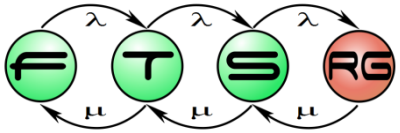
SOA referencia modellek

Szolgáltatás integráció előadás

Huszerl Gábor (BME MIT)



Kérdések

- Szolgáltatás-orientáltság platform függetlenül?
  - A WS is csak egy platform
  - A SOA alatt lehet más is
  - A szolgáltatás „örök”, a platform változik
- A Java platform hogyan támogatja a szolgáltatás alapú integrációt?
  - „Kell egy API”
  - Az alkalmazás szerverek támogassák ezt!

Szolgáltatás-orientált rendszerek

- Alapfogalmak
  - Szolgáltatás, alkalmazás, rendszer
  - Interfész
    - kiajánlott felület
  - Referencia
    - elvárt/használt felület
  - Megvalósítás
    - kibontás vagy technológiához kötés
  - Üzleti objektumok
    - üzenet/hívás adatstruktúrák
- Technológiafüggetlenség

Platformfüggetlenség

- Szabványosítás (nemzetközi, állami, iparági/ipari)
- Cégek közötti megállapodás (piaci részesedés)
  - Nem kötelező érvényű
  - „Közös érdek”, közös álláspontot tükröz
  - Korlátozza a vevő „röghöz kötés”-ét
  - Korlátozza a gyártó innovációs szabadságát

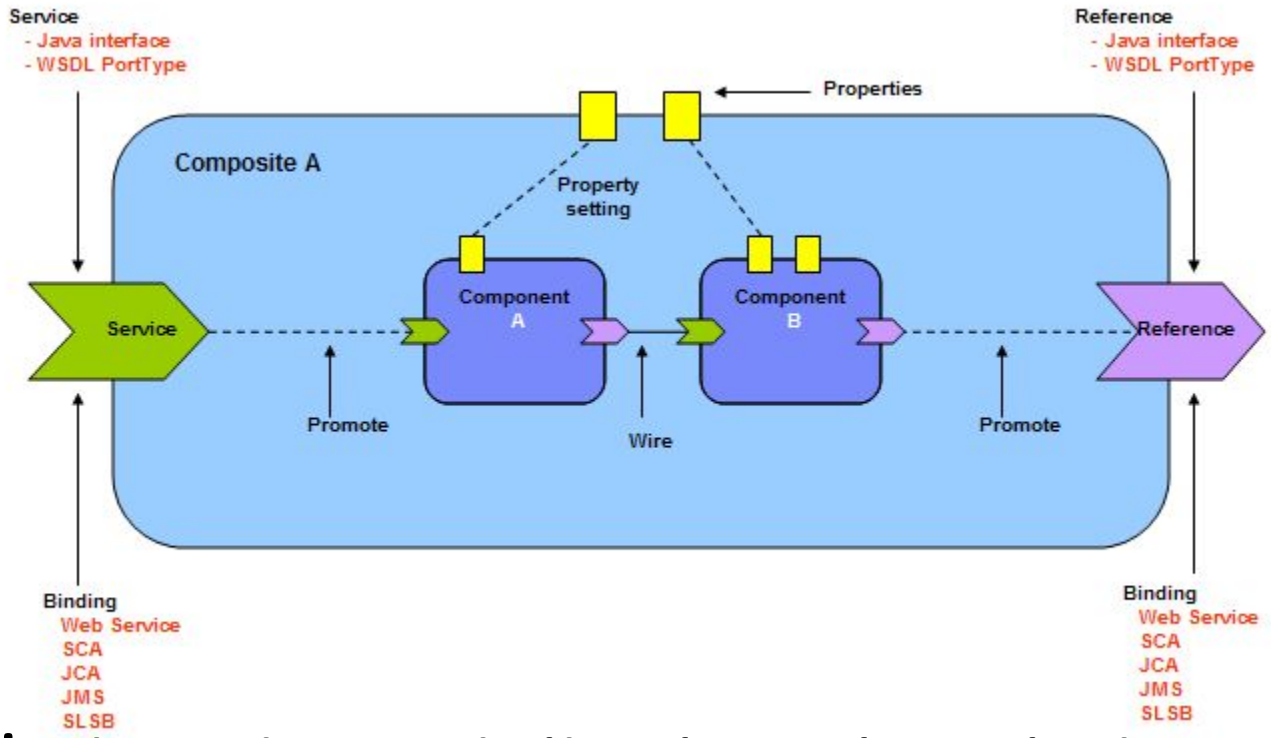
- Többféle szinten lehetséges
  - Pl. portolhatóság, interoperabilitás

Service Component Architecture

- *Open SOA Collaboration* tervezési modellje
  - SCA 1.0 2007 márciusában
  - Formális szabványosítás: OASIS
  - Jobbára Java környéki cégek

    BEA, Fiorano, IBM, Iona, Oracle, Red Hat, SAP, Siemens, Sun, Sybase, Tibco, …
- Microsoft hasonló koncepciója:
  Windows Communication Foundation
  - Inkább komm. platform, mint tervezési modell
- Portolhatóságot tűz ki (ez hiba)

SCA referencia modell



Kompozit: komplex, szolgáltatást nyújt, igénybe vesz

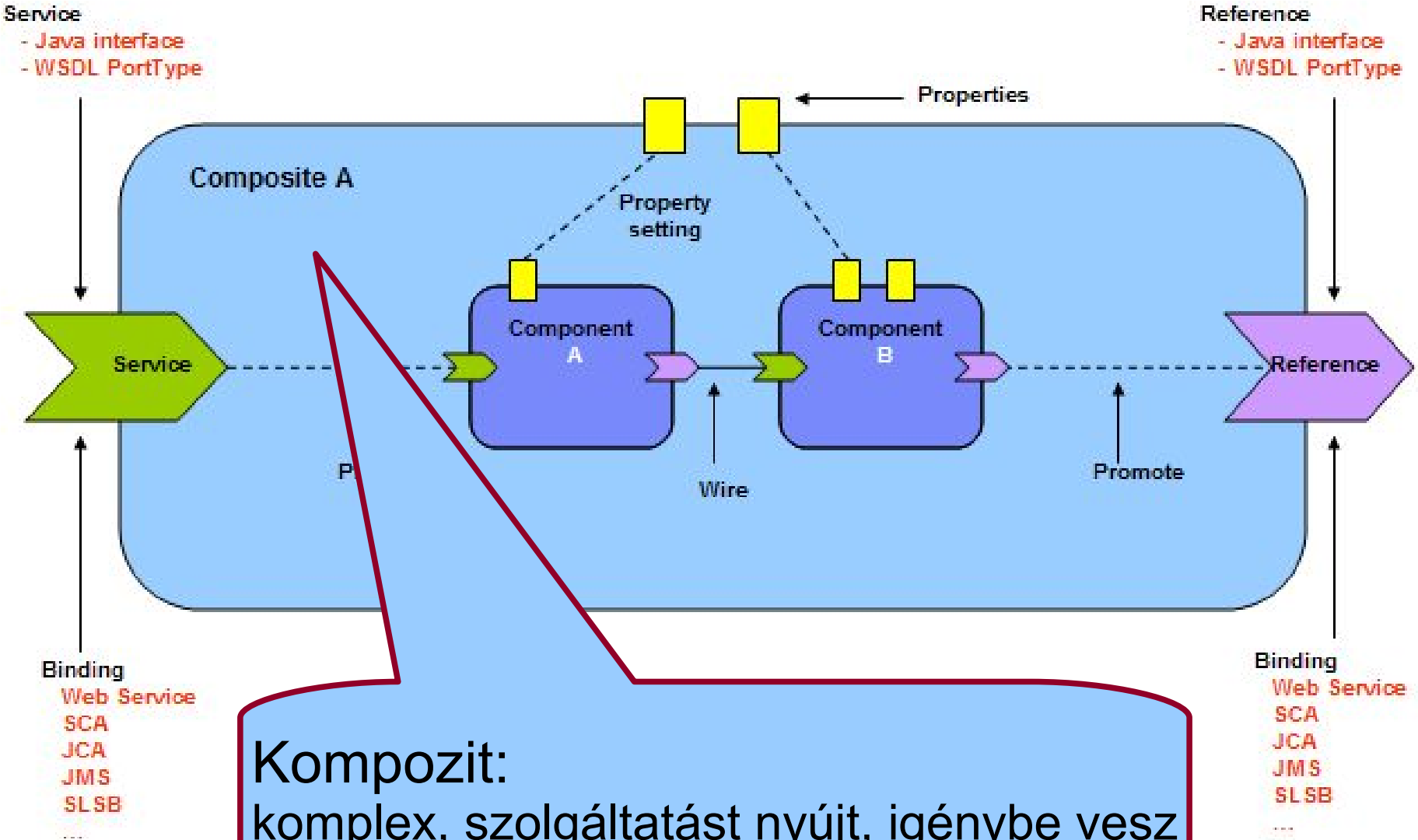
Szolgáltatás: publikus (absztrakt) interfész (+ huzalozás!)

Referencia: elvárt szolgáltatások (absztrakt) interfésze

Tulajdonság: paraméterek, konfiguráció

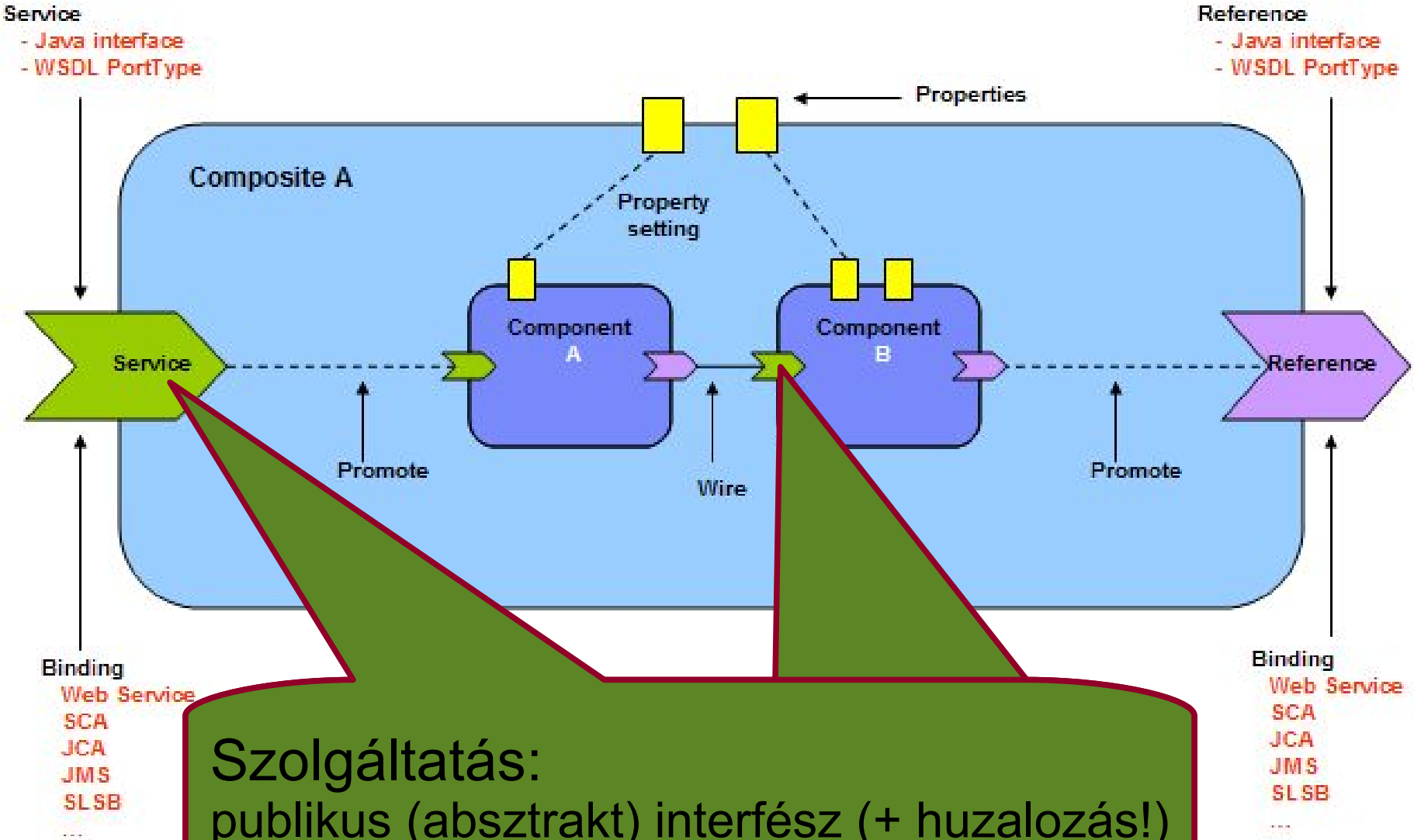
Komponens: a szolgáltatáslogika összetevője

SCA referencia modell

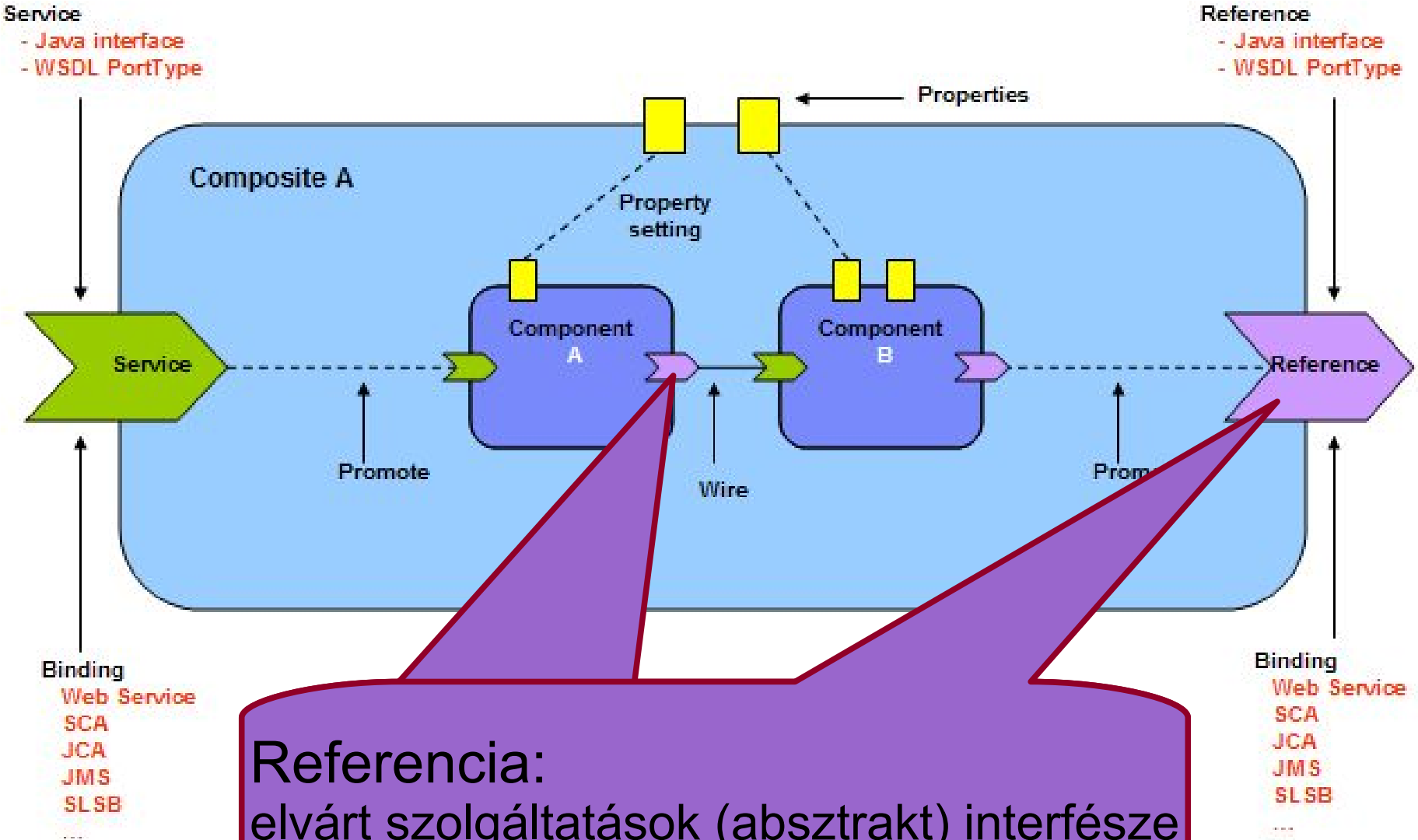


Kompozit:
komplex, szolgáltatást nyújt, igénybe vesz

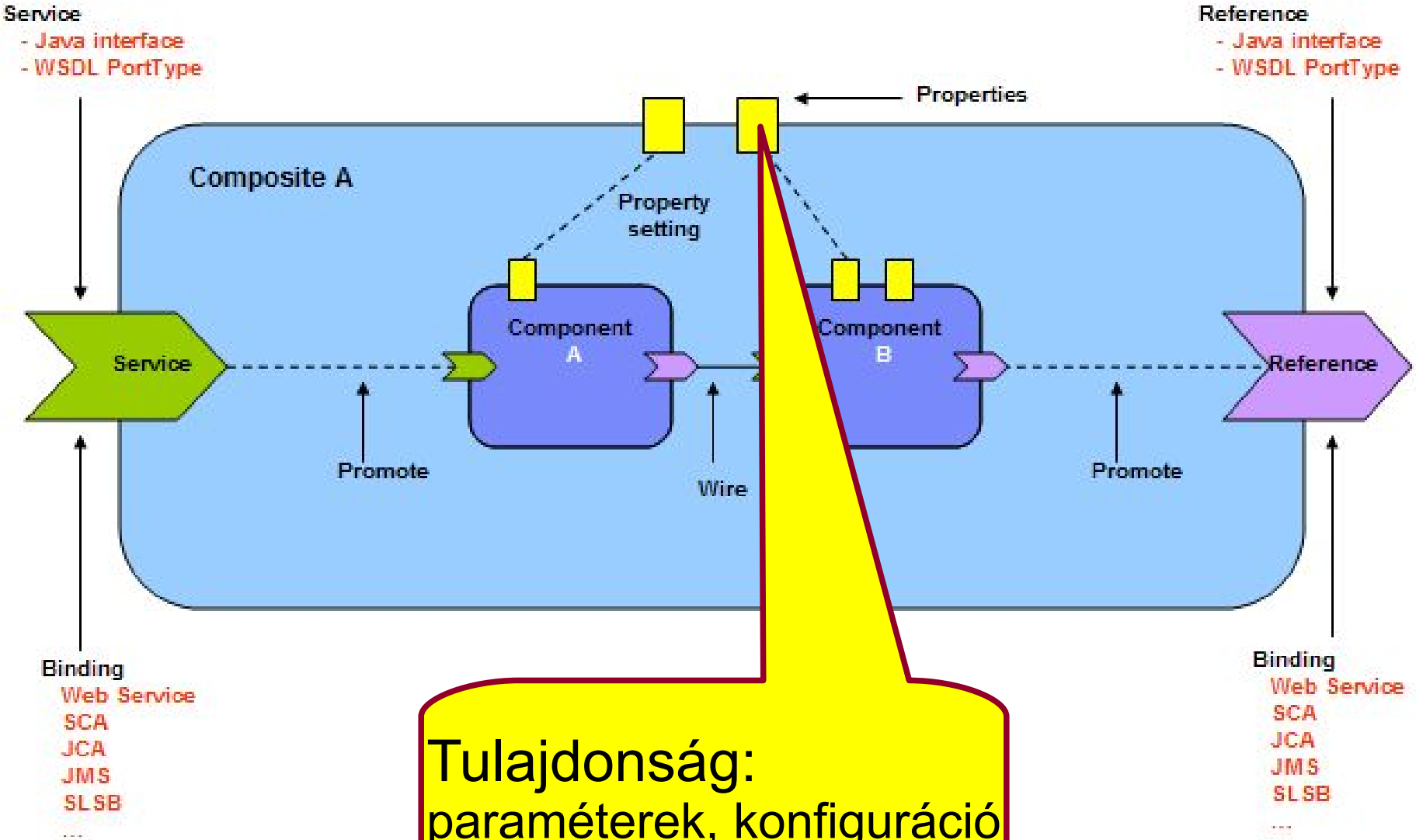
SCA referencia modell



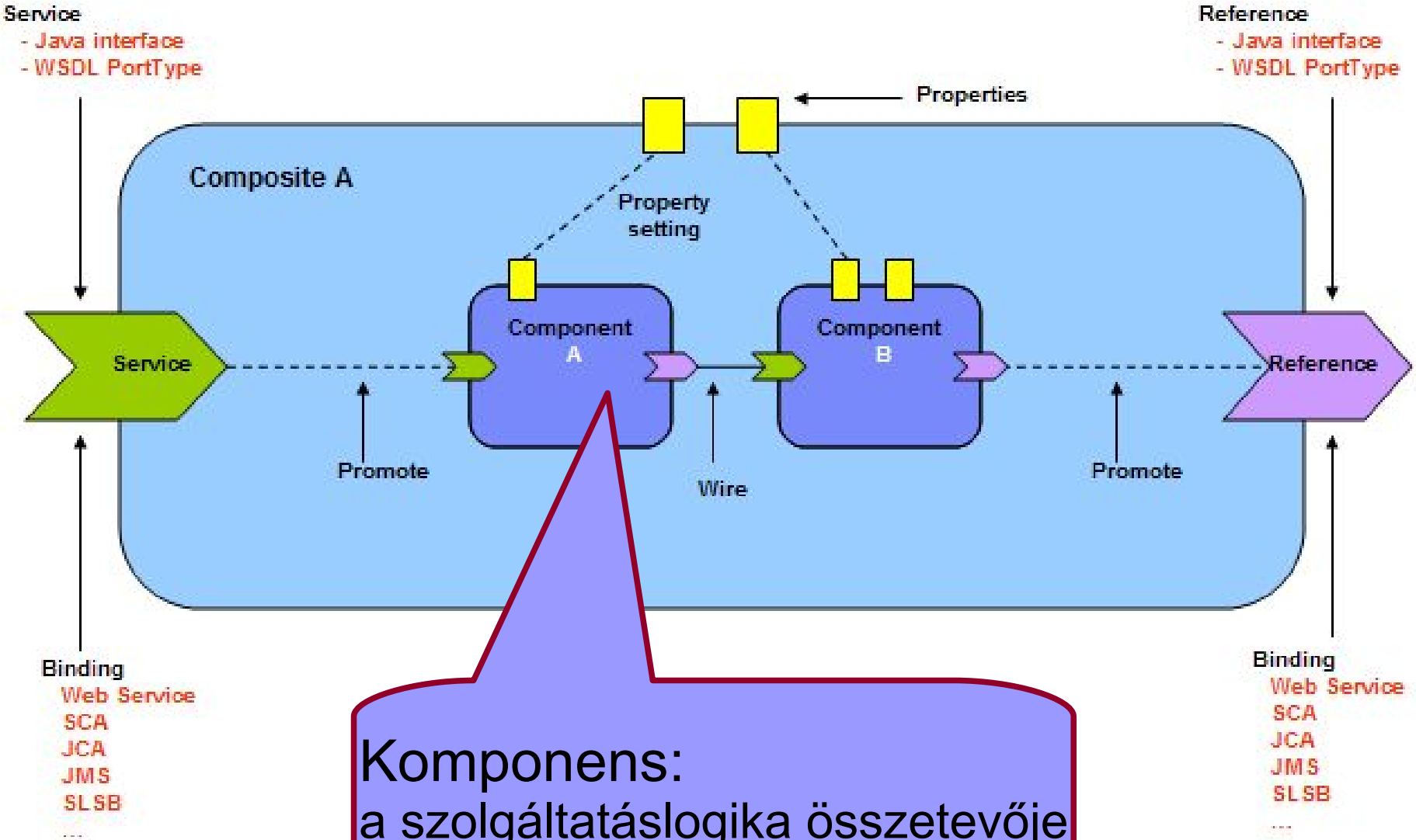
SCA referencia modell



SCA referencia modell



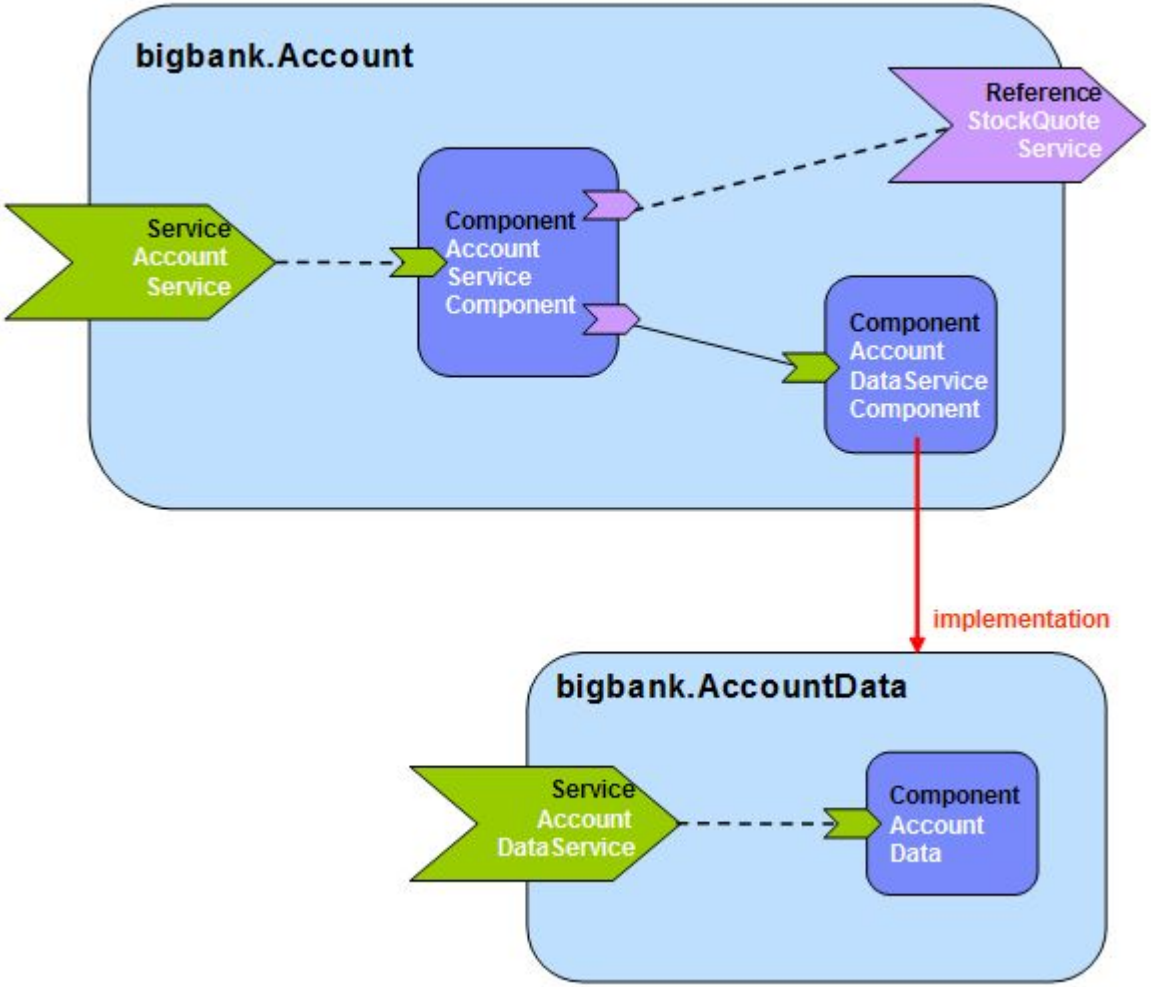
SCA referencia modell



Komponens:
a szolgáltatáslogika összetevője

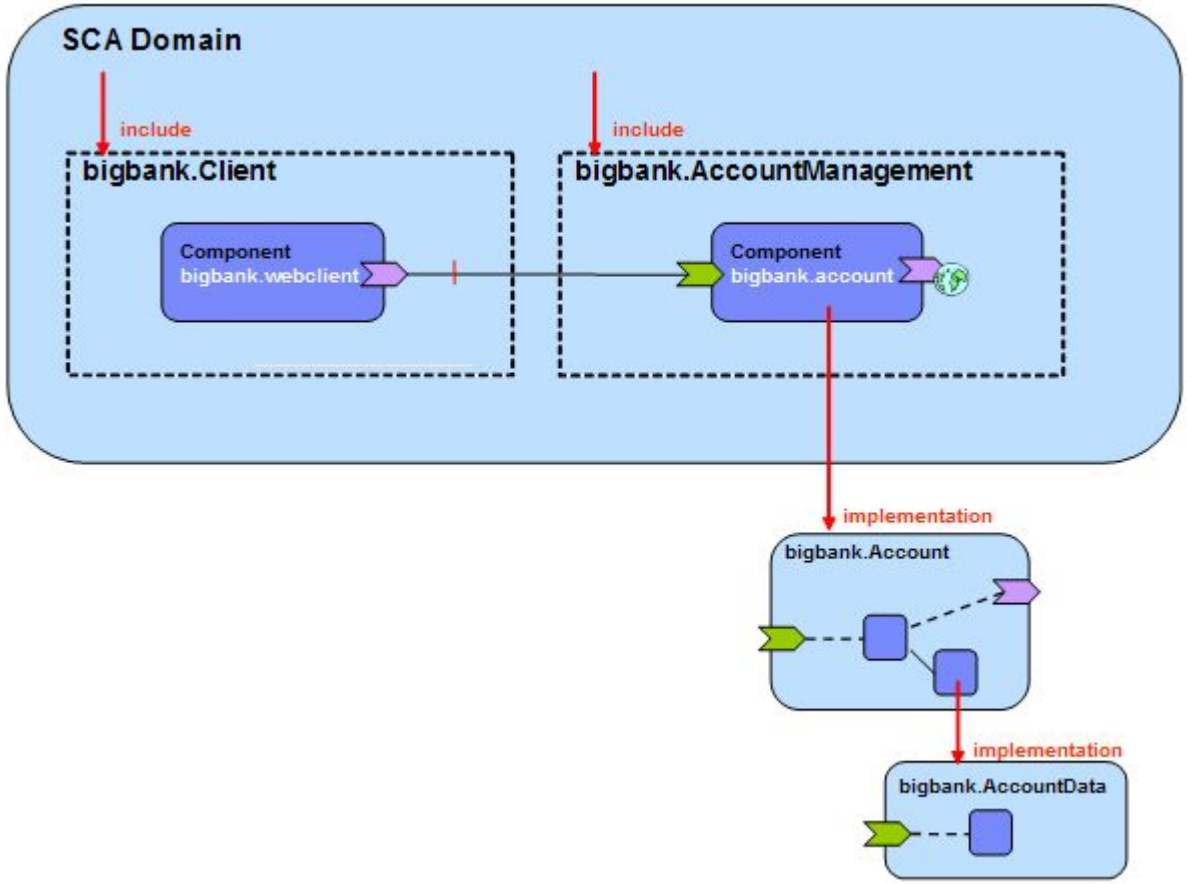
SCA példa

Banki rendszer



SCA példa

Banki rendszer (telepítési információk)



Kompozit

- Komplex, szolgáltatást nyújt, igénybe vesz
  - Milyen elemekből? (top-down jelleg)
  - Melyik szolgáltatást ki nyújtja?
  - Ki kivel kommunikál?
- Lehet „telepítési” egység (deployment package)
- Lehet „azonos környezetben futási” egység
  - A belső kommunikáció nem *távoli* jellegű
- Lehet „láthatósági” egység *(visibility scope)*
  - „include”-olt elemek mindig látnak és láthatóak

Szolgáltatás, referencia

- Nyújtott szolgáltatás publikus (abszt.) interfésze
- Elvárt szolgáltatások (absztrakt) interfésze

- Leírás nyelve tetszőleges
  - Ha *távoli kommunikáció*t feltételez, akkor *WSDL*-be fordíthatónak kell lennie
- Hosszú távú kapcsolódást külön jelölni
  - Session, conversation

- Implementáció:
  - WSDL, Java interfész/osztály, JMS binding, ...

Komponens

- A szolgáltatáslogika összetevője

- Alacsonyabb szintű kompozit (rekurzivitás)
- Java osztály, BPEL folyamat

- Implementáció:
  - BPEL, JSP, POJO, egyéb nyelvű program, selector, human task, business rule, business state machine, ...

SCA kötése hordozóprotokollhoz

- SCA platformfüggetlen, megvalósítás nem
  - Üzleti logika leválasztása az elérés megvalósításáról (de az is kell valahol!)
- Szolgáltatások/referenciák kötése (binding)
  - Hogyan hívható, hogyan akar hívni
  - SCA sokféle kötést ismer (WS, EJB, JMS, JCA, ...)
  - SCA alapú eszköz ezt bővítheti
- Vezetékezés megvalósítása: SCA motorra bízva
  - Mit köt össze a vezeték, mit ismer a motor
  - Ha interoperabilitás kell → WS

Adatáramlás szolgáltatások között

- Referenciák és szolgáltatások összehuzalozása
  - Szép, de önmagában nem elegendő
  - Adatstruktúra egyezést deklarál
  - Eltérő platformú szolgáltatások között probléma

- „Business Object on the Wire”
  - Java osztály
  - Service Data Object

Service Data Objects

- Open SOA Collaboration (SCA/SDO másik fele)
- Absztrakt adatreprezentáció → adatgráf
- Technológiafüggetlen
  - Heterogén adatforrások egységes kezelése
- J2EE adatmodelljénél egyszerűbb
  - Vele azonos tervezési minták alapján

- Implementáció:
  - Adatbázis (RDBS), EJB, XML dokumentum, WS, JCA, JSP, … alapú adatstruktúrák

SDO adatgráf

- Adat objektumok fába rendezett struktúrája

- Mediátor
  - Eredeti adatforrásból adatok kiszedése
  - Alkalmazásnak adatgráf átadása majd átvétele
  - Megfelelő formátumban eredeti adatforrásba
  - Alkalmazás semmit nem tud az adatforrásról

SDO adatgráf megvalósítása

- Az adatgráf szükség esetén kiírható XML-be
  - Átvitelhez, tároláshoz
  - Amúgy absztrakt adatreprezentáció!

- Adatgráf két fele
  - Az adatok egy „root object” alatt
  - A változtatások
    - Alkalmazás ide naplóz
    - Mediátor ez alapján frissíti a forrást
      (így könnyebb, sokszor nem is kell kiírni semmit)

SDO adatgráf elemek

- Egy adatelem → sok tulajdonság (properties)
  - Ezekhez „getter”-ek, „setter”-ek vannak
  - Mindenféle egyszerű típus lehet
    (összetett típust inkább már adatelem reprezentál)
  - XPath alapon lehet navigálni közöttük
- Metaadatok írják le a szemantikát
  - Általános alkalmazásoknak kellhet
- Adatelem definiálása
  - XSD, relációs modell,
    Java interfész, EJB, COBOL record,
    UML modell, üzenet definíciók

Mire jó az SCA/SDO?

- Referenciamodell
- Eszközök építhetőek rá
  - Modellező eszközök
  - Fejlesztő eszközök (Java5 annotációk)
  - Végrehajtó motorok

- Szolgáltatás-orientált fejlesztés
- Meglévő alkalmazások beillesztése
  - Bármi SCA komponensbe csomagolható

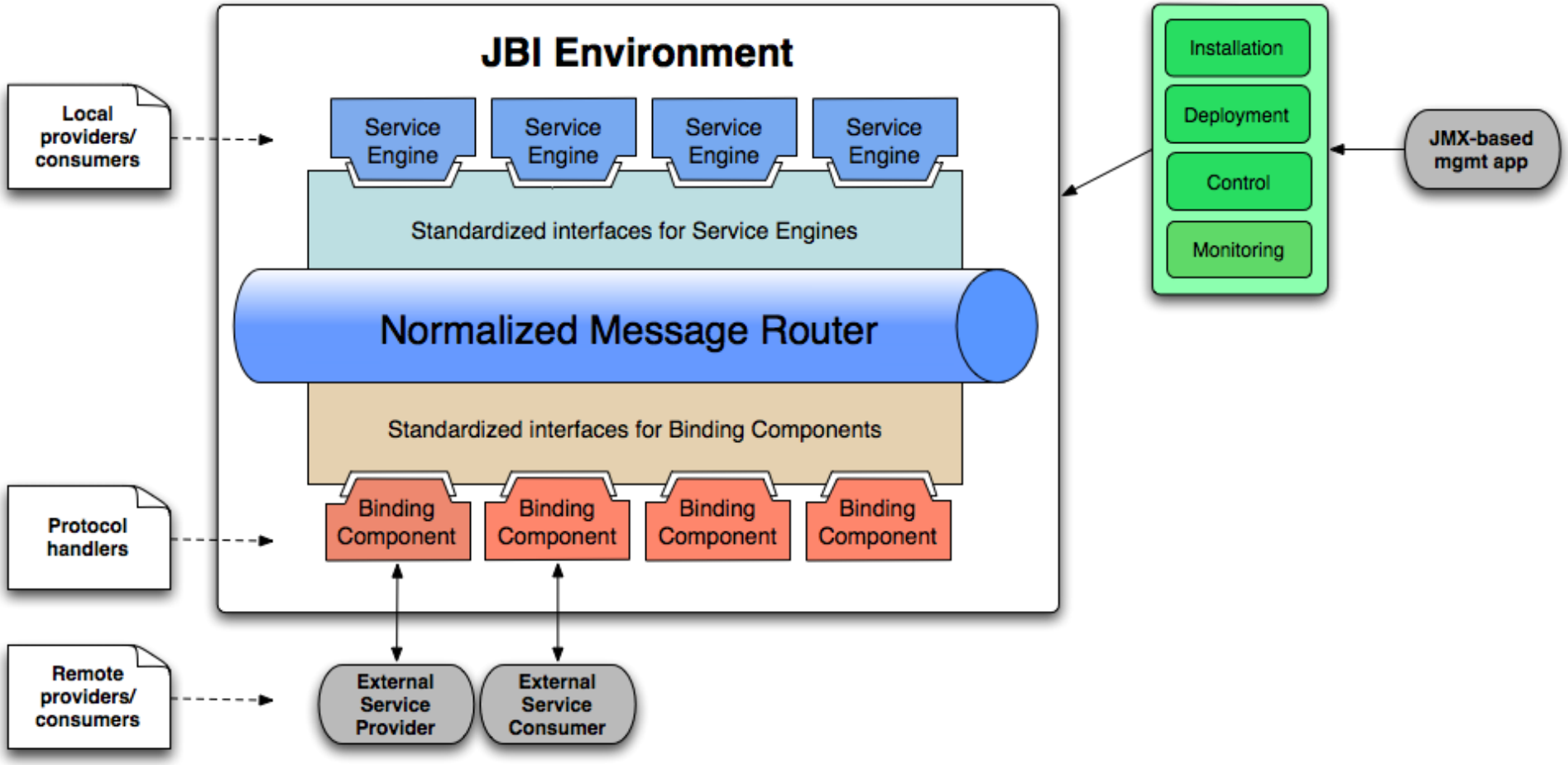
Kérdések

- Szolgáltatás-orientáltság platform függetlenül?
  - A WS is csak egy platform
  - A SOA alatt lehet más is
  - A szolgáltatás „örök”, a platform változik
- A Java platform hogyan támogatja a szolgáltatás alapú integrációt?
  - „Kell egy API”
  - Az alkalmazás szerverek támogassák ezt!

Szolgáltatás-orientált rendszerek & Java

- Java kiegészítése szolg. alapú integrációhoz?
  - JSR-208 (2005) Java Business Integration 1.0
  - JSR-312 (?) Java Business Integration 2.0
- WS + Java alapú EAI/B2B jellegű integráció (ESB)
- Java szabvány → csak egy API
- Nem vetélytársa az SCA/SDO-nak
- Implementációk
  - Apache ServiceMix, Fuse ESB, JBoss ESB, OpenESB, Sun Glassfish, TIBCO ActiveMatrix, Oracle Fusion, ...

JBI környezet



JBI – Normalized Message Router

- Normalizált üzenetek útvonal választója
  - Üzeneteket fogad SE/BC-ektől
  - Eljuttatja a címzett SE/BC-nek
  - Tipikus ESB szolgáltatások (is)
  - Normalizált üzenetekkel dolgozik
- Támogatott üzenetváltás típusok
  - egyirányú, megbízható egyirányú
  - kérés – válasz, kérés – opcionális válasz

JBI – Normalized Message

- Normalizált üzenetek
- 3 része van:
  - Maga az üzenet XML-ben
    - megfelel a megfelelő WSDL-ben leírt típusnak
  - Tulajdonságok/Metaadatok
    - üzenetváltási, biztonsági, tranzakciós infók
  - Csatolt adatok (nem XML)
    - kezelése problémás → önhordóság?
- Normalizálás → egységes feldolgozás
  - ellenőrzés, szűrés, manipuláció, transzformáció

JBI – Service Engines

- (Belső) szolgáltatások
  - üzleti komponensek (forrás, címzett)
  - kommunikációs komponensek (szűrés, ellenőrzés, transzformáció, ...)
- Az adott JBI környezetben fut
- BPEL, Java vagy más alapú implementáció

JBI – Binding Components

- Csatolók külső komponensekhez
- Inkább csak logikailag különböznek a SE-ektől
  - Logikailag tényleg különböznek
  - Az NMR szempontjából nem különböznek
- Az adott külső komponens és a JBI környezet közötti kapcsolat kezelése
  - Hozzáférés kezelés, formátumok, ...
  - Kommunikációs minták
  - Átviteli protokollok (címzés, titkosítás, …)
    - WS alapú, JMS/MQ alapú, fájl alapú, ...

JBI – Problémák

- ESB gyártók szabványosítanak
  - Felhasználók (integrátorok) nélkül
- Túl sok megkötés
  - Minden adat XML alapú (hatékonyság?)
  - Minden adattranszformáció XML alapú
  - Interfész leírások WSDL alapúak
  - Kommunikáció üzenet alapú (üzenet folyamok?)