KIV/SI

8. Integrace na aplikační vrstvě, SOA

Význam systémové integrace

- Integrace aplikací (systémů) do jednoho celku.
- Celý proces nezbytný pro efektivní fungování IS.

Důvody integrace

- Zjednodušení stávající architektury integrace.
- Nižší náklady na modifikaci stávajících systémů a aplikací.
- Nižší náklady na implementaci a integraci nových systémů a aplikací.
- Větší automatizaci business procesů, což přináší nižší náklady a větší rychlost zpracování.
- Možnost snadnější integrace se systémy externích subjektů.

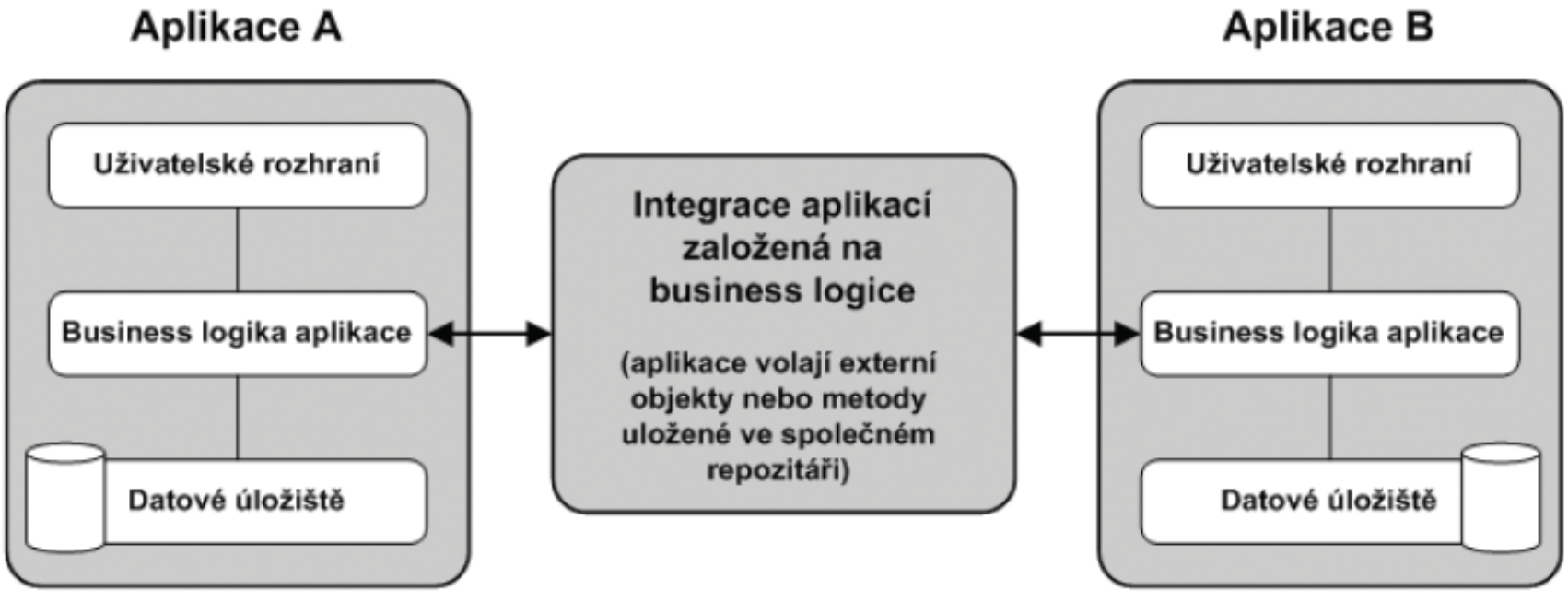
Typy integrace

- Integrace na úrovni prezentační vrstvy (uživatelského rozhraní).
- **Integrace na úrovni aplikační vrstvy.**
- Integrace na úrovni datové vrstvy.
- Integrace mezi rozdílnými vrstvami architektury IS.

Integrační styly

- Vzdálené volání
  - RPC, RMI, REST, webservice, atd.
- Zasíláním zpráv
  - ESB, sběrnice, brokery

Integrace na aplikační vrstvě



Úskalí integrace

- Zásah do aplikace.
- Vzájemná závislost a těsná vazba aplikací.
- Syndrom N^2.
- Heterogenita.

SOA

aneb architektura orientovaná na služby

SOA

- Skládání složitých systémů ze skupiny na sobě nezávislých komponent poskytujících služby.
- Systém si můžeme představit jako stavebnici LEGO složenou ze služeb (komponent).
- **Služba** - dílčí úkol, který je nutno splnit při obsluze uživatele.

SOA - Služby

- Služby lze různě kombinovat, doplňovat, případně snadno nahrazovat.
- Systém je stabilnější a také je zde rovnoměrné rozložení zátěže mezi služby.
- Při výpadku jedné služby může být nahrazena jinou, funkční.
- Systém se lépe spravuje a vylepšuje.

SOA - Webové služby

- Jednoduchá komunikace mezi aplikacemi v heterogenním prostředí.
- Komunikace založená na platformě nezávislých standardech.
- Komunikace přes SOAP (postavený na XML).
- Komunikace nejčastěji přes HTTP.
- Komunikační rozhraní popsáno WSDL.

SOA - Principy I.

- Standardizovaný kontrakt služby
- Slabé vazby mezi komponentami
- Princip abstrakce
- Znovupoužití
- Nezávislost

SOA - Principy II.

- Bezstavovost
- Princip identifikovatelnosti
- Princip skládání
- Princip orientace na služby a použití v různých podmínkách

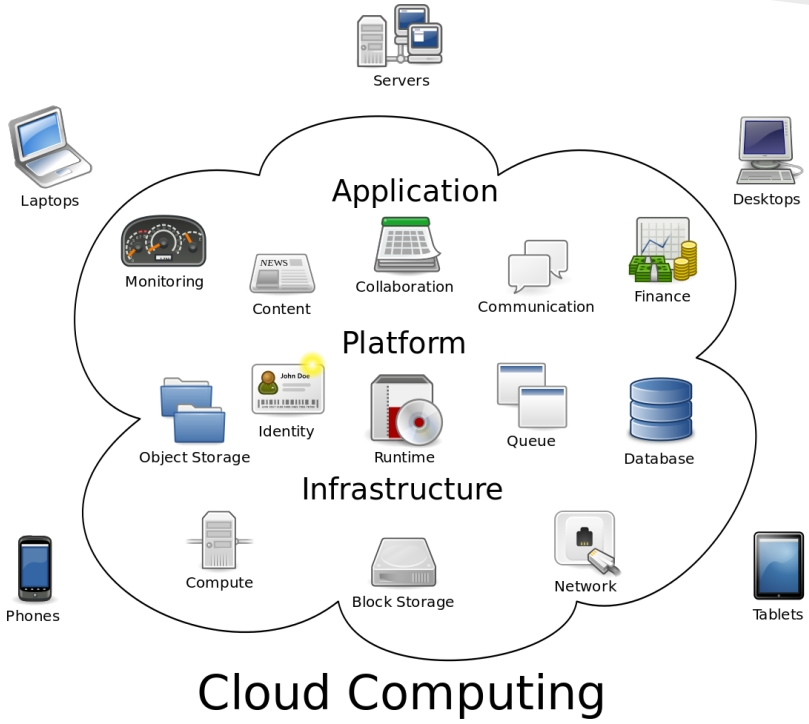
Možnosti realizace

aneb konečně praktická implementace….

Realizace

- Historie - ústně, hliněné destičky, papíry, telefony, sítě
- Standart rozhraní
- Cloud (buzz….) computing (hype)
- $$ ne za přístup, ale za využití (měsíčně / ročně)
- $$ za cokoliv - místo / výpočetní výkon

Cloud computing

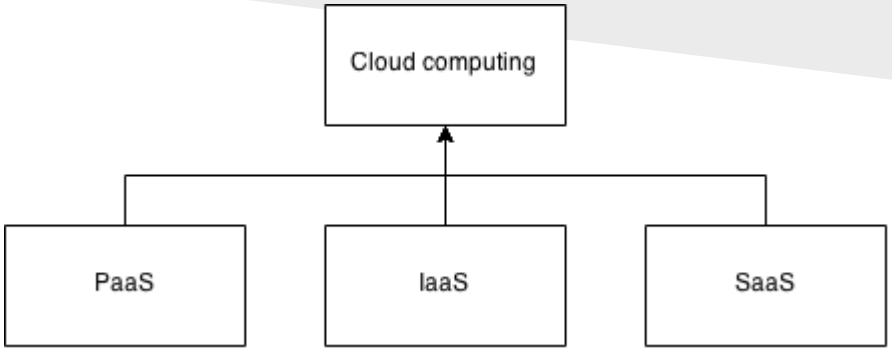


Dělení cloud computingových služeb

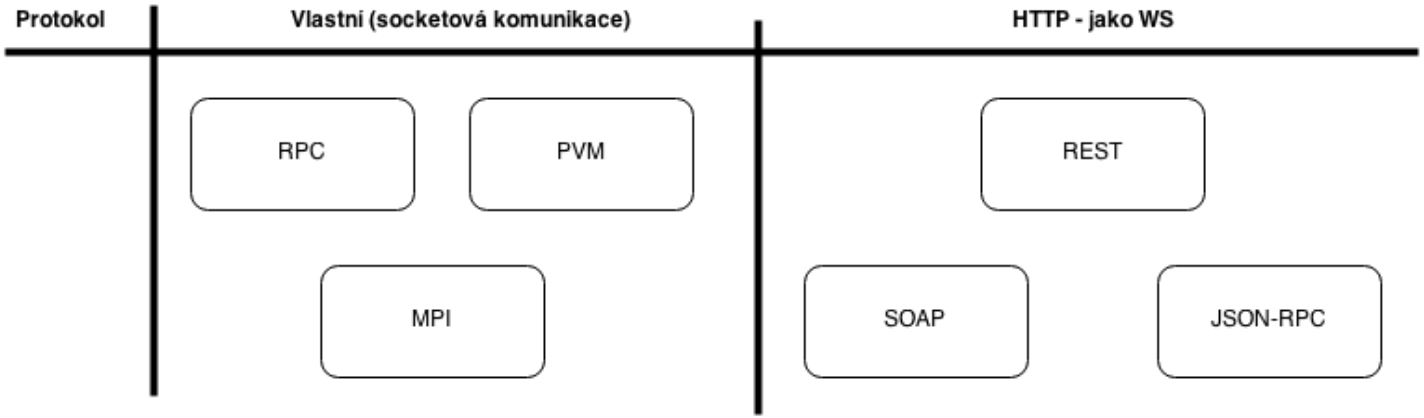
- Model nasazení
  - Veřejné (cokoliv Dropbox, Amazon WS)
  - Privátní (vlastní firemní, pouze pro zaměstnance / klienty)
  - Komunitní (přístup komunity)
- nebo zajímavější - ***distribuční model***

Distribuční modely cc

- **SaaS**
  - Pronájem aplikací
  - Adobe Creative Cloud, Google Apps
- **IaaS**
  - Pronájem infrastruktury (hardwaru), psychologický problém
  - Amazon WS, Windows Azure
- **PaaS**
  - **Nejhorší na vysvětlení**
  - Tvorba webové aplikace bez starosti o běhovou platformu
  - Google App Engine
  - Python, Java, JRuby atp..



Vlastní integrace a mechanismy



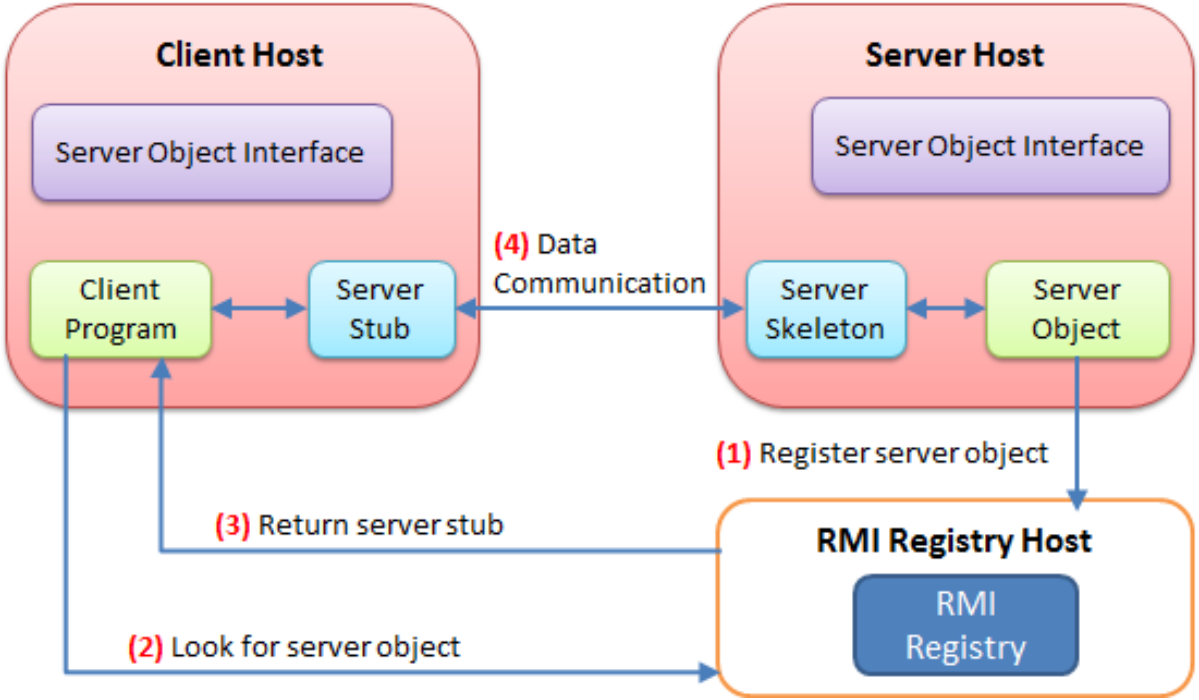
RPC

- Volání výkonného kódu umístěného na jiném stroji..
- Data potřebná pro volání procedury se zabalí (marshalling) odešlou přes síť (zpět stejně)
- Zásadním problémem je to, že server musí být neustále připojen a jakákoliv chyba je nezotavitelná.
- Lze lépe řešit přes WS

RMI I.

- Technologie jazyka Java.
- Založeno na architektuře Klient – Server.
- Umožňuje z jednoho JVM volat metody objektů na jiném virtuálním stroji.
- Komunikace se vzdálenými aplikacemi na úrovni lokálního volání metod.

RMI II.


Client Host
Server Object Interface
Client Program
Server Stub
(4) Data Communication
Server Host
Server Object Interface
Server Skeleton
Server Object
(1) Register server object
(3) Return server stub
RMI Registry Host
RMI Registry
(2) Look for server object

SOAP

- Vytvořeno za podpory Microsoftu (zaměstnanec), ale používají to všude možně
- Náhrada RPC-XML (nic nového)
- Přesně definován formát zprávy
  - Envelope, Header, Body
- Formát XML dříve problém, dnes už ne?!

JSON-RPCv2

- To samé jako SOAP, ale menší režie (JSON)
- Definovaná struktura obsahu jsonu
  - method, id, jsonrpc, params
- Využití hlavně v mobilních API

```
--> {"jsonrpc": "2.0", "method": "subtract", "params": [42, 23], "id": 1}
 <-- {"jsonrpc": "2.0", "result": 19, "id": 1}
```

- Definována i chybová zpráva

REST

- Závislý na HTTP
- Datový ne procedurální!
- Data přístupná pod unikátními URI
- Manipulace předem definovanými metodami
- Volitelný formát výměnny dat
- Může se zdát, že je nadmnožinou SOAP/JSON-RPC, ale není ...

Ukázka REST

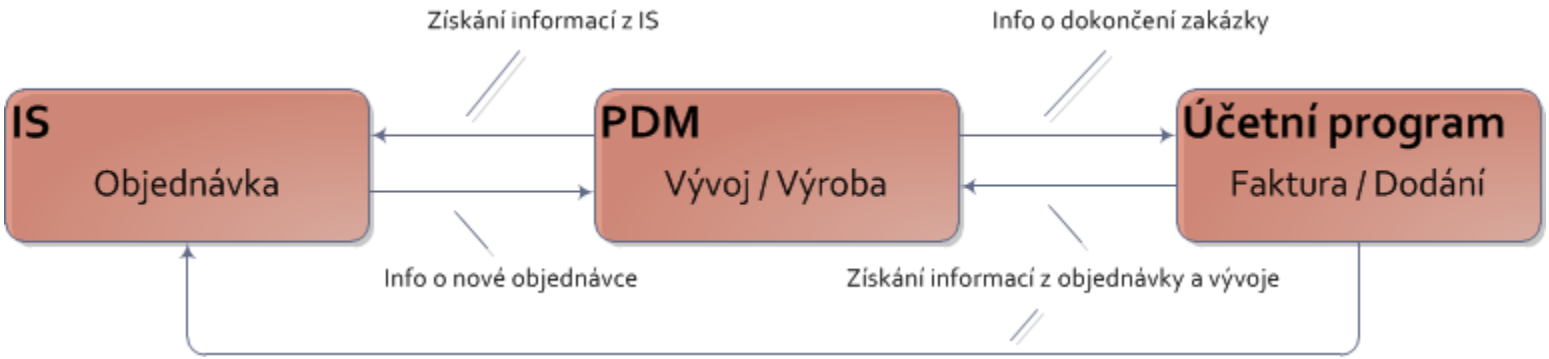
- GET **/collection/** nebo GET **/collection/:cd_id**
  - Získání celé kolekce nebo detailu CD
- POST **/collection/**
  - Vytvoření celé kolekce
- PUT **/collection/**
  - Nahrazení celé kolekce (“zneužíváno”)
- DELETE **/collection/**
  - Smazání celé kolekce

Příklad integrace

aneb jak to vypadá v praxi

Příklad integrace

- Proces před integrací systémů



Příklad integrace

- Proces po integraci systémů