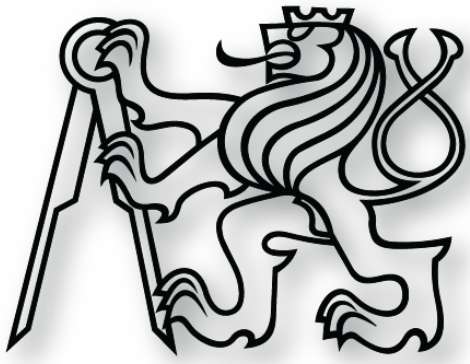


Service Oriented Architecture

Enterprise Service Bus

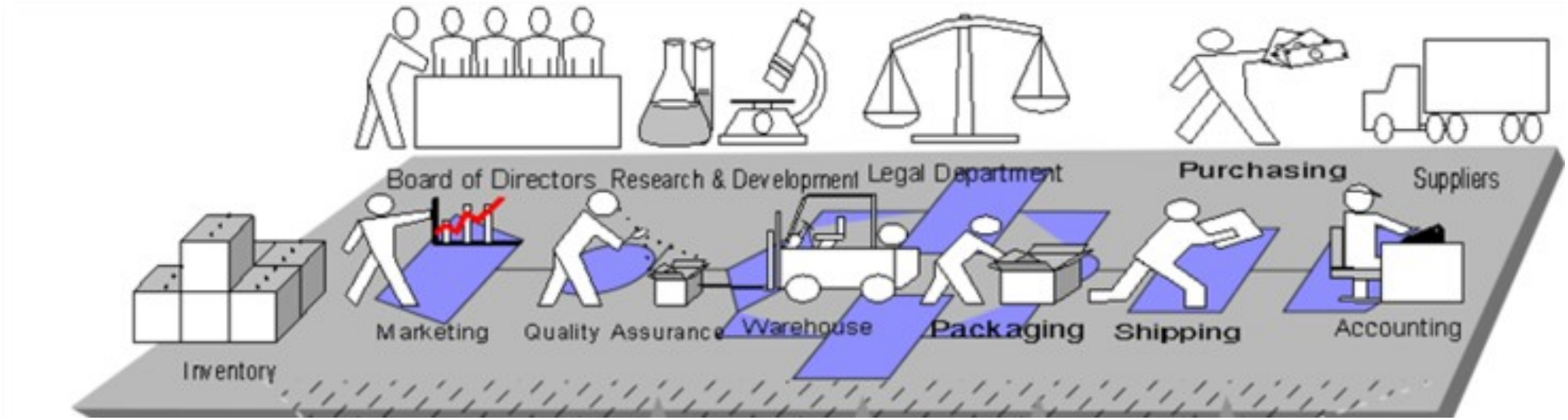
Karel Čemus

cemuskar@fel.cvut.cz

Předpoklady

- Technologie v SOA
  - REST, SOAP, RPC, CORBA, XML, JSON, WSDL, ...
- Co je to služba
  - Definice, vlastnosti, ...
- Orchestrace a choreografie
  - Integrace služeb a realizace procesů

Jak funguje business?



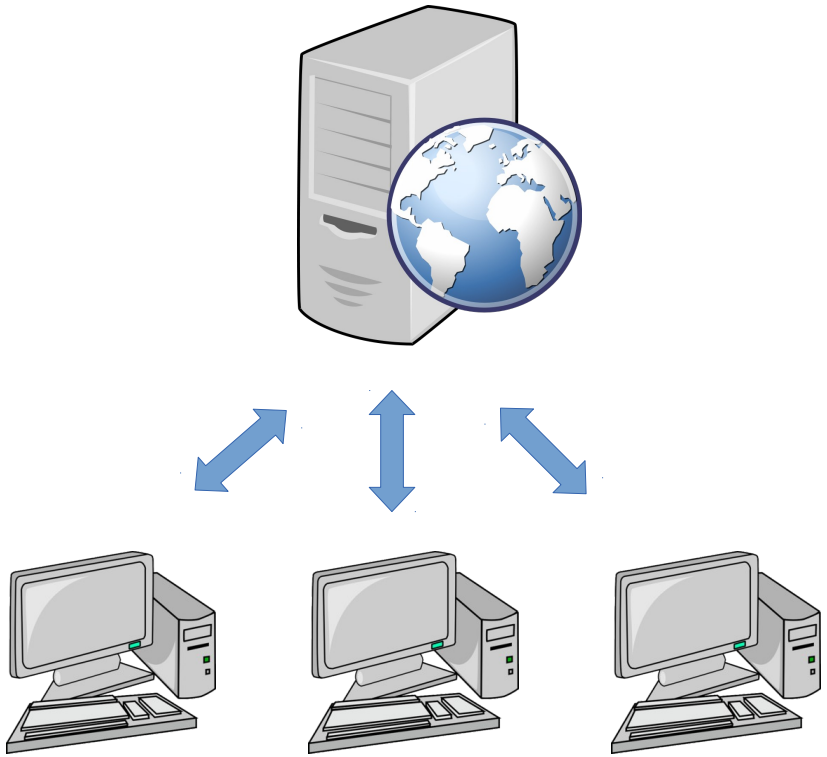
Procesy v businessu

- Business se nezajímá o služby ani technologie
- Definuje svoje vlastní procesy
  - Interní procesy toku dat (formulářů) mezi kancelářemi a odděleními
- Potřebuje naplnit svoje vlastní potřeby
  - Nechce se přizpůsobovat technologiím, je to pro něj neefektivní

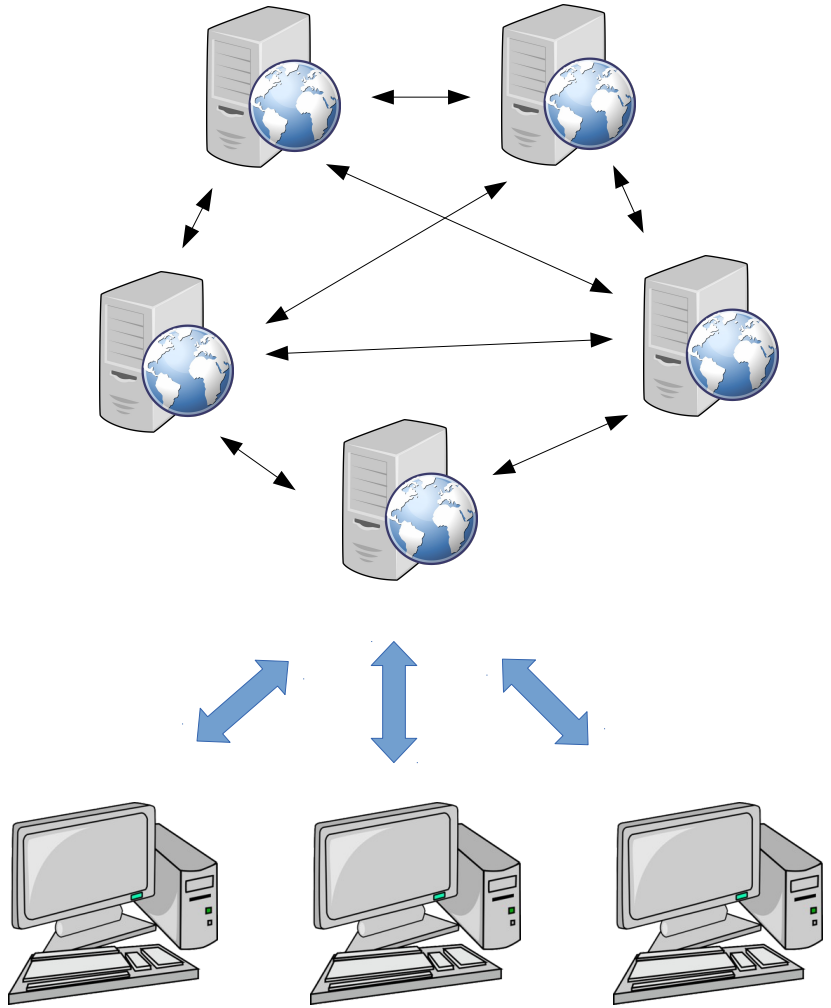
Současné trendy

- Trendy v oblasti SW
  - OOP
  - Komponentový přístup
  - Znovupoužitelnost
- Trendy v oblasti sítí
  - Rychlejší a propustnější sítě
  - Otevřené systémy
  - Webové služby
  - Distribuované aplikace

Architektura systémů

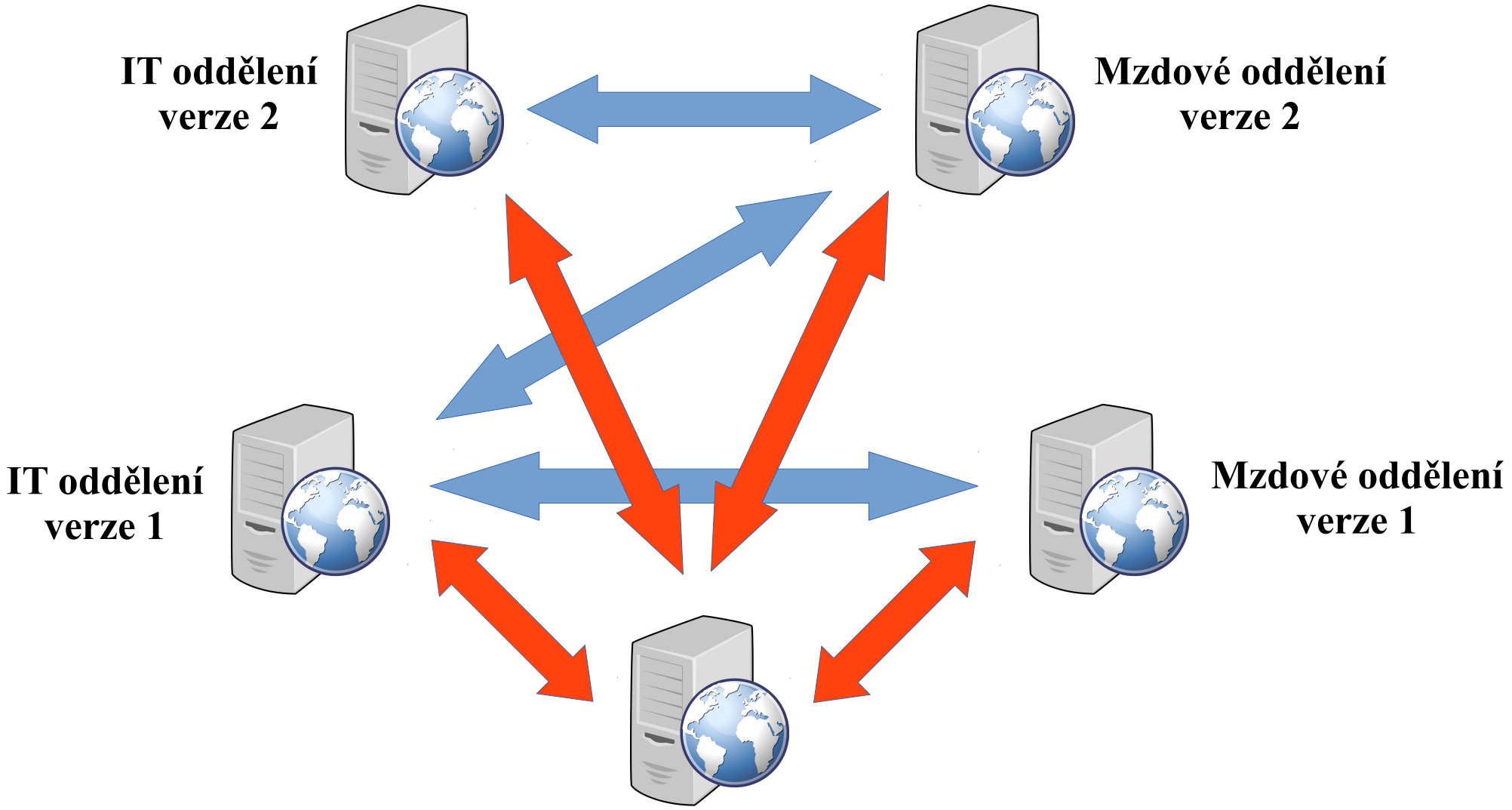


Klient-server



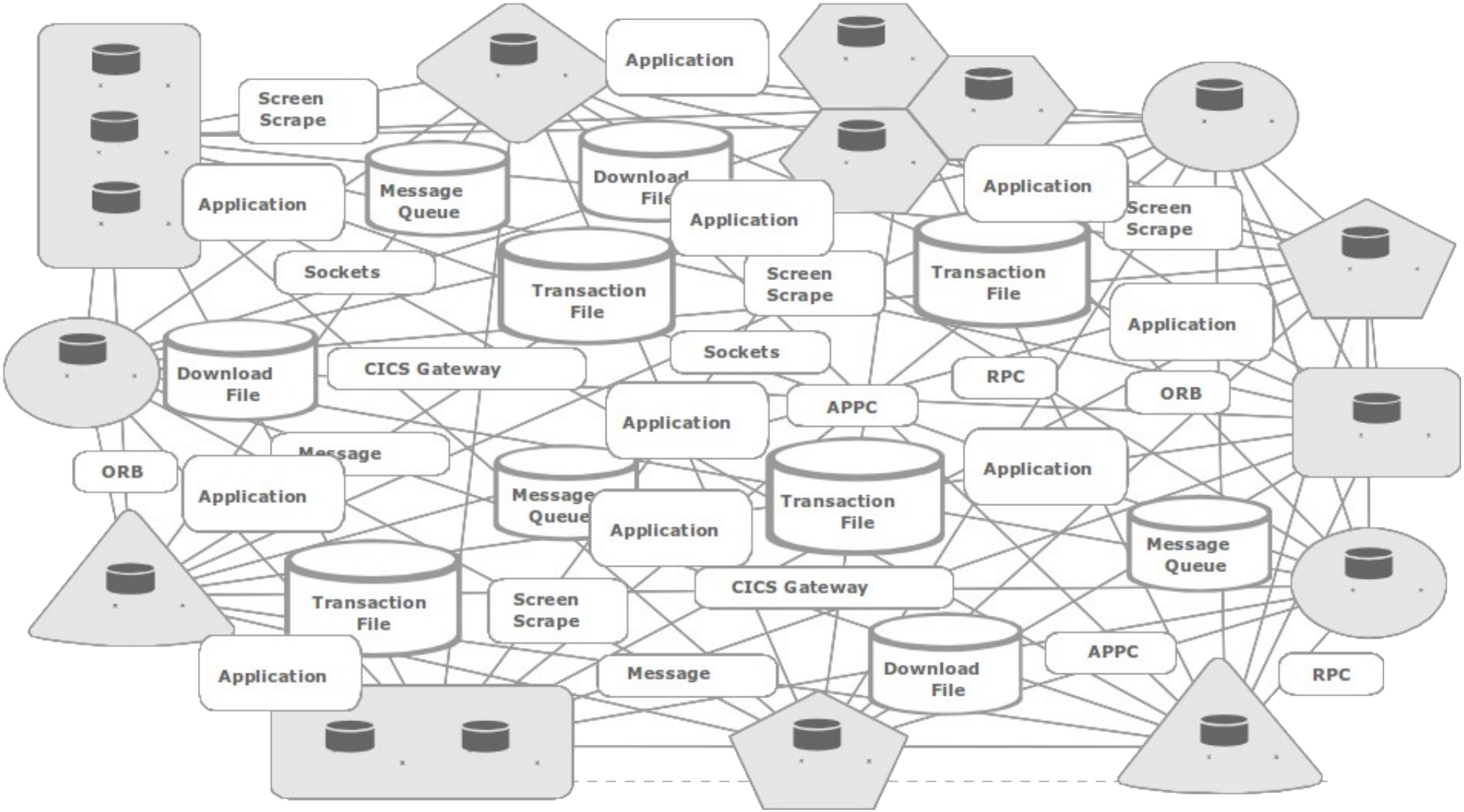
Distribuované aplikace

Vývoj infrastruktury

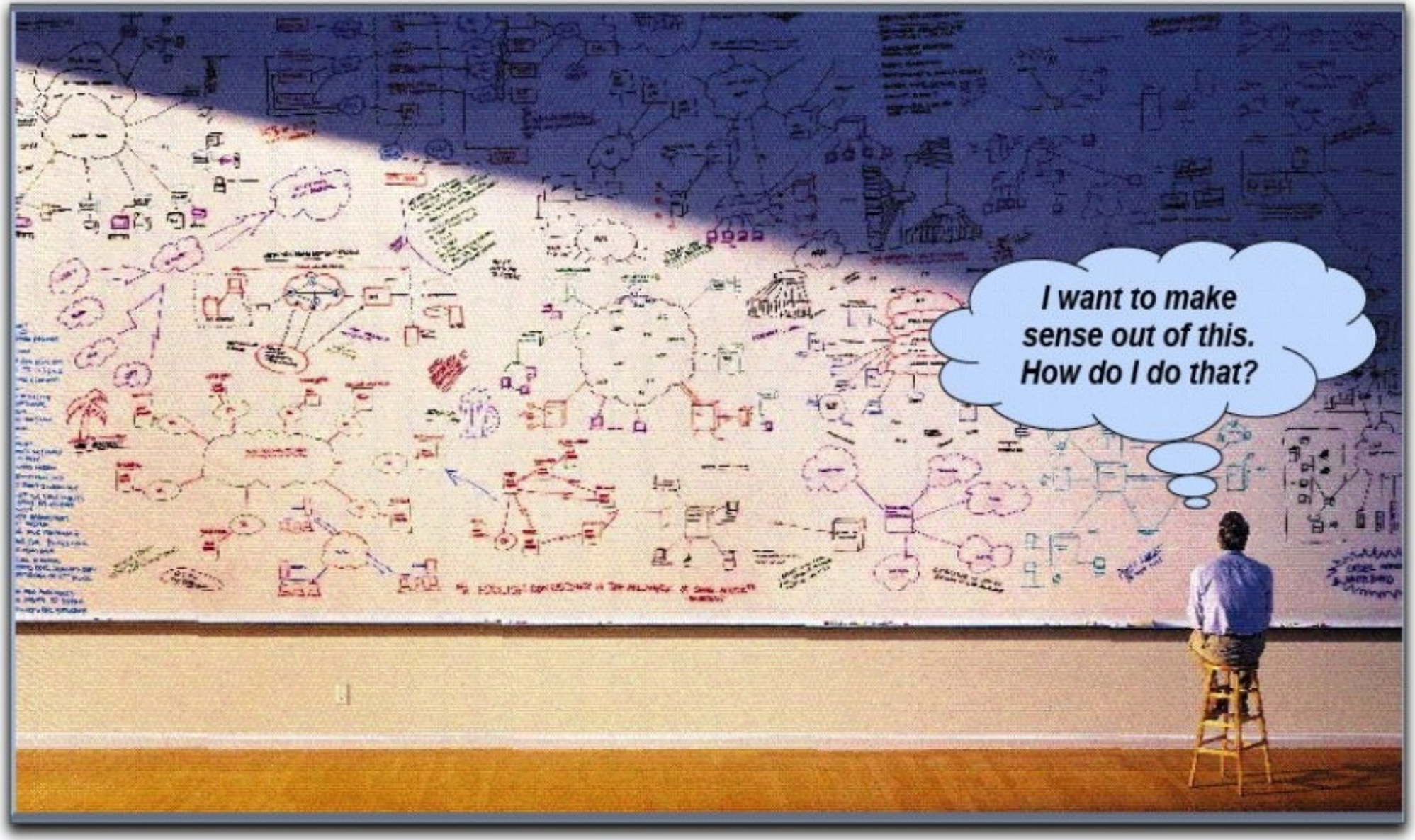


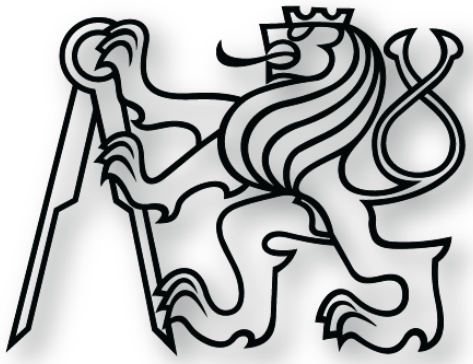
Service Bus Enterprise

Výsledek?!



Jak z toho ven?





Service Oriented Architecture

Co je to SOA?

- Service Oriented Architecture
- Metoda návrhu, nasazení a řízení aplikací a SW infrastruktury, kde:
  - Veškerý software je organizován do služeb
  - Je přístupný po síti a je vykonatelný
  - Rozhraní služeb je postaveno na standardech umožňujících jejich propojení

Charakteristika SOA

- Služby jsou platformě nezávislé
- Dopředu známá specifikace pro
  - Rozhraní služby, zpravidla ve formátu XML
  - Quality of service, security and performance
- Zprávy jsou formálně definovány
  - Je definovaná struktura, např. XSD
- Infrastruktura je zodpovědná za správu a řízení
  - Existuje prvek, který vše řídí a zodpovídá za to
- Protokoly jsou postavené na průmyslových standardech

Služba v SOA

- Základní stavební jednotka SOA
- Je recyklovatelná
- Mění byznys data z jednoho stavu do jiného
- Má přístup k datům
  - Jen ona, není jiné cesty
- Lze-li komponentu popsat ve WSDL, tak je to služba

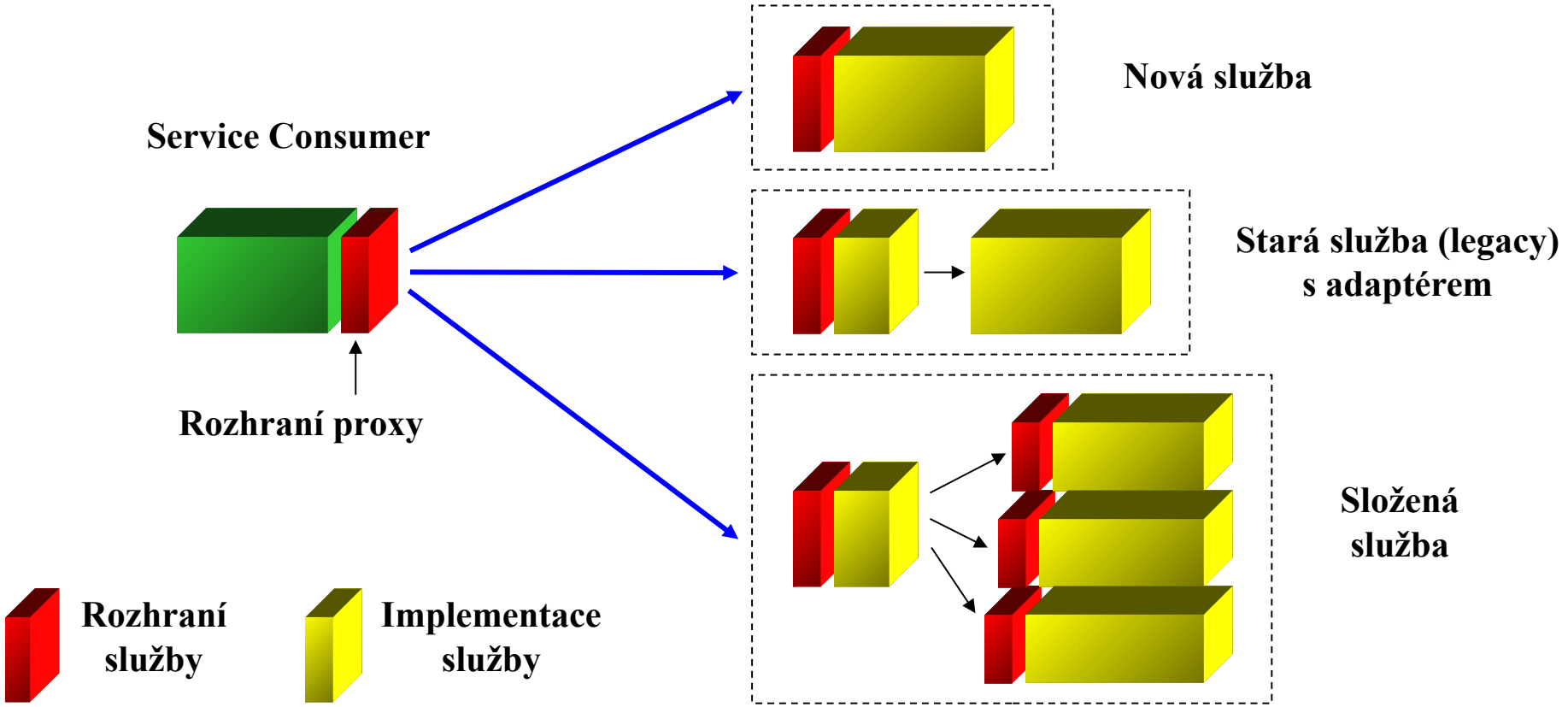
Principy návrhu služeb

- Abstraction
- Autonomy
- Composability
- Discoverability
- Formal contract
- Loose coupling
- Reusability
- Statelessness

Výhody SOA

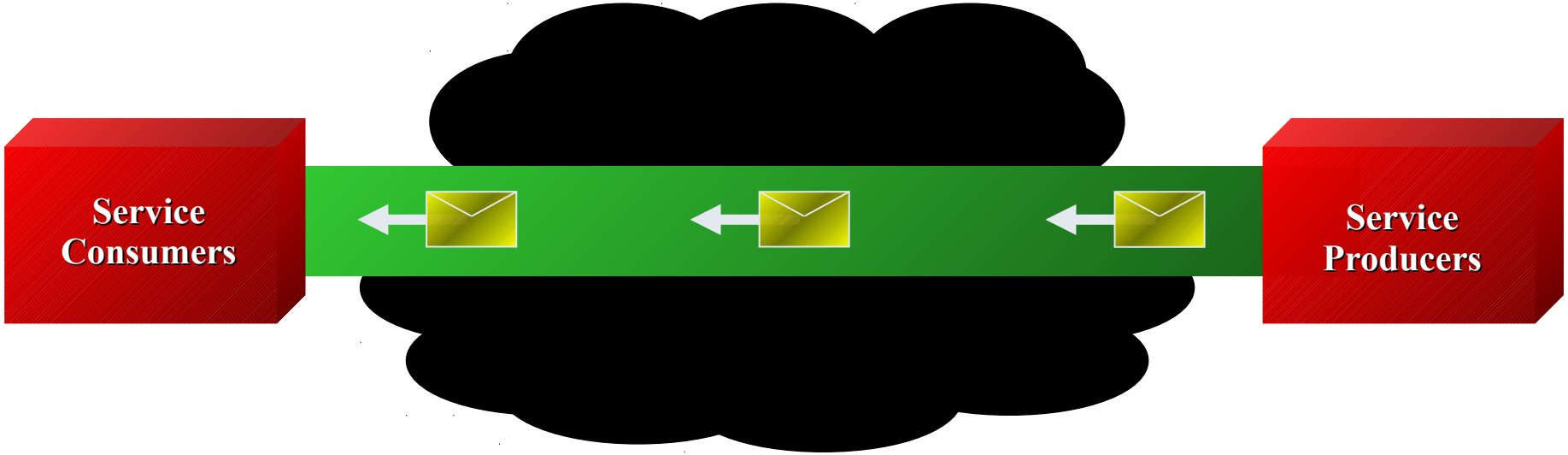
- Znovupoužitelnost komponent
  - Existující služby se recyklují, skládají se do lepších a lepších produktů
- Jasně definovaná rozhraní
  - Snadná úprava služeb, pokud dodržují předepsaný kontrakt (rozhraní i chování)
- Snadná údržba a nasazení nové verze
- Umí efektivně reagovat na měnínící se business pravidla
  - Adresuje nové potřeby v existujících aplikacích
- Podporuje přirozený vývoj
  - Obvykle reflektuje strukturu organizace

Anatomie služby



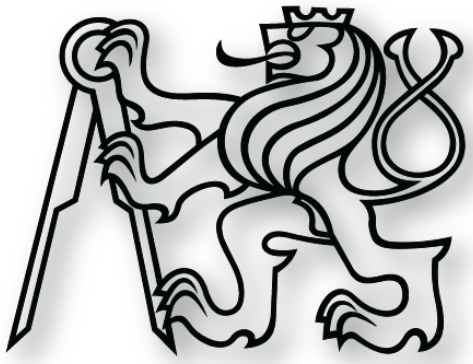
Komunikace v SOA

- Komunikuje se posíláním domluvených zpráv
- Heterogenní prostředí
- Neznalost druhé strany



Kontrakt služby

- Popis rozhraní služby
  - Umístění, metody, parametry, návratové hodnoty, ...
- Popis očekávaného chování
- Quality of Service
  - Dostupnost, spolehlivost, …
- Zabezpečení
  - Šifrování, podepisování, autentizace, …
- Typicky se používá WSDL
  - Web Service Description Language

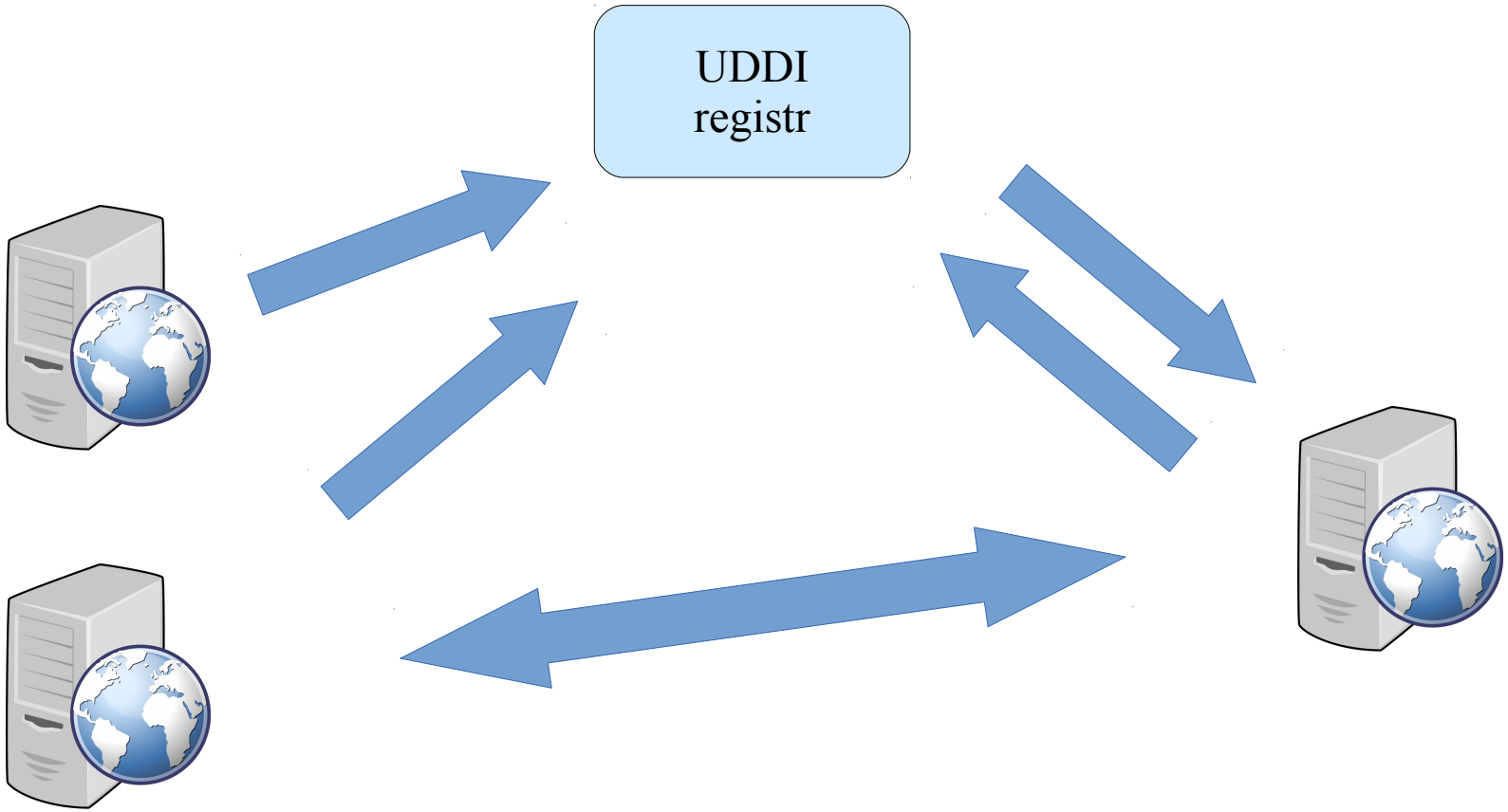


Realizace

Choreografie

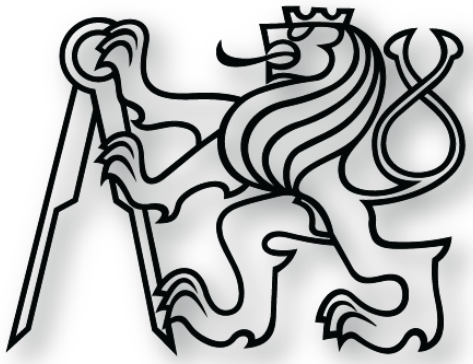
- Popis spolupráce služeb
- Standardizováno W3C
- Existuje popis toho, jak mají služby spolupracovat
  - Obvykle v XML
- Každá služba zná svůj part a podle toho se chová
- Neexistuje řídící prvek
- Lze přirovnat k tanečnímu souboru

Katalogizace služeb



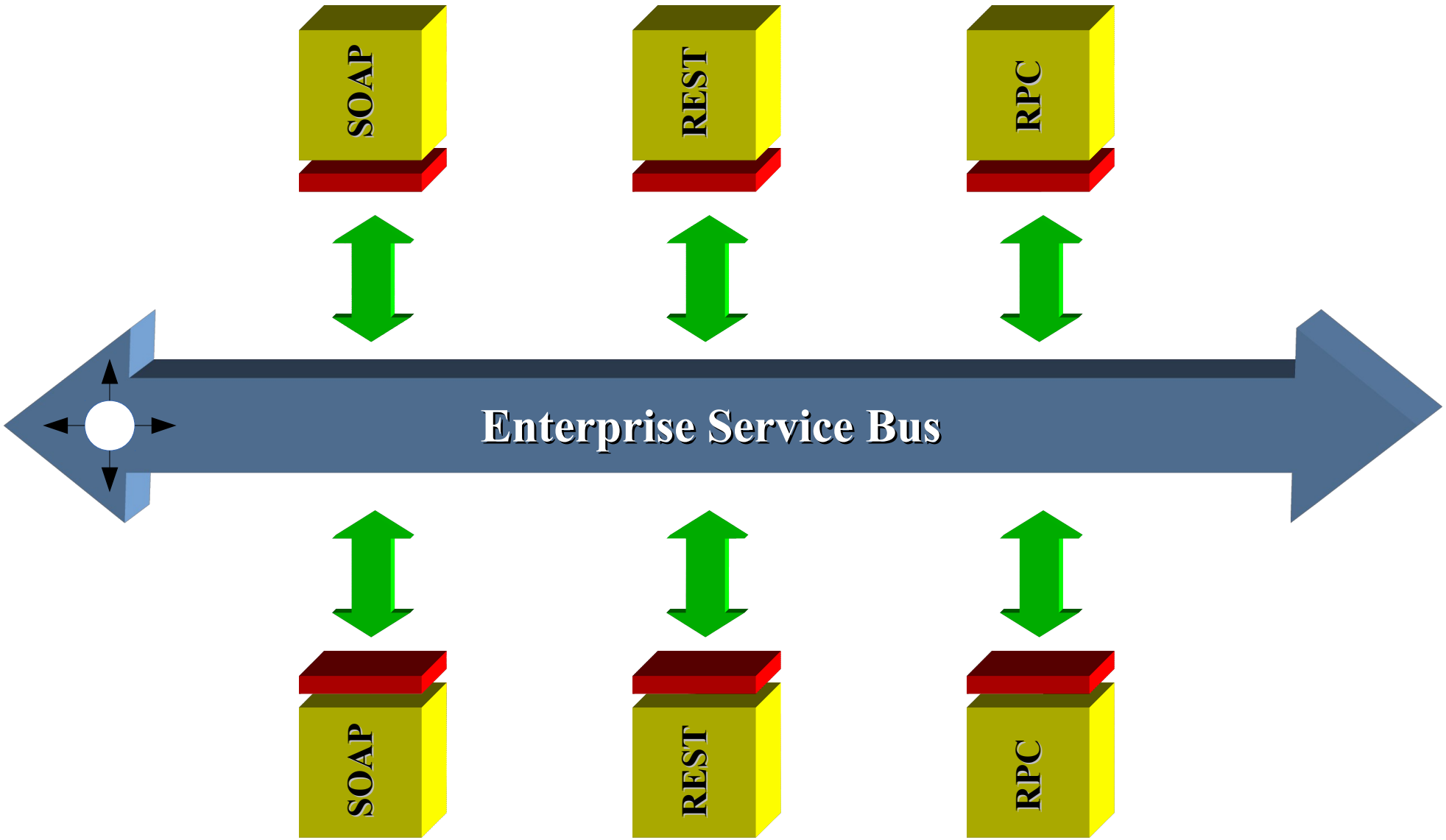
Orchestrace

- Popis spolupráce služeb
- Standardizováno W3C
- Existuje popis toho, jak mají služby spolupracovat
  - Např. BPEL
- Existuje řídící prvek, který zná proces
  - Inicializuje instance, řídí komunikaci
- Přináší automatizaci nasazování a ovládání služeb
- Lze přirovnat k hudebnímu souboru s dirigentem

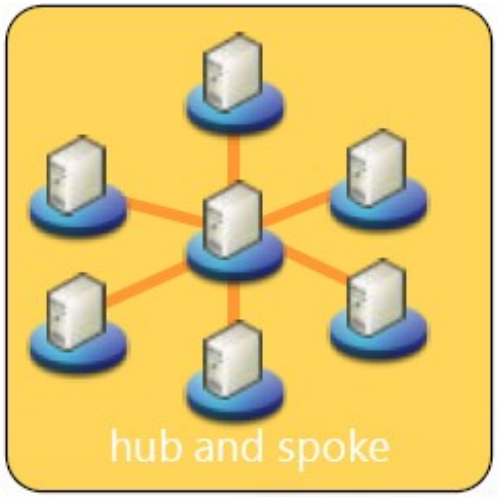


Enterprise Service Bus

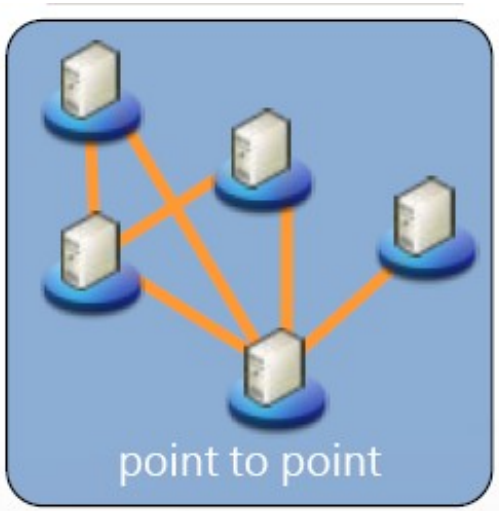
Architektura



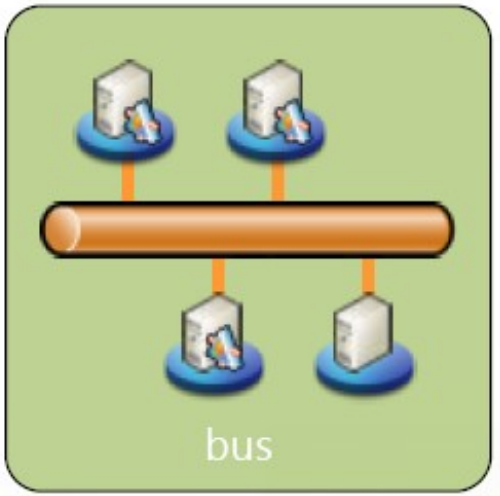
Pokrok jménem ESB



"too centralized



"too decentralized"



"just right"

Co to je?

- Architektonický model pro propojení služeb v SOA architektuře
- Zaměřuje se na systémy s distribuovanými výpočty
- Propojení heterogenních prostředí, různých technologií a protokolů
- Varianta na klient-server architekturu
- Využívá messaging
  - Komunikace prostřednictvím zpráv

Messaging

- Forma komunikace prostřednictvím zpráv
- Využívá se u distribuovaných výpočtů
- Kontext výpočtu se přenáší jako zpráva mezi komponentami
- Zavádí nepřímou komunikaci mezi komponentami
- Minimalizuje provázání mezi službami (low coupling)
  - Využívá prostředníka
- Dovoluje propojení heterogenních prostředí a platforem
- Zvyšuje škálovatelnost

Vlastnosti

- Nezávislé na platformě služeb
- Řízené událostmi
- Podporuje orchestraci
- Splňuje ACID
- Přináší podpůrné služby
  - Logování, zabezpečení, správa chyb, sledování Quality of Service ...

Výhody návrhu

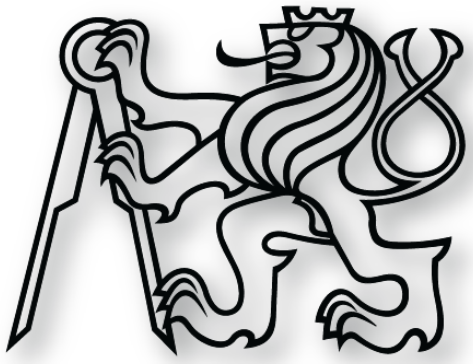
- Větší flexibilita a reakce na změnu
  - Požadavků, rozhraní služby, technologie služby, procesu, ...
- Monitoruje a ovládá jednotlivé služby
  - Podle zátěže snižuje/zvyšuje počet instancí dané služby
  - Spravuje nasazení a verzování služeb
- Specifikace procesu na jednom místě
- Převládá konfigurace nad implementací
  - Propojení služeb popisujeme, neprogramujeme
- Výhoda aktualizace systému bez nutnosti jeho odstavení
  - Nulový „down-time“

Nevýhody návrhu

- Zvýšený overhead
  - Adaptéry služeb
  - Mapování
  - Nepřímá komunikace
- Pomalejší zpracování dat
  - Serializace a parsování zpráv
  - Obsluha podpůrných služeb

Service management

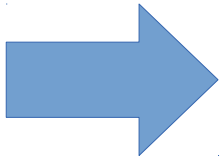
- Škálování: automatické nasazení/vypnutí instance
- Logování
  - Kdo s kým komunikaval, co posílal, ...
- Reportování problémů
- Monitorování provozu a stavu služeb
- Bezpečnost komunikace a autorizace subjektů
- Zaručuje spolehlivost komunikace



Java Message Service

Dostupnost ESB

- Kvalitní implementace nejsou zdarma
- Obvykle se dodávají celá hotová řešení
- Komplexnost ESB často není potřeba



stačí light verze

Co to je?

- Implementace messagingu na platformě Java
- Umožňuje popojení několika klientů ve stejném JVM
- Součástí standardu a Java Enterprise Edition
  - Publikováno jako JSR-914

Proč to používat?

- Vhodný návrh distribuovaných výpočtů s využitím JMS dovoluje:
  - Snížit provázání komponent (low coupling)
  - Zvýšit spolehlivost a dostupnost (více instancí stejné služby)
  - Zavést asynchronní zpracování
- Je to povinná součást Java EE aplikačního serveru
  - Od verze Java EE 1.4

Prvky JMS

- Provider
  - Implementace JMS
- Client
  - producent nebo spotřebitel zpráv
- JMS Message
  - Zpráva, objekt obsahující data, například kontext distribuovaného výpočtu nebo parametry pro vykonání nějaké akce
  - Zprávy nemusí být doručeny v pořadí svého odeslání, ale neztratí se
  - JMS garantuje, že každou zprávu zpracuje právě jednou

Prvky JMS

- Consumer/Subscriber
  - spotřebitele, příjemce zpráv
- Producer/Publisher
  - producent zpráv
- JMS queue
  - Fronta přijatých zpráv čekajících k vyzvednutí spotřebiteli
- JMS topic
  - Mechanismus pro posílání zpráv více příjemcům

Message driven beans

- Business objekty volané událostmi
- Odebírají JMS zprávy
- Obsluhují zprávy přijaté JMS
- Představují službu z hlediska SOA
  - Mohou i produkovat další zprávy
- Může být více instancí v případě dlouhých výpočtů
- Metody jsou volány asynchronně dle potřeby
- Long polling

Point to point

- Každou zprávu přijme pouze 1 příjemce
- Instancí příjemce může být více, ale doručena je pouze jednomu
- JMS realizuje doručování pomocí fronty (queue)
- Pokud není registrován žádný příjemce, zprávy čekají na doručení
- Příklad: Žádost vytvoření IT účtů pro nového zaměstnance

Publish and subscribe

- Zpráva je doručena všem příjemcům daného tématu
- Pokud žádní nejsou registrovaní, zpráva se ztratí, nečeká na doručení
- Existuje i trvalé odebírání tématu
  - Zpráva se archivuje po určitou dobu
  - Doručení je možné i v případě, že příjemce nebyl v okamžiku publikace aktivní
- JMS realizuje doručování pomocí tématu (topic)
- Příklad: Publikování příspěvku na fórum

Navazující témata

- Modelování SOA
- Modelování orchestrací (BPEL, BPMN)