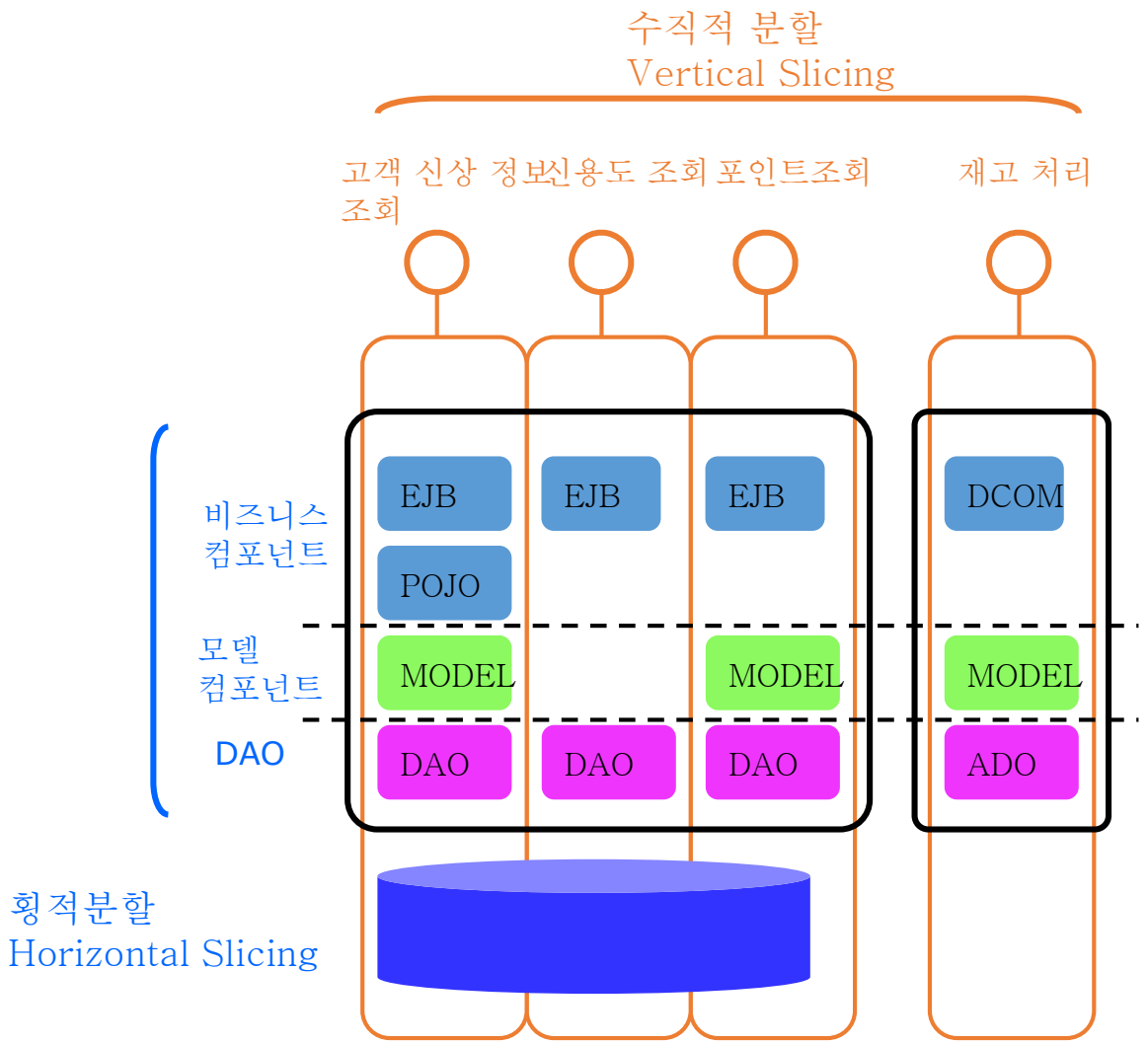
서비스의 개념

- 서비스 구성
  - 비즈니스적 의미를 가지는 기능(Method)들을 모아놓은 소프트웨어 컴포넌트이다.



서비스의 특징

- 서비스의 특징
  - Vertical Slicing
  - Has standard based interface
  - Loosely coupled
  - Composable
  - Coarse grained
  - Discoverable

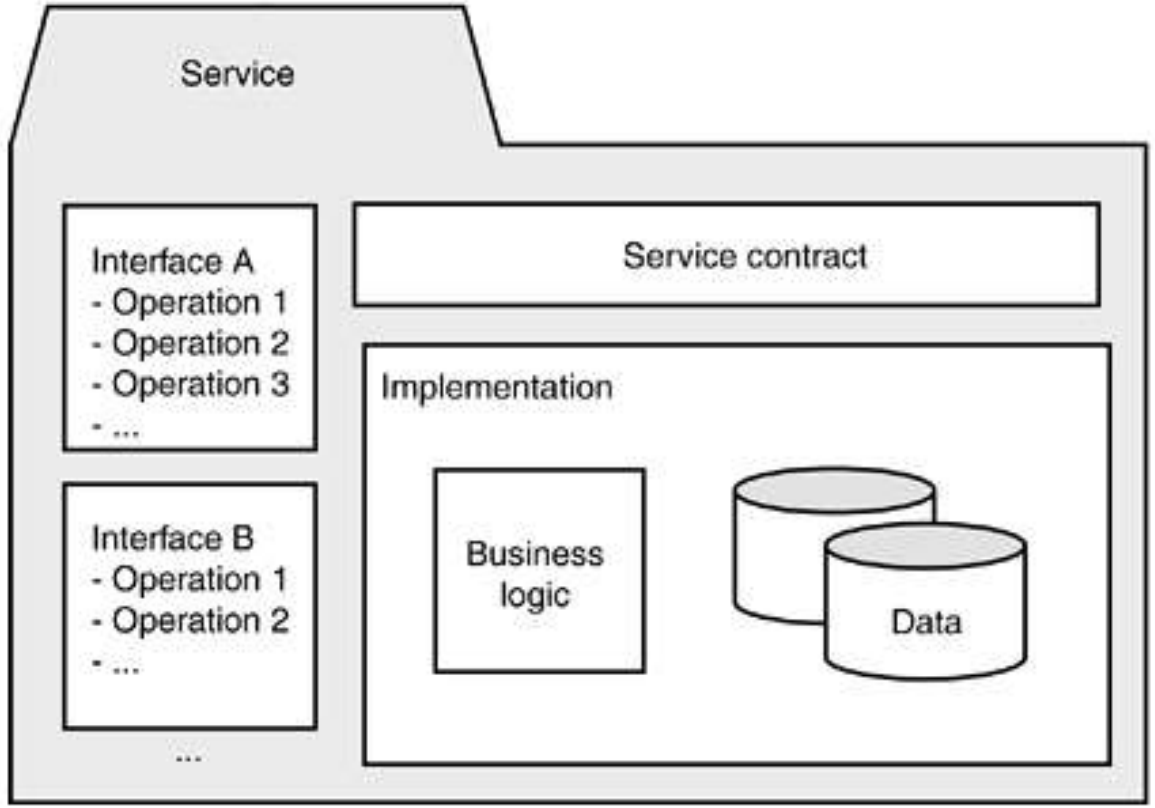


서비스 구성

- 서비스의 구성
  - 서비스 규약(Contract)

    ex) WSDL
  - 서비스 인터페이스
  - 서비스 구현체(Implementation)

    데이터와 비지니스로직을 모두 포함



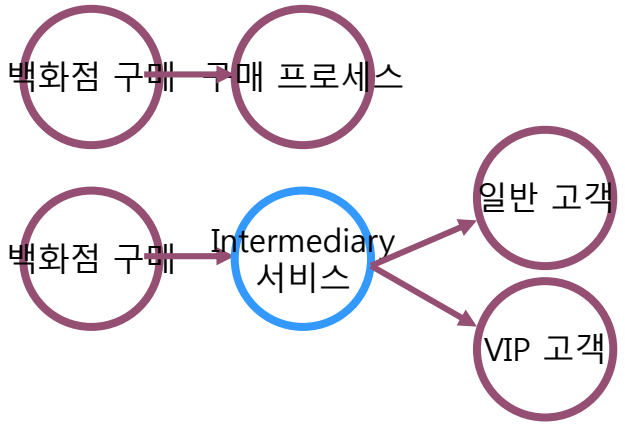
서비스의 종류

- 비즈니스 서비스
  - 일반적으로 SOA에서 정의하는 서비스
  - Task centric service → EJB Session Bean
  - Data centric service → EJB Entity Bean

※서비스의 종류 분류 기준 Enterprise SOA - DIRK KRAFZIG / Prentice hall

서비스의 종류

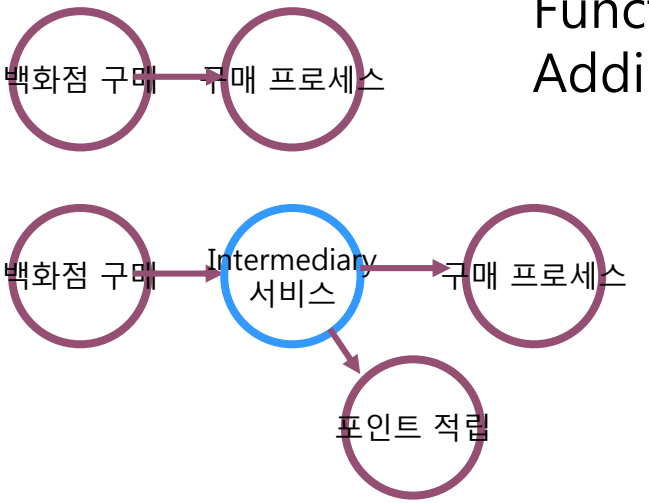
- Intermediary 서비스
  - Routing
  - Transformation
  - Functional adding
  - Façade etc. * 채널 시스템



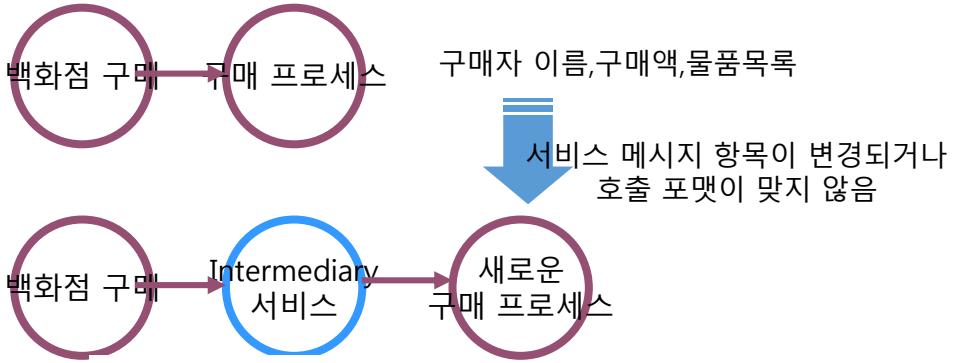
Routing

총 구매 금액 > 1000만원

Functional Adding



Transformation

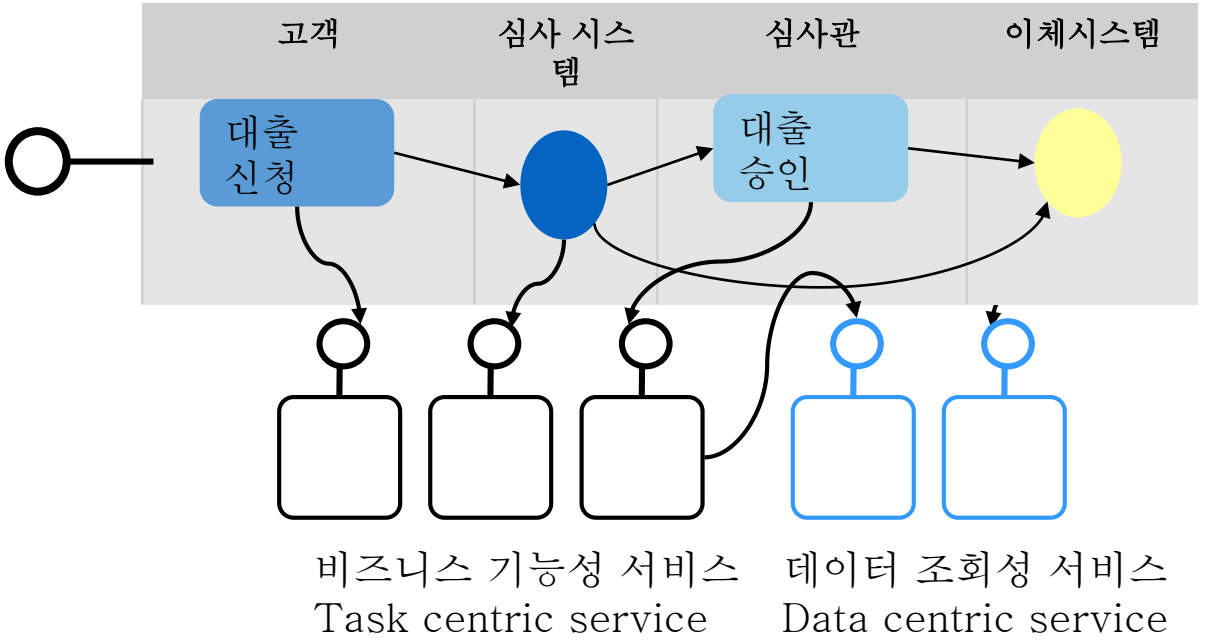


구매 내용 = 구매액 + 물품목록

구매자 이름,구매 내용

서비스의 종류

- Process Centric 서비스
  - 비즈니스 서비스를 조합(Orchestration)
  - 업무 프로세스를 구현
  - 상태 정보가 있을 수 있음 (Stateful)



서비스 종류 요약

- Application 서비스
  - 테크니컬한 기능을 구현한 서비스

  예) Load Balancing 서비스, Transaction 관리 서비스, Service adapter, 보안 인증, Document Parsing, Logging etc

- Public Enterprise 서비스
  - 타기업이나 외부 시스템으로 서비스를 제공하고자할때 정의됨
  - 보안,과금,성능등이 고려됨

  예) 금융권 대외계 업무

서비스 종류 요약

	Application Service	**Business Service**	Intermediary Service	Process centric Service	Public Enterprise Service
Description	Express technology-specific functionality	**Contains business logic and data.**	Bride the gap from concept or design	Encapsulate business process and orchestrate other services	Export business logic to other enterprise or organization
Sub category	Wrapper svc Utility svc	**Task centric ,Entity centric**	Façade ,Function Adding,Routing,Transforming	Process centric	
State mgmt	No	**No**	No	Yes	No
Reusability	??	**High**	Low	Very Low	High
Mandatory in SOA	No	**Yes**	No	No	No
Note		**Most Important Service** *** Hybrid svc**	It needs when new service or new service connection is made		

#3

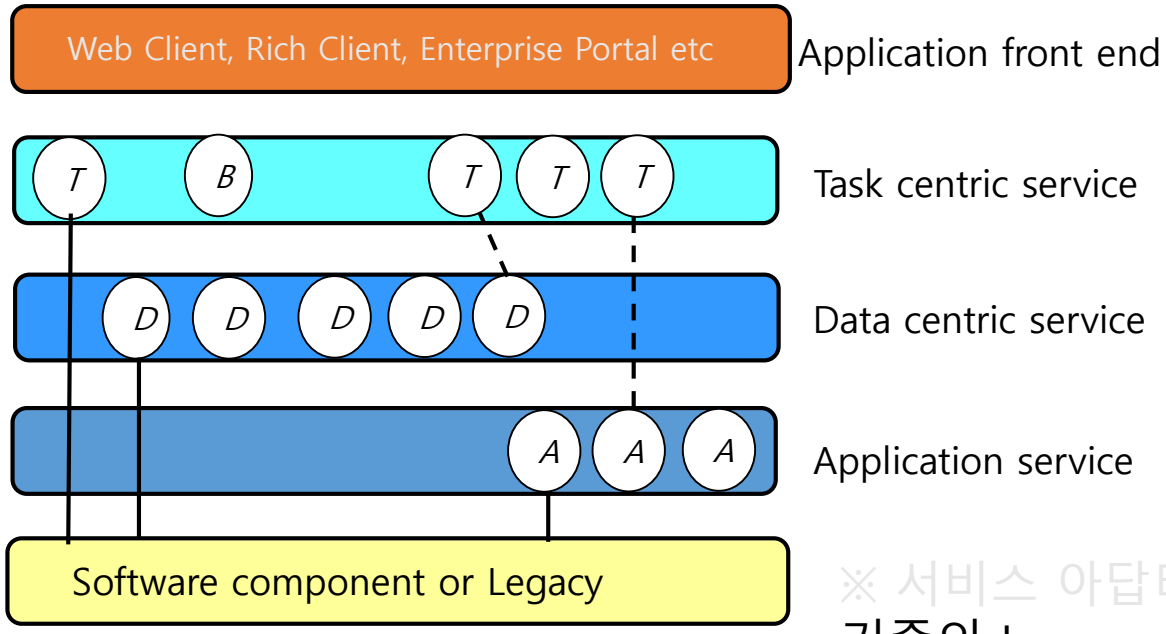
SOA 아키텍쳐 발전 모델

SOA 단계적 발전구조

- SOA는 시스템의 규모와 업무적 요구 사항에 따라 다음 3단계 순서로 발전할 수 있다.
  - Fundamental SOA
  - Networked SOA
  - Process Oriented SOA

Fundamental SOA

- 기존 시스템들을 서비스화하여, 각 시스템들을 통합하는 단계
  - 서비스화와 통합이 중점 전체를 한 시스템화함
  - 서비스에 대한 조합 → Application Front End에서 담당
  - 비즈니스 서비스 + Application 서비스로만 구성됨

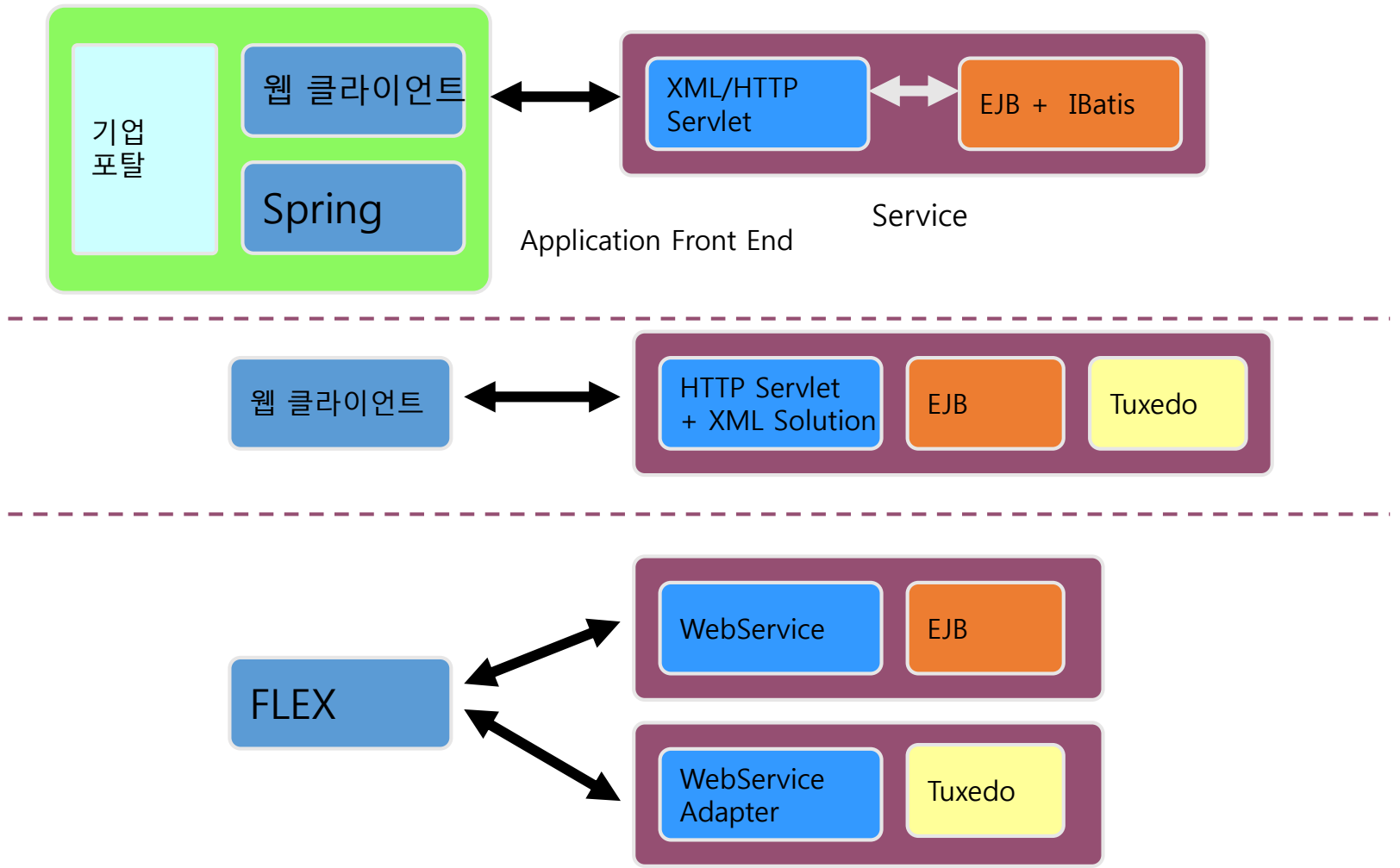


< Fundamental SOA 개념도 >

※ 서비스 아답터
기존의 Legacy 시스템의 기능을 웹서비스화 해주는 솔루션
EX) SALT (Tuxedo to WebService)
IWAY 아답터 시리즈 (CICS,SAP,Siebl

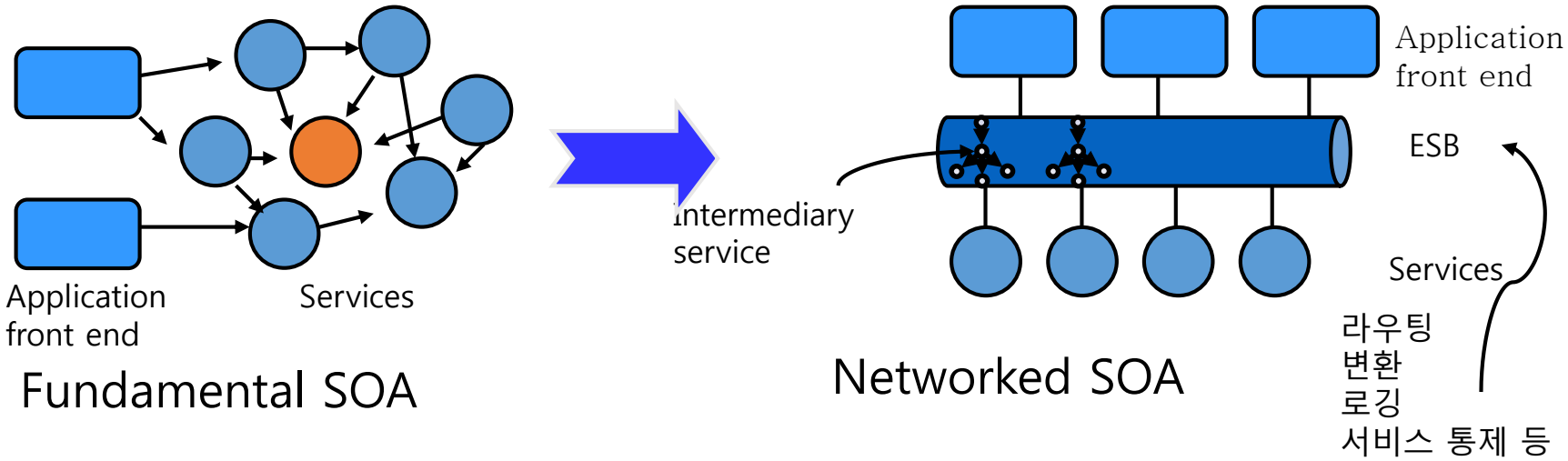
Fundamental SOA

- Fundamental SOA 예제



Networked SOA

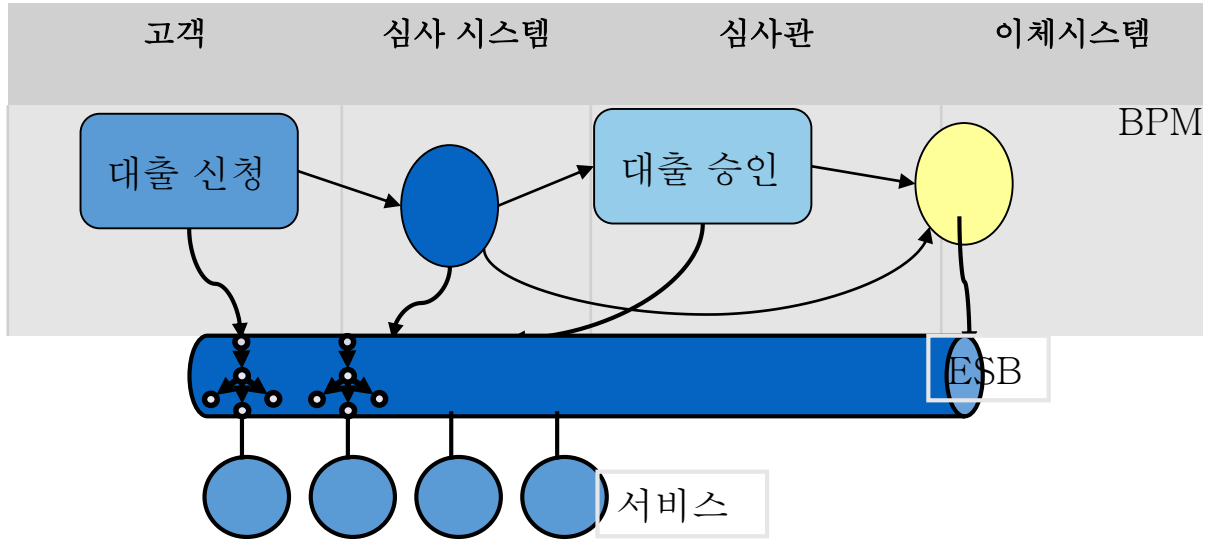
- Fundamental SOA의 문제점
  - 시스템의 크기가 증가함에 따라 서비스와 서비스, 서비스와 Application Front End단의 연결이 매우 복잡해짐 (거미줄식 P2P연결)
  - 시스템의 유연성이 떨어짐
  - 관리 및 중앙 통제에 있어서 문제가 발생
- Networked SOA
  - SOA 시스템의 가운데 서비스 허브 (ESB:Enterprise Service Bus)를 둬서 서비스의 중앙 통제력 및 유연성을 강화함
  - Intermediary 서비스가 ESB에 위치함



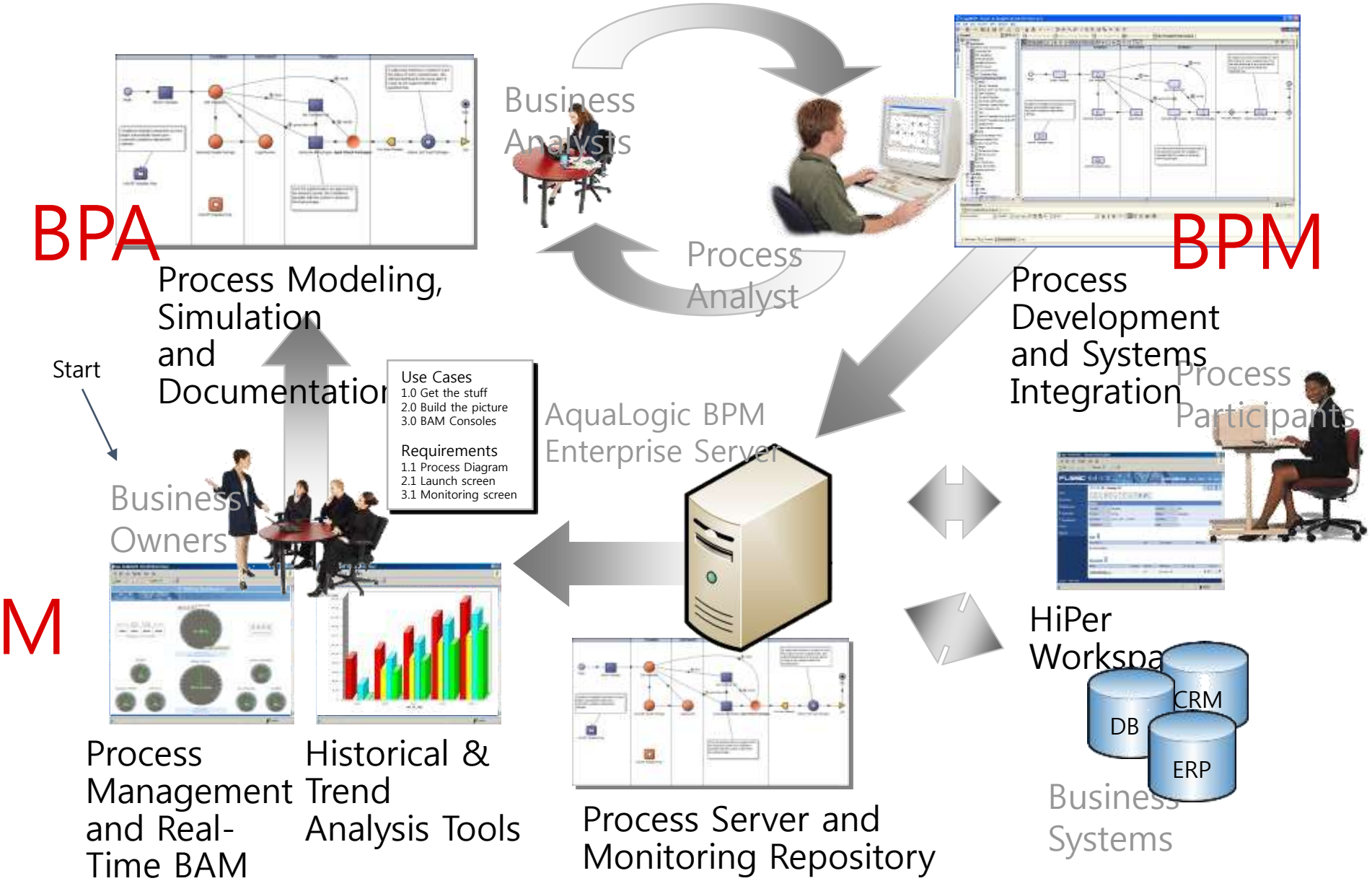
Process Oriented SOA

- Process Oriented SOA
  - 비즈니스 플로우 (Business Flow)가 있을 경우에만 적용
  - 서비스의 조합을 통한 업무의 구현을 BPM을 이용함
  - 업무 (기능) 변화에 매우 민첩하게 반응 가능 (Agility)
  - 기술조직과 비즈니스 조직간의 의사 소통이 원할함

  CF. BPA, BPM

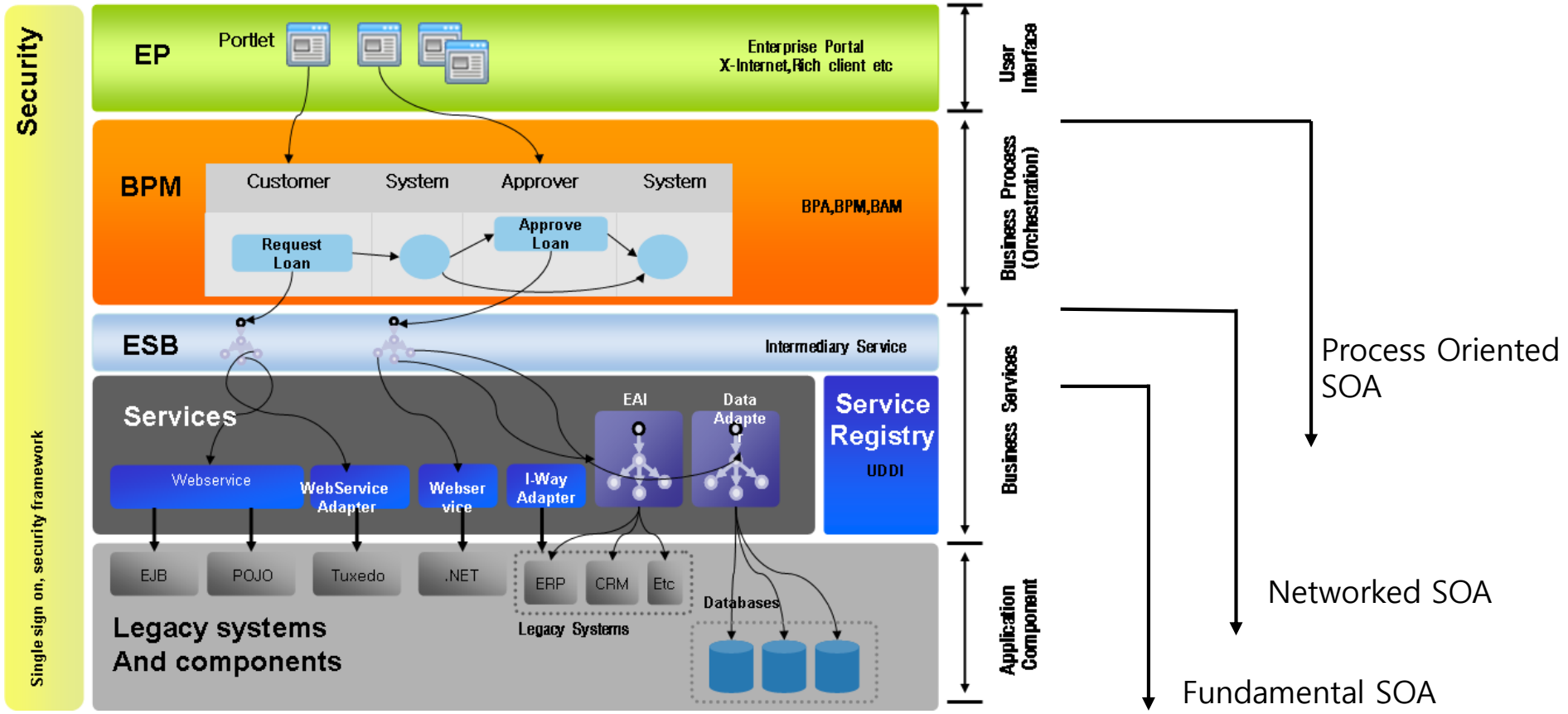


Process Oriented SOA

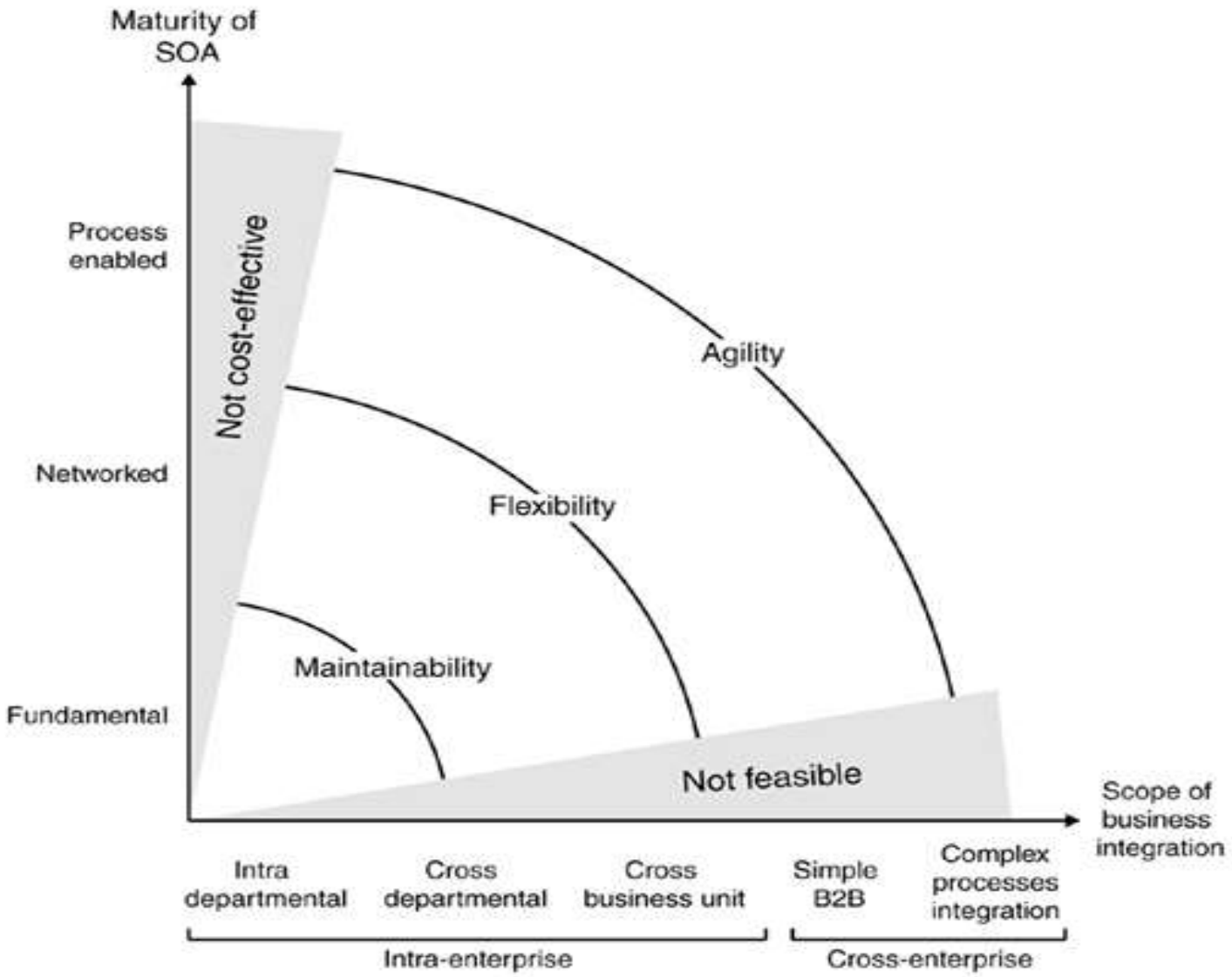


SOA reference structure

- Reference architecture



SOA 단계적 발전 단계


Maturity of SOA
Process enabled
Networked
Fundamental
Not cost-effective
Agility
Flexibility
Maintainability
Not feasible
Scope of business integration
Intra departmental
Cross departmental
Cross business unit
Simple B2B
Complex processes integration
Intra-enterprise
Cross-enterprise

#4

SOA 구현시 고려 사항

서비스화

- Service Adapter 고려
  - 기존의 서비스를 손쉽게 웹서비스화 하기 위해서는 Service Adapter의 도입을 고려
  - Ant Task등을 이용한 EJB,POJO의 자동 웹서비스화
  - Adapter를 이용한 Tuxedo,SAP등의 Legacy 자동 웹서비스화
- 서비스 인터페이스 표준 결정
  - 인터페이스 표준을 어떤것을 사용할것인가?
    - 웹서비스? CORBA? XML/HTTP? → 확장성,기술 도입 편이성,호환성
    - 웹서비스의 경우

      웹서비스의 확장 규격인 WS* (WS-Transaction,WS-Coordination,WS-Security etc)을 사용할 경우 Service Adapter별로 지원하는 수준이 다름

트렌젝션 처리

- 표준 Webservice 스펙으로는 서비스간 트렌젝션 관리가 불가
- 전체 서비스중 트렌젝션 연계가 필요한 업무는 10%미만(Enterprise SOA- DIRK KRAFZIG)
- 방안
  1. WS*중 WS-Transaction & WS-Coordination을 통해서 구현가능 → 아직 솔루션들에서 완벽하게 지원하지 않음
  2. EAI를 통한 Transaction이 보장되는 Tightly coupled service 구성
  3. Compensation Transaction(보상 트렌젝션)등 대안 구현
  4. 서비스별 Logging

보안

- 인증과 권한
  - 분산된 서비스에 대한 통합된 사용자 인증과 권한 관리 필요
- 암호화
  - 암호화 방법(대칭키,비대칭키) 결정
  - 암호화 범위 결정
    - 전체 메시지를 암호화 할것인가? 메시지 내용중 중요 데이터만 암호화 할것인가?
    - 암호화할 내용을 메시지 헤더에 넣을것인가? BODY에 넣을것인가?

모니터링

- 모니터링
  - 각 서비스의 성능/용량 데이터 필요
    - 서비스들을 조합하여 새로운 업무를 구현하고자 할때 업무의 수행시간과 가용 사용자를 예측할 수 있어야 함
    - 이를 위한 성능 데이터 수집이 필요
  - 서비스 간 연동시 병목 구간 추적 필요
    - 장애시에 병목 구간에 대한 원인 추적을 위해 필요
- 로깅
  - 어느 수준(단위)까지 로그를 남길것인가? 업무? 서비스?
  - 어디에 로그를 저장할것인가? 각각 시스템? 중앙 집중형?

서비스 검색

- 서비스를 조합하여 업무를 구현하고자 할 때 서비스를 검색할 수 있어야 함
  - 운영시 서비스 검색 (UDDI)
    - 서비스의 위치와 서비스 명세(WSDL)
    - 그외 메타 정보 (과금, 권한, 보안 규약 etc)
  - 개발시 서비스 검색 (Enterprise Repository)
    - 개발시에 서비스를 개발이나 수정할 경우, 서비스 구현에 필요한 Resource를 찾을 수 있어야 함
    - 분석,설계 명세,LIB,source code,DB 정보
    - 서비스 버전 관리

#5

어떻게 SOA를 수행할것인가?

SOA 수행 방법

- SOA 프로젝트를 수행함에 있어서, 각 관점에 따른 진행 전략
  - 전략
  - 비용
  - 통제
  - 프로젝트 관리
  - 레퍼런스 아키텍쳐

전략

- 기업의 장기적인 비즈니스 전략에 따라 IT 시스템을 전략 적용 단계에 맞춰서 개발
  - 기존 →비즈니스 각 전략 단계별로 각각의 독립된 시스템을 따로 개발
  - SOA →하나의 SOA시스템에 비즈니스 전략에 따라 해당 기능을 추가해 나감
- 서비스의 우선순위와 SOA화 범위를 비즈니스 전략의 실행단계에 맞춰서 정의

  * 기업 전략

  2004년 매출 증대
  2005년 고객 만족 실현
  2006년 브랜드 이미지 관리

  * SOA 전략

  2004년 매출 내용 전산화
  2005년 CRM 도입을 통한 고객 정보 수집과 매출 내용을 기반으로 고객 패턴 추출
  2006년 수집된 고객 정보를 토대로 마케팅 집중

레퍼런스 아키텍쳐

- 전체 기업 업무를 단일 시스템으로 운영하기 위한 플랫폼이 필요
- SOA 레퍼런스 아키텍쳐
  - Fundamental SOA
  - Networked SOA
  - Process Oriented SOA
  - Google과 Naver의 SOA전략
  - Adobe 기반의 SOA

  + 보안 인증, Application Front end, Service repository
- SOA 시스템의 크기, 기업 시스템의 SOA화, 기업 비즈니스 전략에 따라서 지속적으로 레퍼런스 아키텍쳐(플랫폼)을 발전 시켜 나감

제어 통제

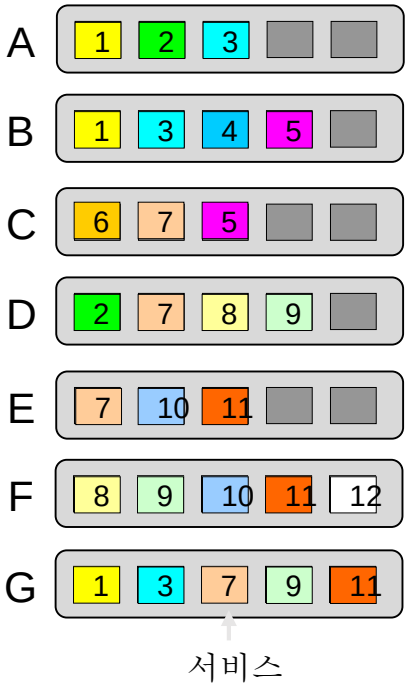
- 제어 통제가 필요한 이유 (대부분 이것 때문에 실패했음)
  - 전체 IT 시스템을 SOA화 함에 따라 장기적인 중앙 통제 그룹과 관리 도구가 필요함
- 통제 조직
  - SOA 시스템에 대한 정책 수립 및 표준화 - Standard
  - SOA 관련 기술 전파 및 가이드 - Evangelist
  - SOA 구축 계획 수립 및 실행 (로드맵)- Strategy
  - 자금 조달 및 집행 계획
  - 업무 분석 및 설계
  - 문화 변화 → IT조직과 비즈니스 협업 조직의 협업문화 개발
  - 모범사례 수집과 배포

제어 통제

- 통제 도구
  - ESB & 모니터링 툴
    - 서비스의 상태,사용 현황,성능등을 중앙 관리
  - UDDI (Run-time)
    - 배포 서비스의 검색, 위치 정보
    - 메타 정보 (가격,보안 정책 등)
  - Enterprise Repository (Implementation-time)
    - 서비스 개발에 필요한 사항(분석,설계 내용,패턴,개발정책,라이브러리,소스)
    - 서비스 개발 및 배포전 승인 프로세스
    - 프로젝트 일정 관리등
    - 버그 트랙킹 시스템 등

비용

- 초기 플랫폼을 구축하는 데 비교적 많은 비용이 소요됨
- 서비스를 재사용 재조합하여 새로운 업무를 구현함으로써, 처음부터 개발하는 기존 시스템에 비해서 시스템이 성숙화 되어감에 따라 개발 비용이 감소함
- 중앙 통제와 제어를 통해서 유지보수 비용이 감소함



애플리키이션

4,5 만 신규 개발

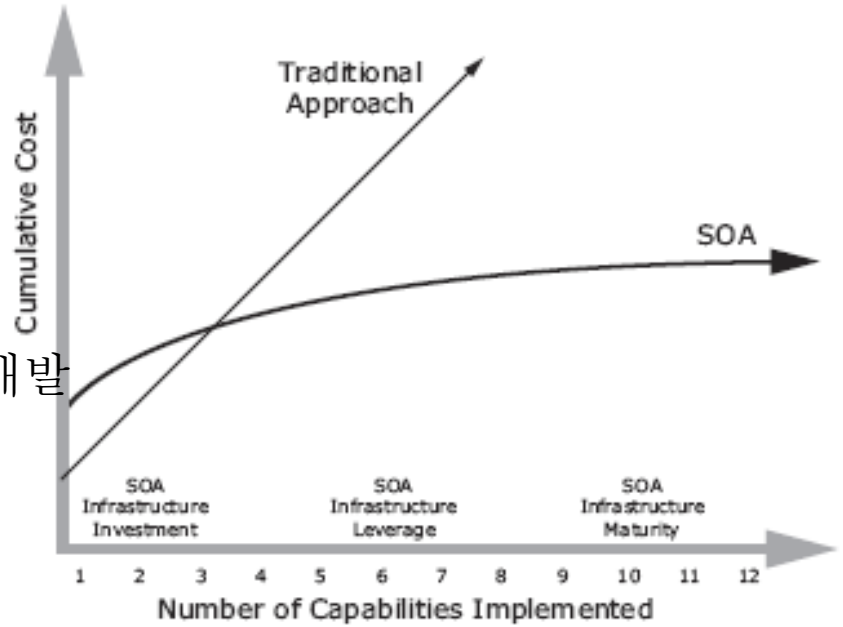
6,7만 신규 개발

8,9만 신규 개발

10과11만 신규 개발

12만 신규 개발

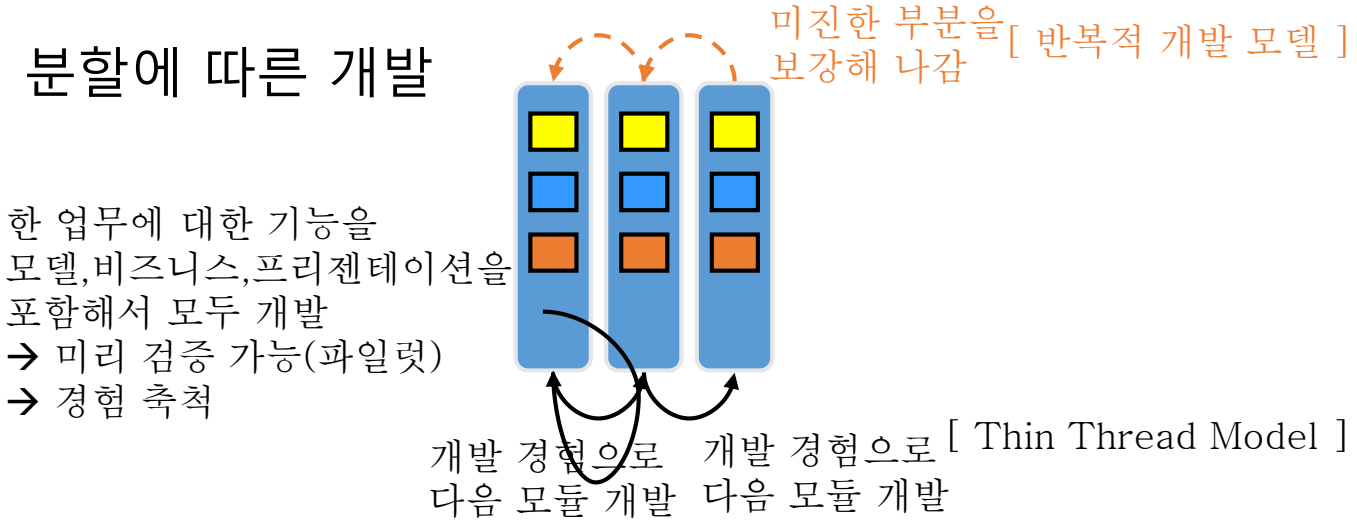
신규 개발 없음



프로젝트 진행

- 반복적 개발 모델
  - 업무를 개발해서 운영 하면서 점진적으로 업무에 맞도록 개선 및 변경해나가면서 시스템의 성숙도를 늘려감 BPA→BPM→BAM
  - 각 단위 서비스 업그레이드를 통해서 가능
- Thin Thread Model

  CF. 수직적 분할에 따른 개발



결론

- SOA는 이미 녹아들어있다.
  - 의도를 했건 안했건 현재 시스템들은 SOA적인 성격을 가지는 경우가 많다.
- 기업의 업무 환경과 요구에 맞는 단계적인 SOA 적용이 필요
  - 현재 단계에서 ESB,BPM이 필요한가?
  - SOA 단계적 발전 모델
- 서비스 개발에 대한 통제와 관리 시스템 구축 (Governance Tool) 필요
  - 개발 시작 단계에서 부터 서비스 개발에 대한 표준, 재사용을 위한 Repository 관리 필요
  - UDDI,CVS에서 부터라도 시작

현대의 웹 분산 아키텍쳐

- 구현 기술의 변화
  - ESB → API Platform
  - UDDI → API Portal
  - SOAP → JSON/HTTP
  - WebService → REST
- 세부 구현 기술은 변했을지라도, 기반 아키텍쳐 사상은 그대로 적용됨