PA165
Integrační technologie

Jiří Kolář

Představení

- Jiří Kolář
  - Doktorský student u Doc. Pitnera
  - Člen laboratoře LaSArIS
    - Zaměření na BPM a BAM
    - BPM for SME
    - Projekt MEDUSY
    - Zájmy v oblasti BPM a systémové integrace

Obsah dnešní přednášky

- Motivace
- Úrovně integrace
- Component containery
  - JBI , OSGi
- Systémová integrace
  - Messaging – přístupy, standardy, technologie
  - Integrační vzory
  - Typické architektury middleware
  - Integrační scénáře

Co si odnést ?

- Ideu co to je integrace a proč je jí třeba
- Hrubou představu o tom jak a proč integrovat
  - Komponenty systému
  - Systémy navzájem
- Přehled o hlavních technologiích v této oblasti
- Přehled o základních architektonických přístupech k integraci

Motivace - Proč integrovat?

- Komponentový přístup k vývoji SW
  - Maximální nezávislost komponent
  - Universální přenositelné komponenty
  - Zapouzdření komponent
  - Minimalizace veřejných rozhraní
  - Poskytování služeb
  - „Prodej komponent“
- Propojení platforem
  - Různé jazyky, běhová prostředí a OS

Motivace - Proč integrovat? (cont.)

- Legacy systémy
  - Zastaralé, ale dobře fungující komponenty
  - Wrappery poskytující navenek moderní rozhraní
- Propojení autonomních Enterprise systémů
  - Více Enterprise systémů v rámci podniku
  - Poskytování funkcionality navenek
    - Zákazníkům
    - Business partnerům
  - Fůse a štěpení podniků, outsourcing

Problémy v rámci integrace

- Nekompatibilní protokoly
- Změna veřejných API
- Validace zpráv – chyby v přenosu dat
- Bezpečnost
- Performance

Úrovně integrace – Aplikační úroveň

- Integrace aplikačních komponent uvnitř systému
  - Autonomně fungující komponenty komunikující mezi sebou uvnitř containeru
  - Např. Uvnitř aplikačního serveru (J2EE)

- Integrace na úrovni OS
  - Standardizovaná komunikace jádra OS s aplikacemi
  - Komunikace aplikací mezi sebou
  - Např. D-BUS Linux

Úrovně integrace – Systémová úroveň

- Propojení komponent Enterprise systémů
  - EAI – Enterprise Application Integration
  - Vysoce heterogenní prostředí
  - Komponenty uvnitř systémů
    - Finance, E-commerce, ERP, Internal systems
- Propojení systémů samotných
  - Poskytování služeb mezi autonomními systémy
    - Business partners, Customers, Goverment ..
  - Typicky pomocí Web Services, public APIs
  - Často nutnost orchestrace = BPM

BPM: orchestrace vs choreografie (viz předmět PV207)

- Choreografie
  - Pevně daná posloupnost interakce mezi komponentami
  - Komponenty o sobě navzájem ví
  - Aplikační logiku zajišťují služby/aplikace
- Orchestrace
  - Služby jsou orchestrovány „dirigentem“
    - Business Process Execution engines
    - Business Rules
  - Aplikační logiku zajišťuje dirigent

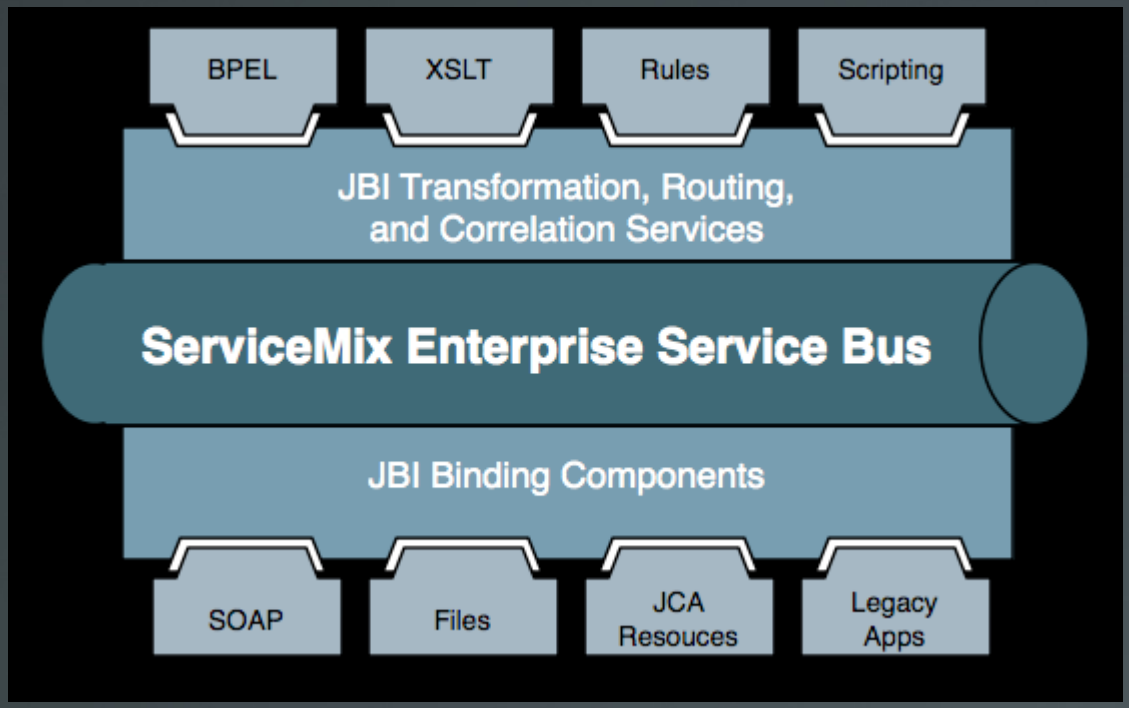
Component containery v Javě

- Slouží jako běhové prostředí pro komponenty
- Běží jako součást aplikačního serveru
- Obsluhují životní cyklus komponent,typicky:
  - Deploy
  - Init
  - Start
  - Stop
  - Undeploy

JBI

- První pokus o standardizaci meta-containeru
  - Navrženo SUN Microsystems
    - Implementováno v Open ESB a Apache Service Mix
    - Nepříliš podpořeno ostatními
  - Definuje architekturu připojitelných komponent
    - Nedefinuje komponenty samotné
    - Komponenty komunikují pomocí messagingu
    - Komponenty samy mohou sloužit jako containery
  - Příklad komponenty: BPEL engine

Architecture - Service Bus (ESB)



OSGi framework

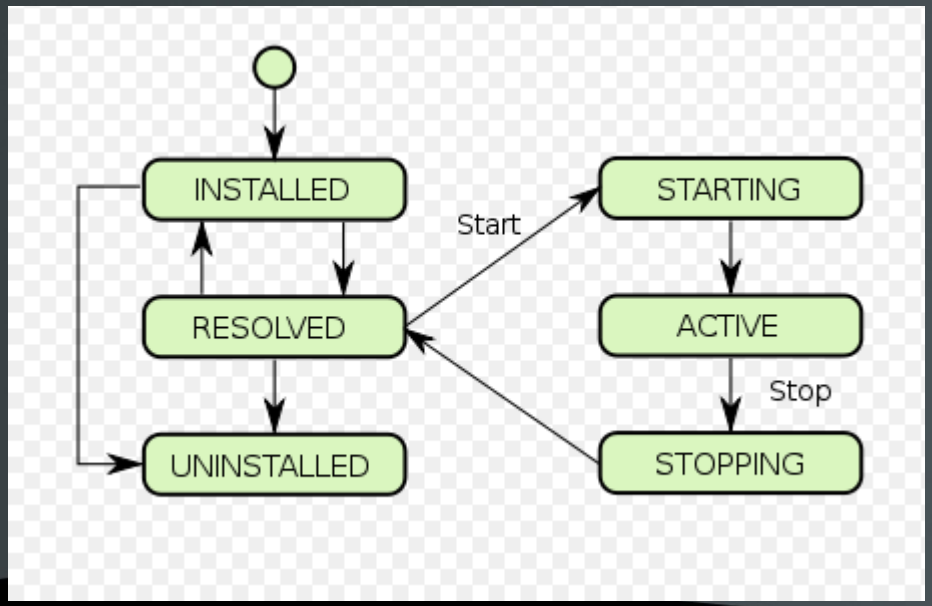
- Aktuální standard pro microcontainery
  - Původně určeno pro systémy chytré domácnosti :)
  - Specifikace je universální
    - J2EE
    - Java SE
    - Java ME
- Použito/plánováno ve všech AS J2EE
  - Obdoba JBI
  - POJO opět v módě

OSGi framework

- Dynamický komponentový model
  - Podpora životního cyklu komponent
  - Možnost manipulace s komponentami za běhu
  - Dynamický classloading
- Komponenty poskytují služby
  - Service management
  - Service registry

OSGi framework

- Komponenty (bundles) jsou jednoduše *.jar
  - Popis komponenty v MANIFEST.MF
    - Závislosti na ostatních komponentách
    - Verze komponenty
- Component lifecycle

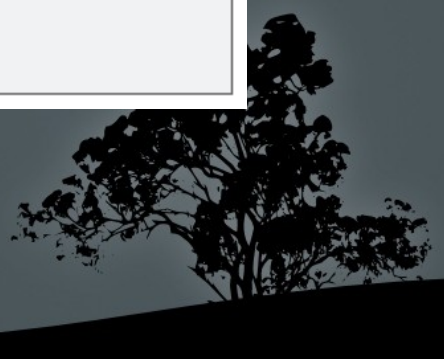


OSGi framework

- Komponenty (bundles) jsou jednoduše *.jar
  - Popis komponenty v MANIFEST.MF

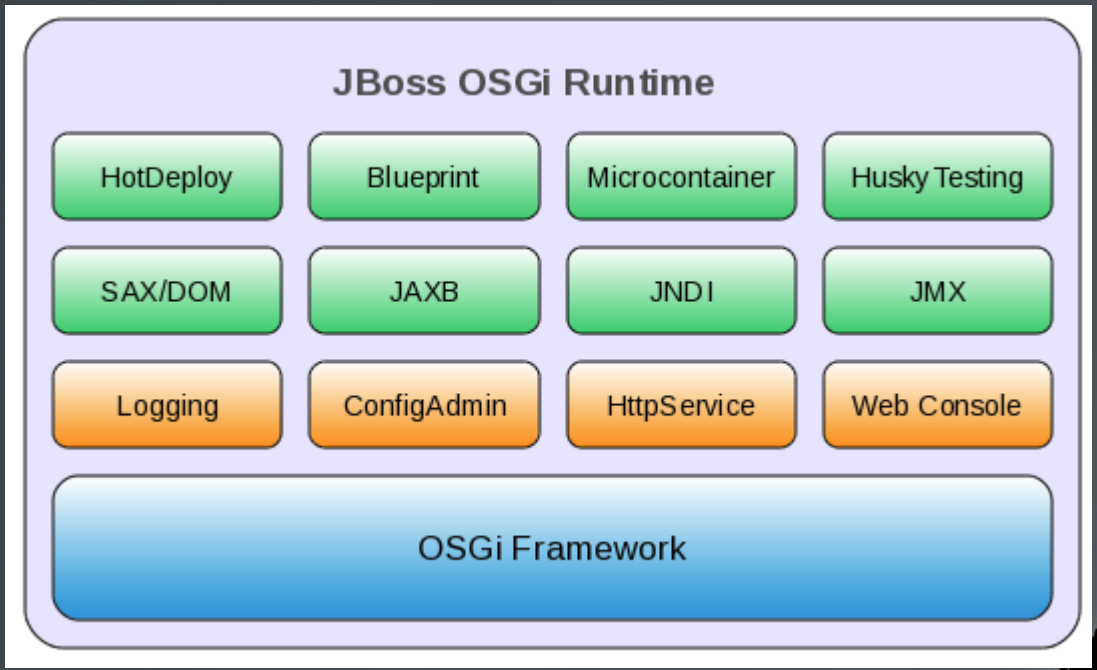
```
Manifest-Version: 1.0
Bundle-ManifestVersion: 2
Bundle-Name: HelloWorld Plug-in
Bundle-SymbolicName: com.javaworld.sample.HelloWorld
Bundle-Version: 1.0.0
Bundle-Activator: com.javaworld.sample.helloworld.Activator
Bundle-Vendor: JAVAWORLD
Bundle-Localization: plugin
Import-Package: org.osgi.framework;version="1.3.0"
```

```
package com.javaworld.sample.helloworld;
import org.osgi.framework.BundleActivator;
import org.osgi.framework.BundleContext;
public class Activator implements BundleActivator {
    public void start(BundleContext context) throws Exception {
        System.out.println("Hello world");
    }
    public void stop(BundleContext context) throws Exception {
        System.out.println("Goodbye World");
    }
}
```



Příklad OSGi containeru: JBoss Microcontainer

- Standardní OSGi container
- Obsahuje základní bundles



OSGi framework

- Hype dnešní doby
- Konsenzus velkých hráčů

  Glassfish (SUN), SpringSource, JBoss (RedHat),
  WebSphere (IBM), Weblogic (Oracle)..
- Důsledně komponentový přístup
- Idea ”znova a jednodušeji”

- Nicméně to trvalo 10 let a výsledek je stejně pochybný

OSGi vs JBI

- JBI
  - Klade důraz na integraci a transformaci dat
  - Bohužel už asi neprorazí
  - Definuje NMR a komunikaci mezi komponentami
- OSGi
  - Universální modularní framework
  - Existují i specifkace pro více jazyků než Java
  - Lightweighted

OSGi based products

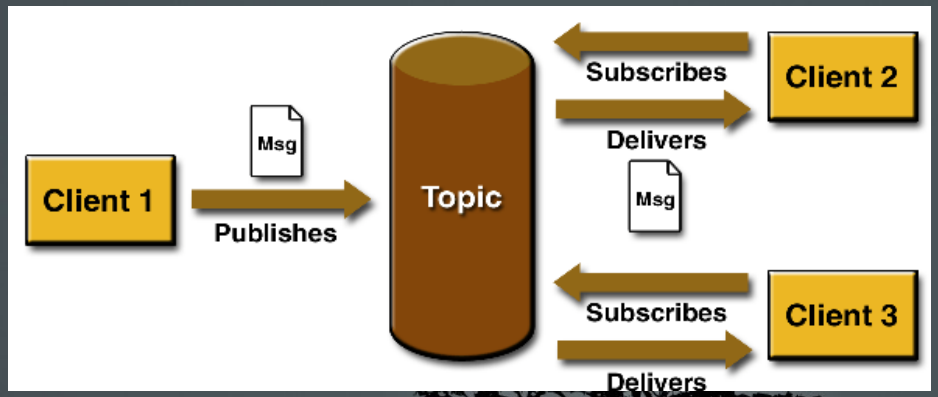
- Equinox– referenční implementace OSGi (EclipseRT)
- Virgo – OSGi container (WS/AS) (EclipseRT)
- JbossAS
- Apache ServiceMix 4.X.X (paralelně s JBI)

Systémová integrace

- Propojení autonomních systémů
- Různé komunikační protokoly
- Různá rozhraní
- Často použití Web Services
- Integrační vzory
- Architektonické přístupy k middleware
  - Message Broker (Service Hub)
  - Service Bus (ESB)

Messaging systems

- Systémy pro posílání zpráv
  - Messsaging modely
    - Point to Point
    - Publisher-Subscriber
  - Distribuovaný messaging (clustering)
  - Persistence
  - Monitoring
  - Management

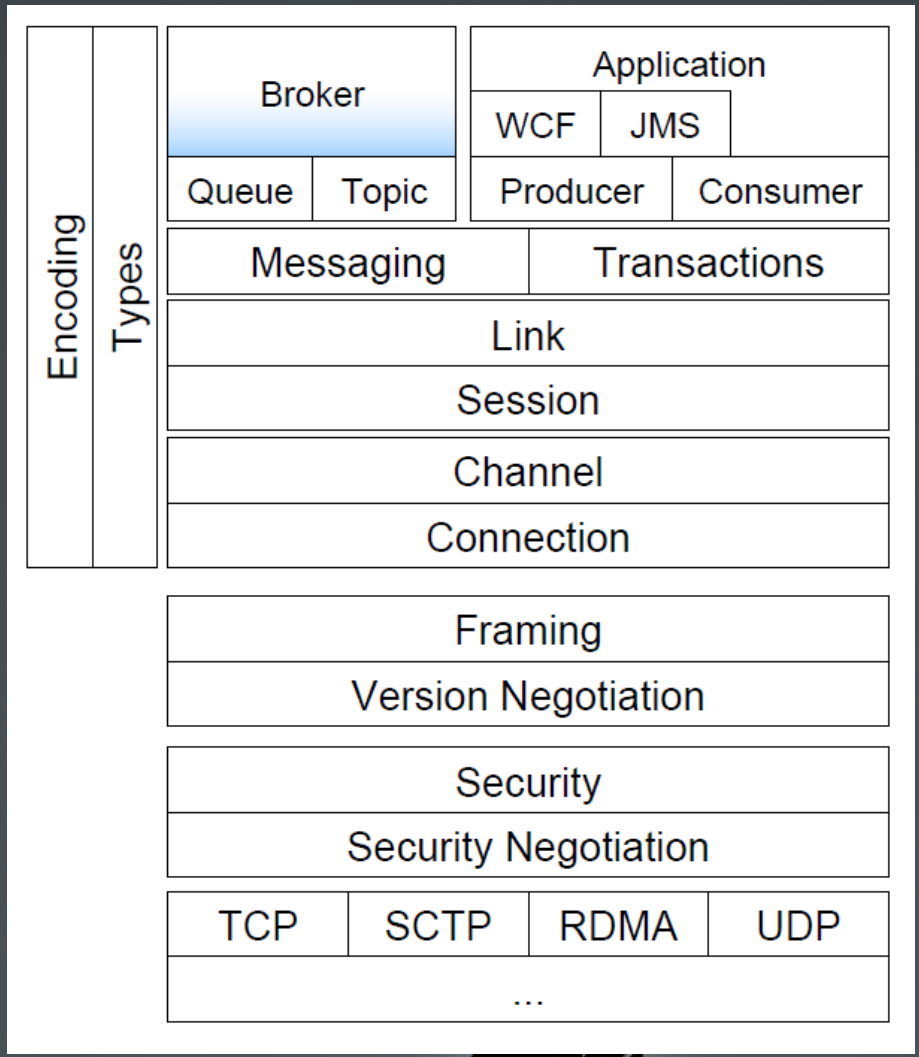


JMS – Messaging API Javy

- Standardizované messaging API
- High level, odstiňuje od mnoha detailů
- Dnes implementováno většinou messaging systémů
- Používáno uvnitř Aplikačních serverů
- Určeno primárně pro aplikační integraci

AMQP - Messagingový protokol

- Standard vytvořený konsorciem
  - Wire level protocol
  - Definuje i API
- Qpid – Apache
- RabbitMQ – Vmware
- Red Hat Enterprise MRG (Qpid-based)



- Bank of America, N.A., Barclays Bank PLC, Cisco Systems, Credit Suisse, Deutsche Börse Systems,Goldman Sachs, INETCO Systems Limited, Informatica Corporation, JPMorgan Chase Bank Inc. N.A, Microsoft Corporation, Novell, Progress Software, Rabbit Technologies Ltd., Red Hat Inc., Software AG, Solace Systems Inc., Tervela Inc., TWIST Process Innovations Ltd, VMware, Inc., WS02 Inc. and 29West Inc.

Web Services

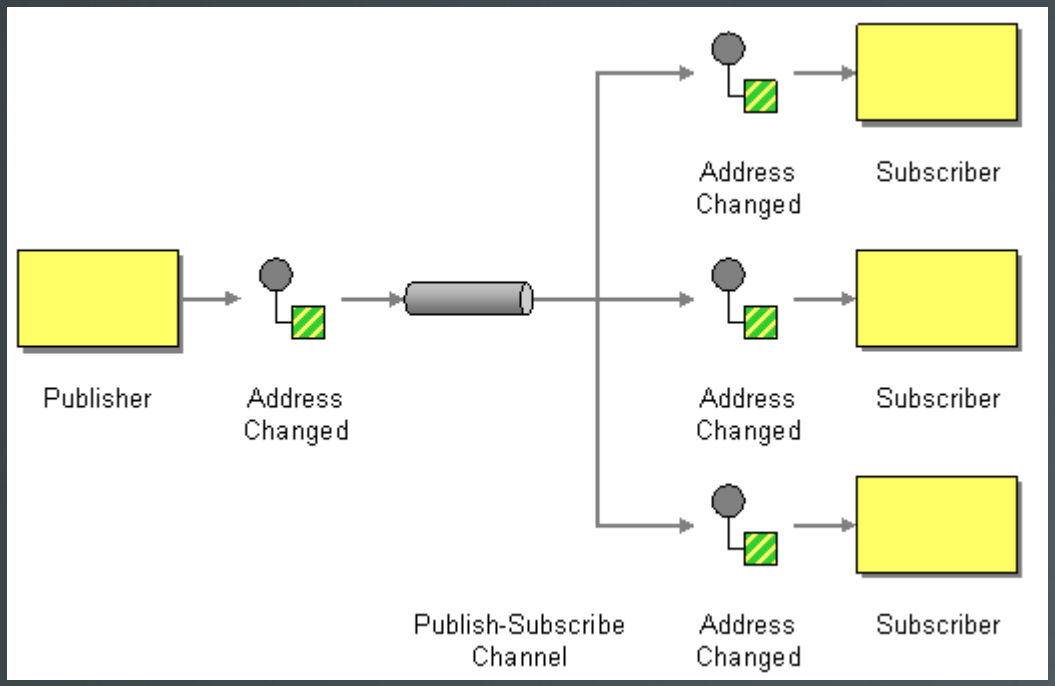
- Standardizované rozhraní
- Universální, vhodné pro heterogenní prostředí
- Použití při systémové integraci
- Potřeba ”service registry”
- Potřeba orchestrace (BPM)
- Značná režie, nevhodné při požadavkům na výkon
- Různé protokoly SOAP, REST ...

Integration patterns

- Návrhové vzory pro integraci
  - Po vzoru GoF
  - Pokus o standardizaci
  - Popisují ”best practices” v integraci
  - Podporovány integračními platformami
  - Podporovány vývojovými prostředími

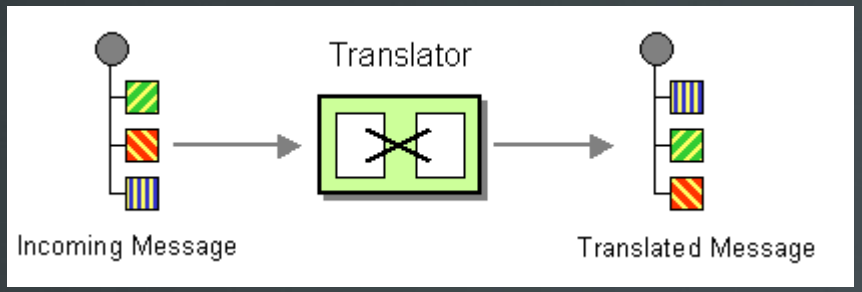
Integration patterns

- Vzor Publisher - Subscriber

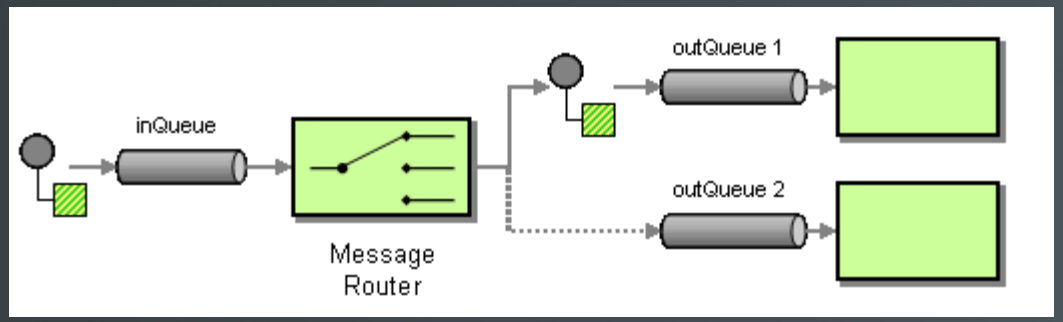


Integration patterns

- Vzor translator



- Vzor message router

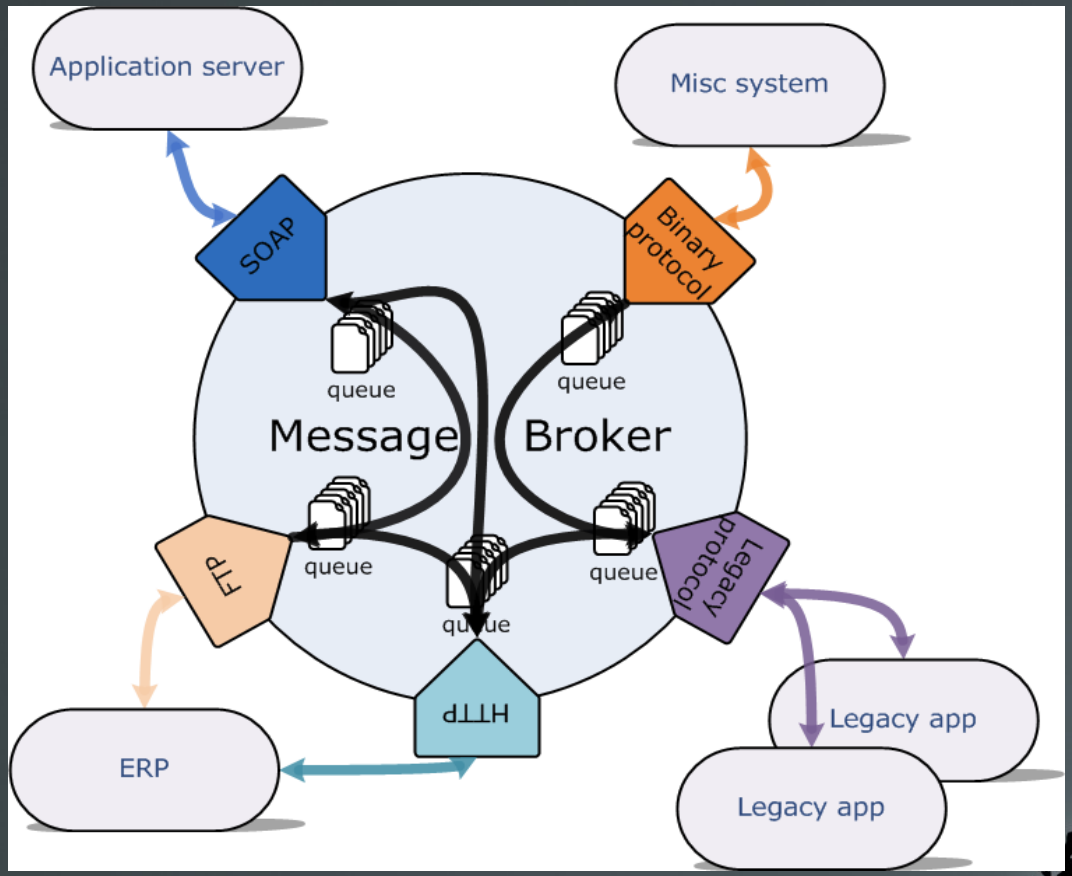


Architektury middleware

- Architektonické koncepty
- Popisuje logiku integračního řešení
- Různé vzory mohou být postaveny na stejné fyzické implentaci
- Vice-versa
- Za komplexním konceptem se může skrývat poměrně jednoduchá implementace

Architecture - Service hub

- Překlad mezi komunikačními protokoly



Architecture - Service hub (cont.)

- Starší přístup,ale stále hojně využívaný

  + Menší režie

  + Jednodušší zpráva

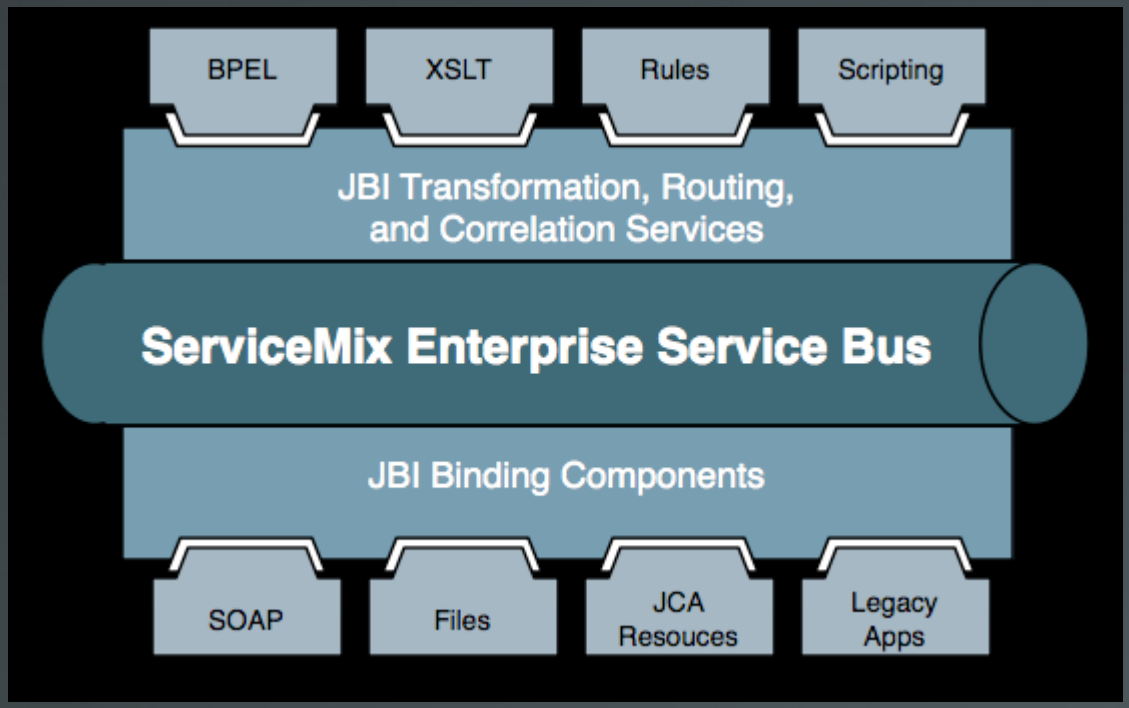
  - Menší prostor pro transformace zpráv

  - Nevhodné pro složitější topologie

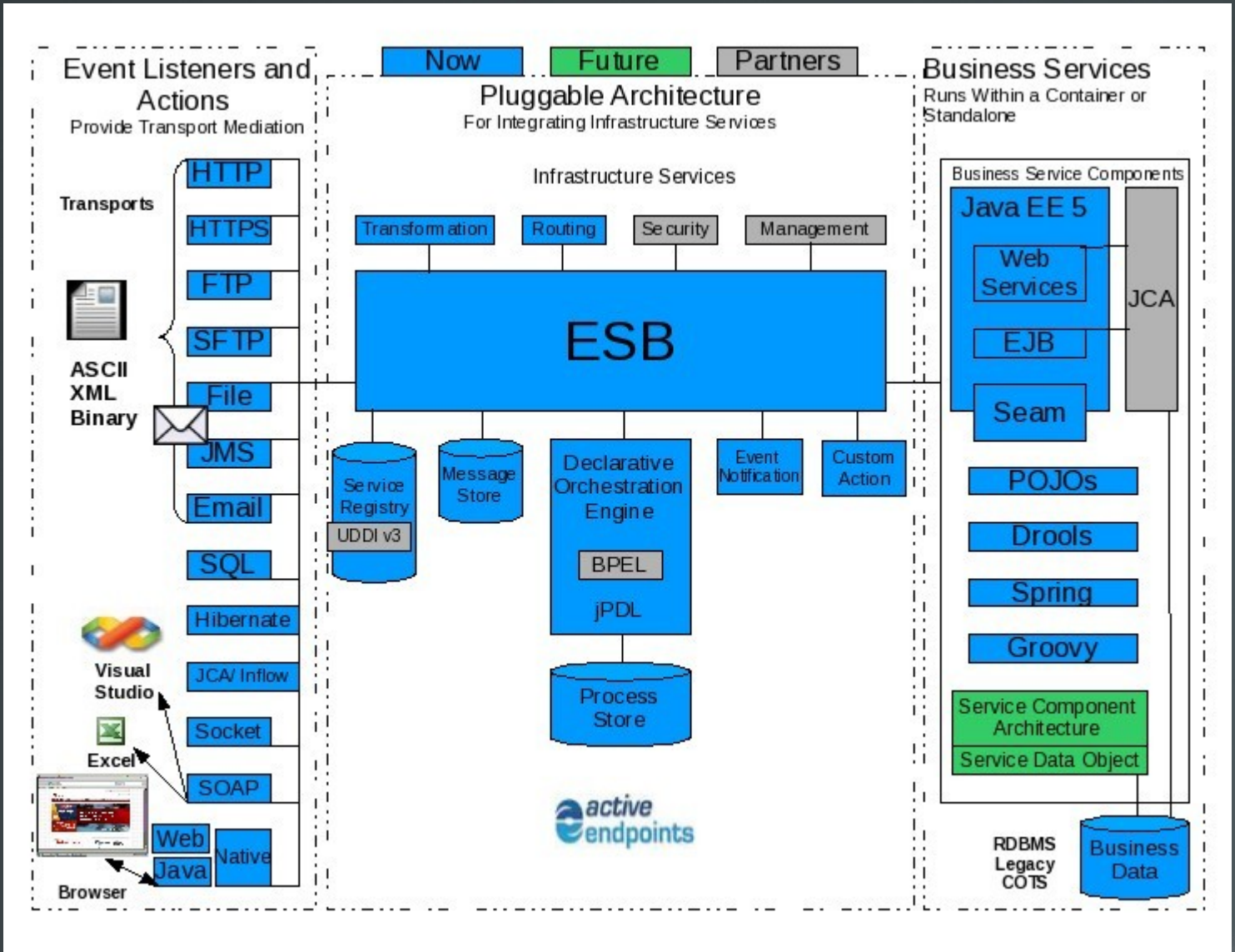
Architecture - Service Bus (ESB)

- Model sběrnice
  - Zprávy jsou překládány do ”standardního formátu” (Normalized messages)
  - Překlad z jednotlivých protokolů pomocí adaptérů (binding components)

Architecture - Service Bus (ESB)



Architecture - Service Bus (ESB)



Service Bus (ESB)

- Model sběrnice
  - Primárně v prostředí Javy a Web Services

    + Flexibilnější – Lepší podpora integračních vzorů

    + Lepší pro clustering a složitější topologie

    - Větší režie

ESB products

- Open source
  - Apache ServiceMix
  - Apache Synapse (micro ESB)
  - Jboss ESB
  - OpenESB
  - Mule ESB
- Commercial
  - Websphere ESB
  - Oracle ESB

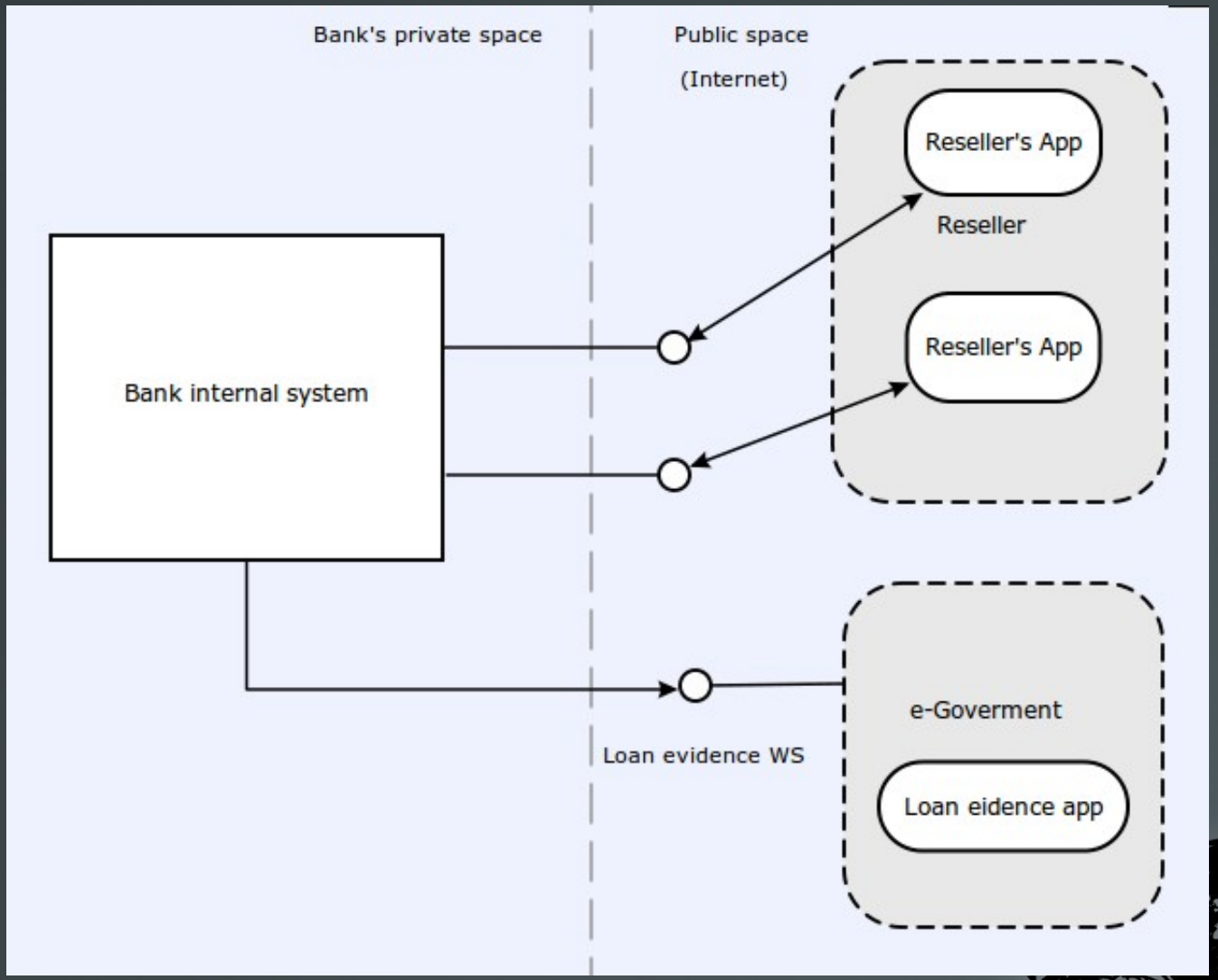
OSGi na ESB

- OSGi je separátní framework
- Služby komponent dostupné jen uvnitř containeru
- Lze jej však připojit k ESB
- Tak mohou aplikace pomoci běžných protokolů volat služby komponent uvnitř OSGi containeru

Scénář - půjčky

- Banka provozuje interní systém na správu půjček.
- V rámci systému existuje aplikace na výpočet výhodnosti půjček.
- Banka chce umožnit svým resellerům používat tuto aplikaci.
- Reselleři si chtějí tuto funkcionalitu integrovat do jejich vlastních systémů.
- Banka chce ověřit profil klientů ve státní evidenci dlužníků

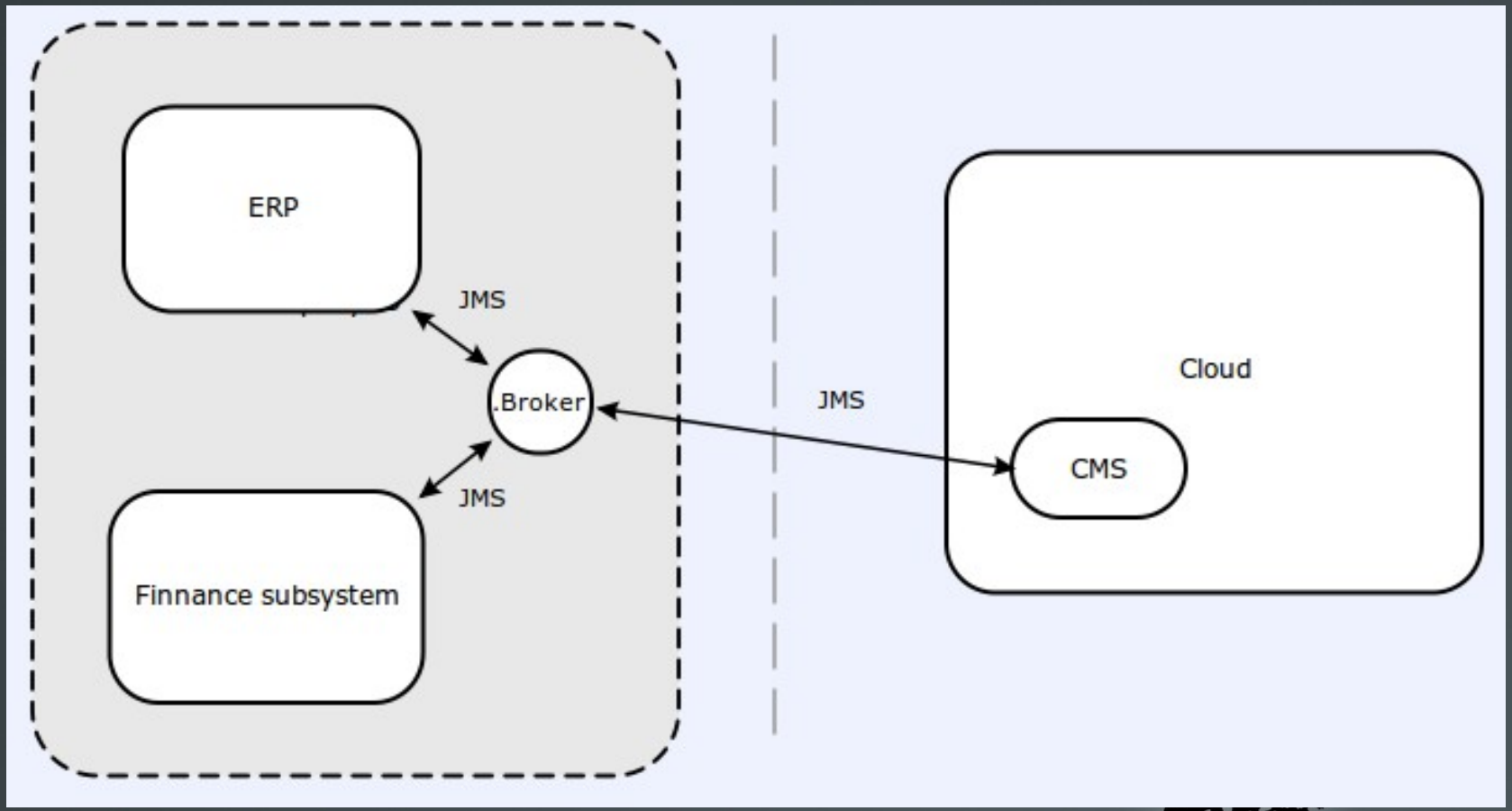
Scénář - půjčky



Scenář - cloud

- Firma provzuje CRM jako cloudovou službu
- V rámci vlastního systému provozuje účetní systém a e-store
- Firma chce integrovat tyto aplikace

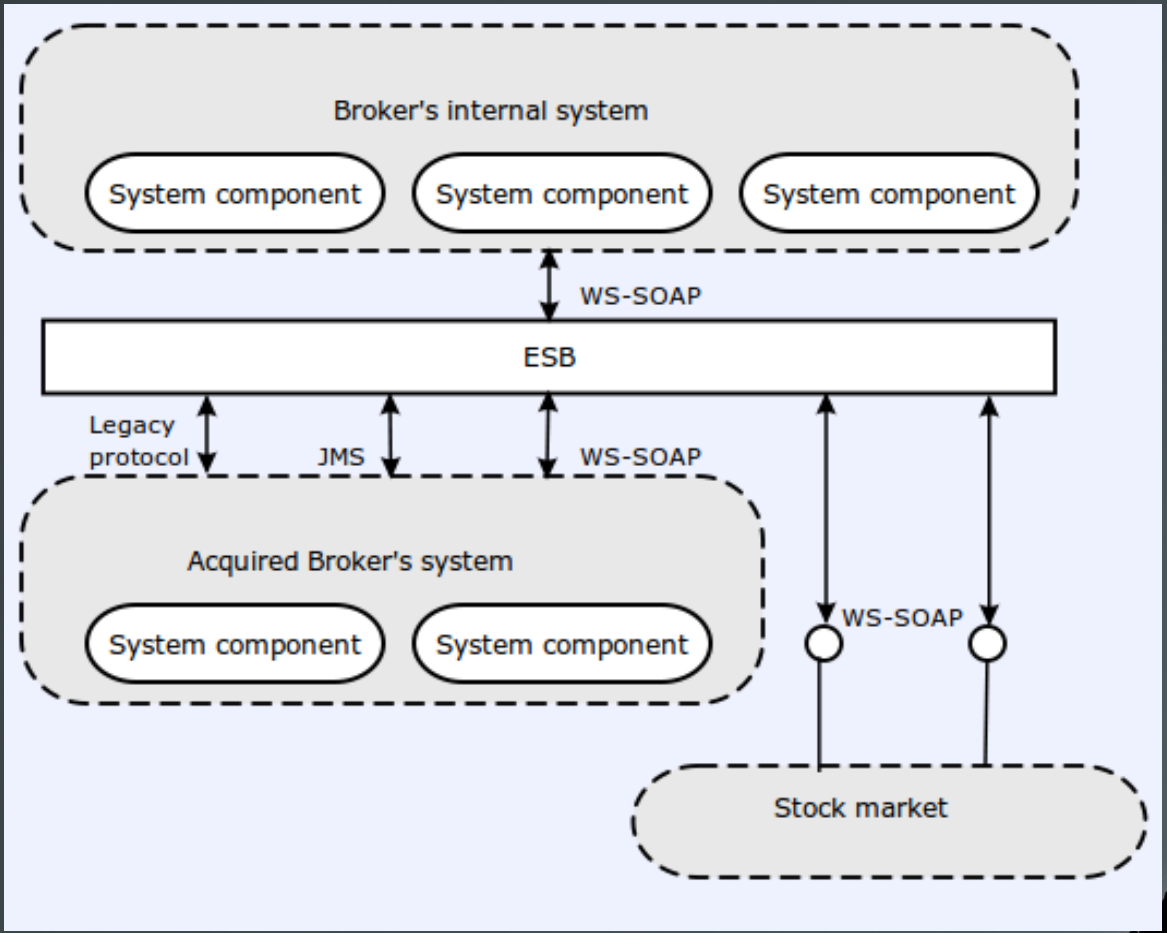
Scénář - cloud



Scénář - Fůze

- Brokerská společnost koupí svého konkurenta
- Konkurent provozuje dobře fungující systém na manipulaci s akciemi
- Je potřeba používat tyto systémy současně
- Je zapotřebí velká propustnost a rychlost integračního řešení

Scénář – Fůze


Broker's internal system
System component
System component
System component
WS-SOAP
ESB
Legacy protocol
JMS
WS-SOAP
Acquired Broker's system
System component
System component
WS-SOAP
Stock market

FIN

Otázky?