# ARCHITEKTURY INFORMAČNÍCH SYSTÉMŮ

*Jaroslav Žáček*
*jaroslav.zacek@osu.cz*
*http://www1.osu.cz/~zacek/*

# NUTNÉ POJMY

- Co je to informační systém?
    - Jaké oblasti zahrnuje?
- Jaká je vazba IS na podnikovou strategii?
- Jaká je vazba na podnikové procesy?

# NUTNÉ POJMY

- **ERP** - Enterprise resource planing
- **CRM** - Customer Relationship Manager
- **SCM** - Supply Chain Management
- **B2B** - Business to Business
- **B2C** - Business to Customer

# ARCHITEKTURY

# MODEL ŽIVOTNÍHO CYKLU

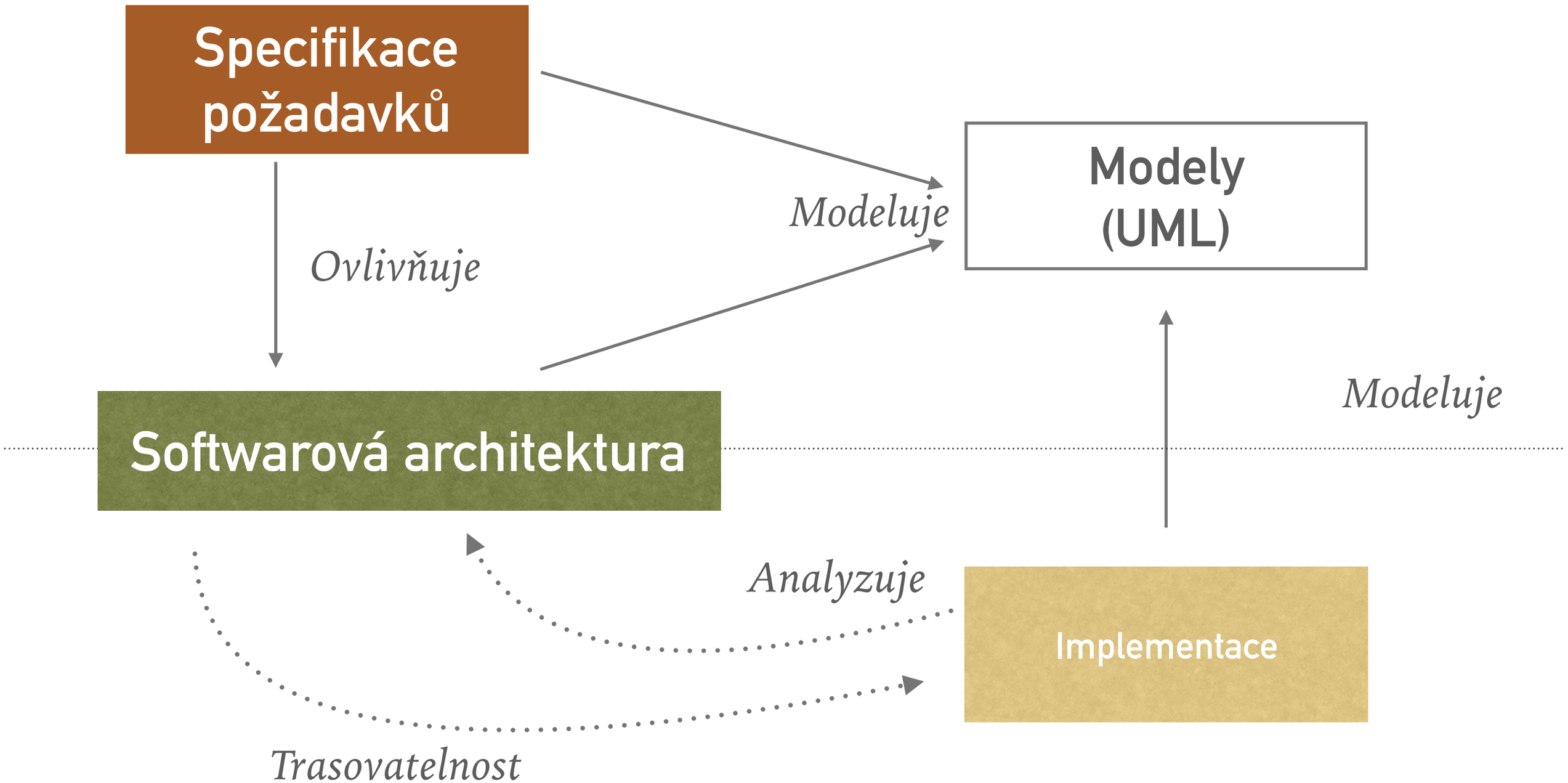
- Analýza a specifikace požadavků (8%)
- Architektonický a podrobný návrh (7%)
- Implementace (12%)
- Integrace a testování (6%)
- Provoz a údržba (67%)

# PROČ SE STARAT O ARCHITEKTURU?

# TROCHU HISTORIE

- 1992-1993 - Softwarová architektura jako samostatný směr výzkumu
- 1993-1995 - Popisována neformálními diagramy
- 1995-1997 - Vytvořen ADL - Architectural Description Language
- 1997-2003 - SA stanovena jako nástroj pro řešení pro popis komplexních, konkuretních systémů a systémů pro zpracování v reálném čase
- 2003 - ? - SOA, Cloud Design Patterns, Front-end vs- Back-end

# PROCES


Specifikace požadavků
Modeluje
Modely (UML)
Ovlivňuje
Softwarová architektura
Modeluje
Analyzuje
Implementace
Trasovatelnost

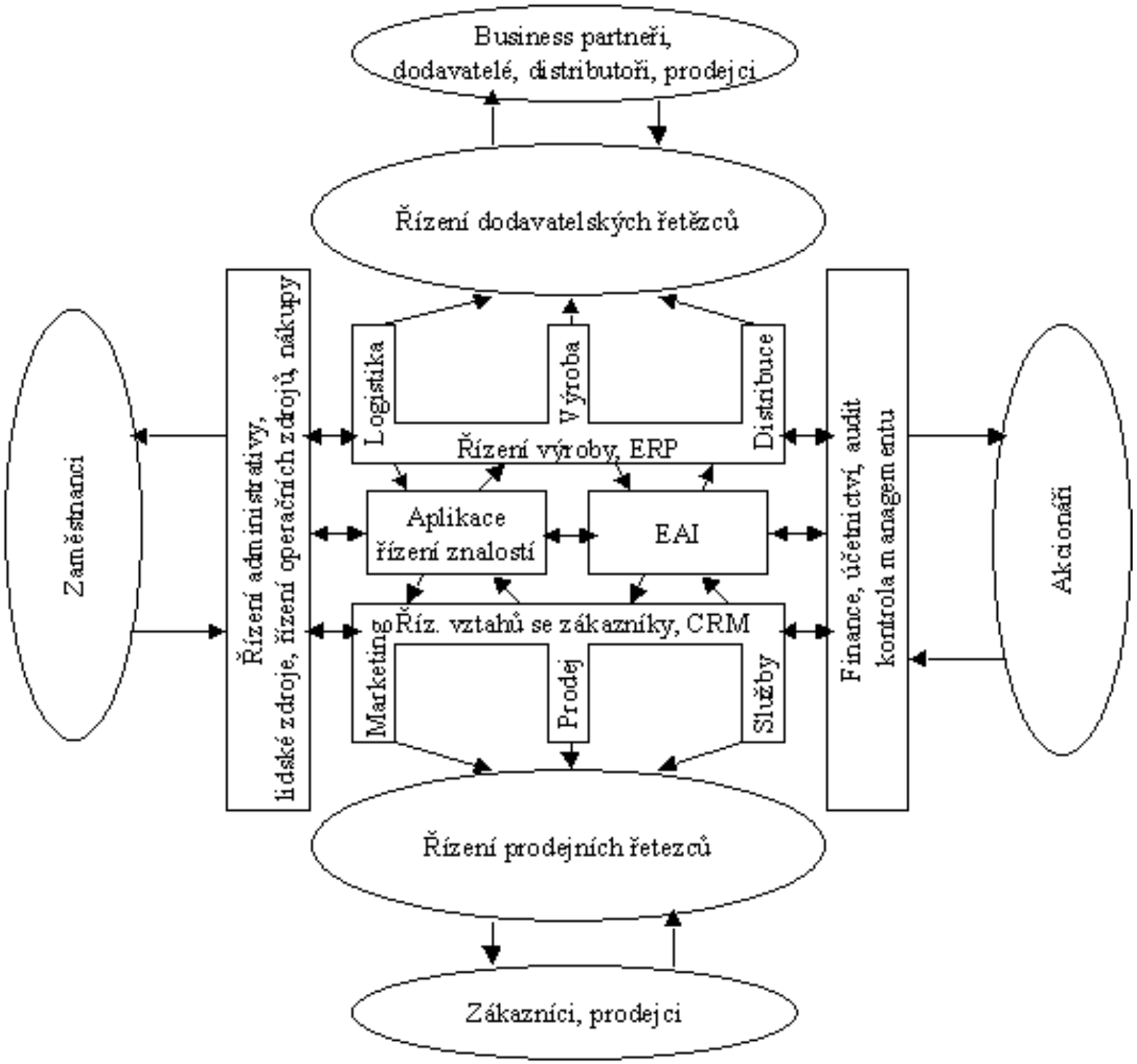
# ARCHITEKTURA IS

- Klíčový prvek řízení IS
- Respektuje strategii podniku, podnikové cíle
- Architektura IS vyjadřuje celkovou vizi
- Neexistence architektury způsobuje:
  - Nepokryté požadavky a funkce
  - Nejsou dostupné požadované nástroje (vše se dělá ručně)
  - Draze nakoupený HW a SW, který se nedá využít
  - Nekompatibilita formátů při výměně informací

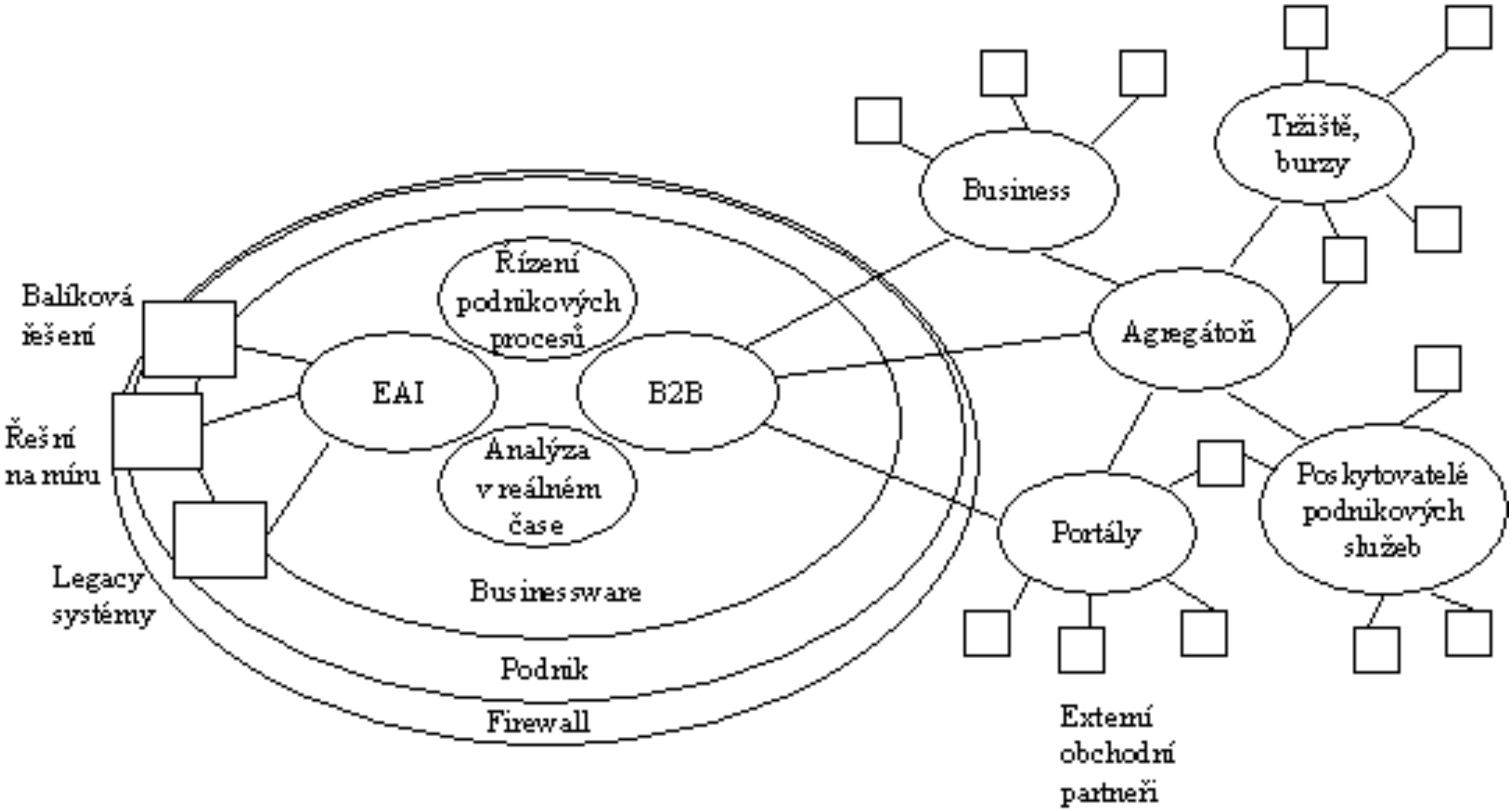
# POHLEDY NA ARCHITEKTURU

- Klasický pohled
  - Globální
  - Dílčí
    - řízení, služeb, aplikační, technologická
- Moderní pohled
  - architektura 4+1 (komponenty)
  - MDA
  - SOA

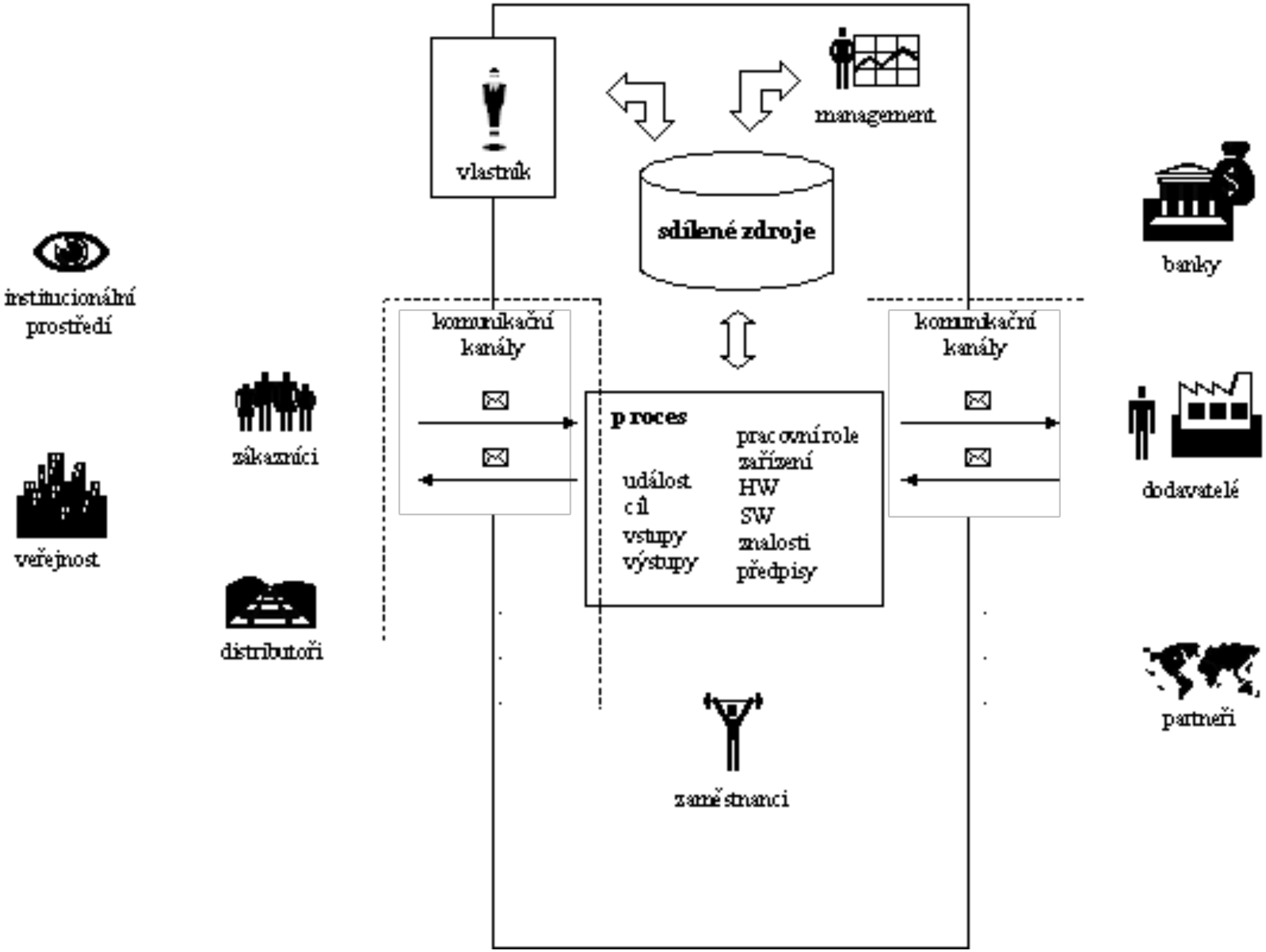
# GLOBÁLNÍ ARCHITEKTURA PODNIKU



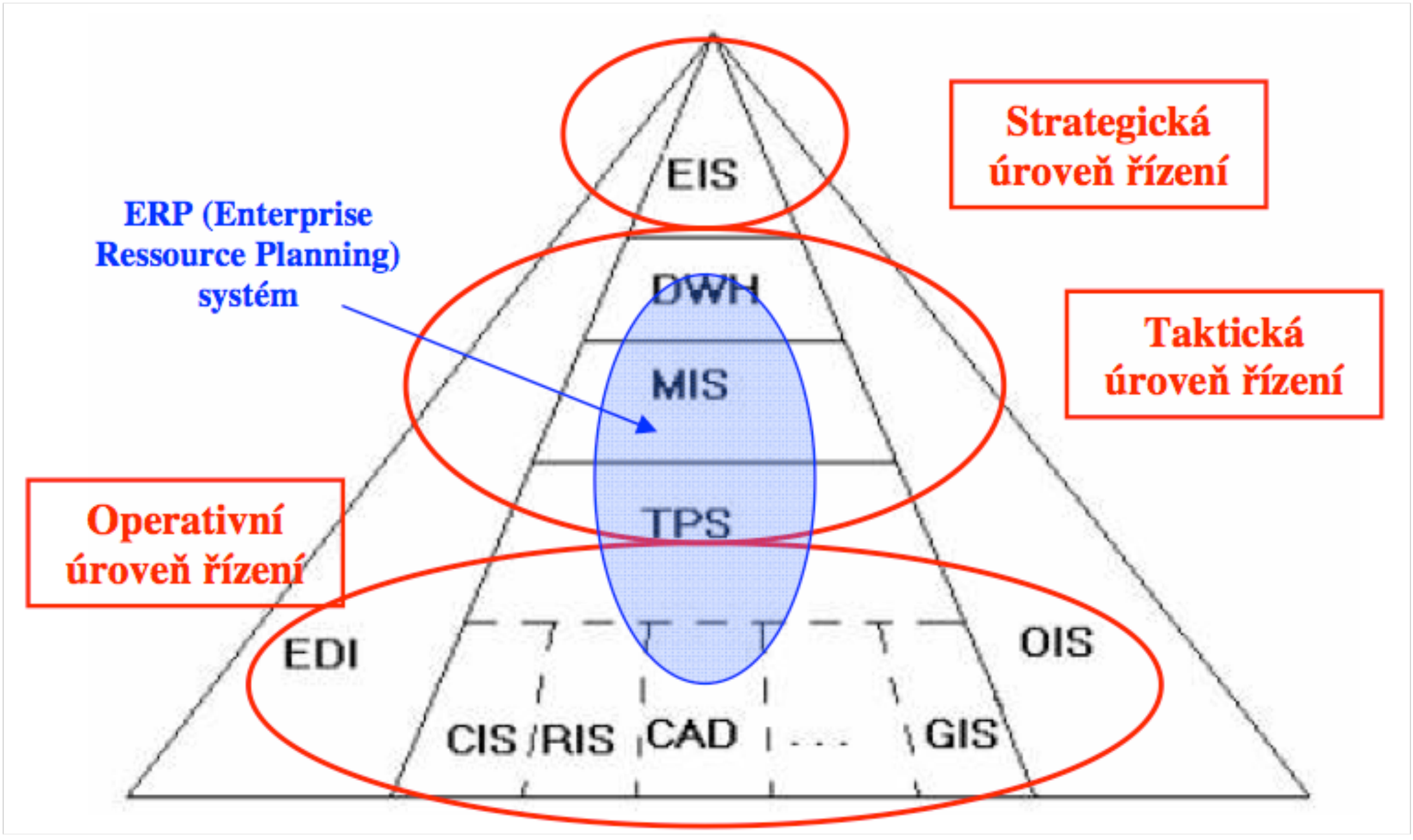
# JINÝ POHLED NA ARCHITEKTURU



# PROCESNÍ POHLED NA ARCHITEKTURU



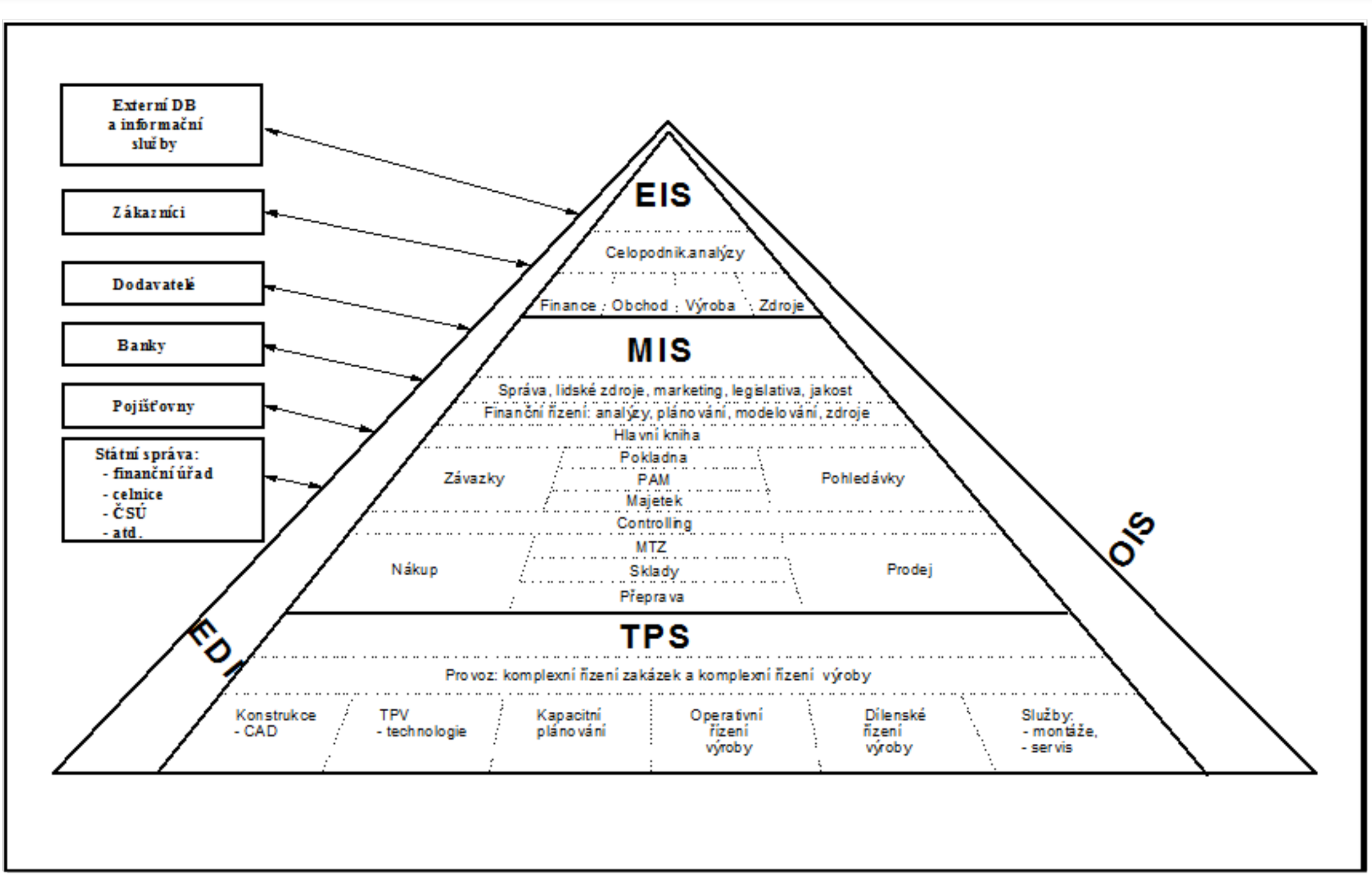
# OBECNÁ GLOBÁLNÍ ARCHITEKTURA



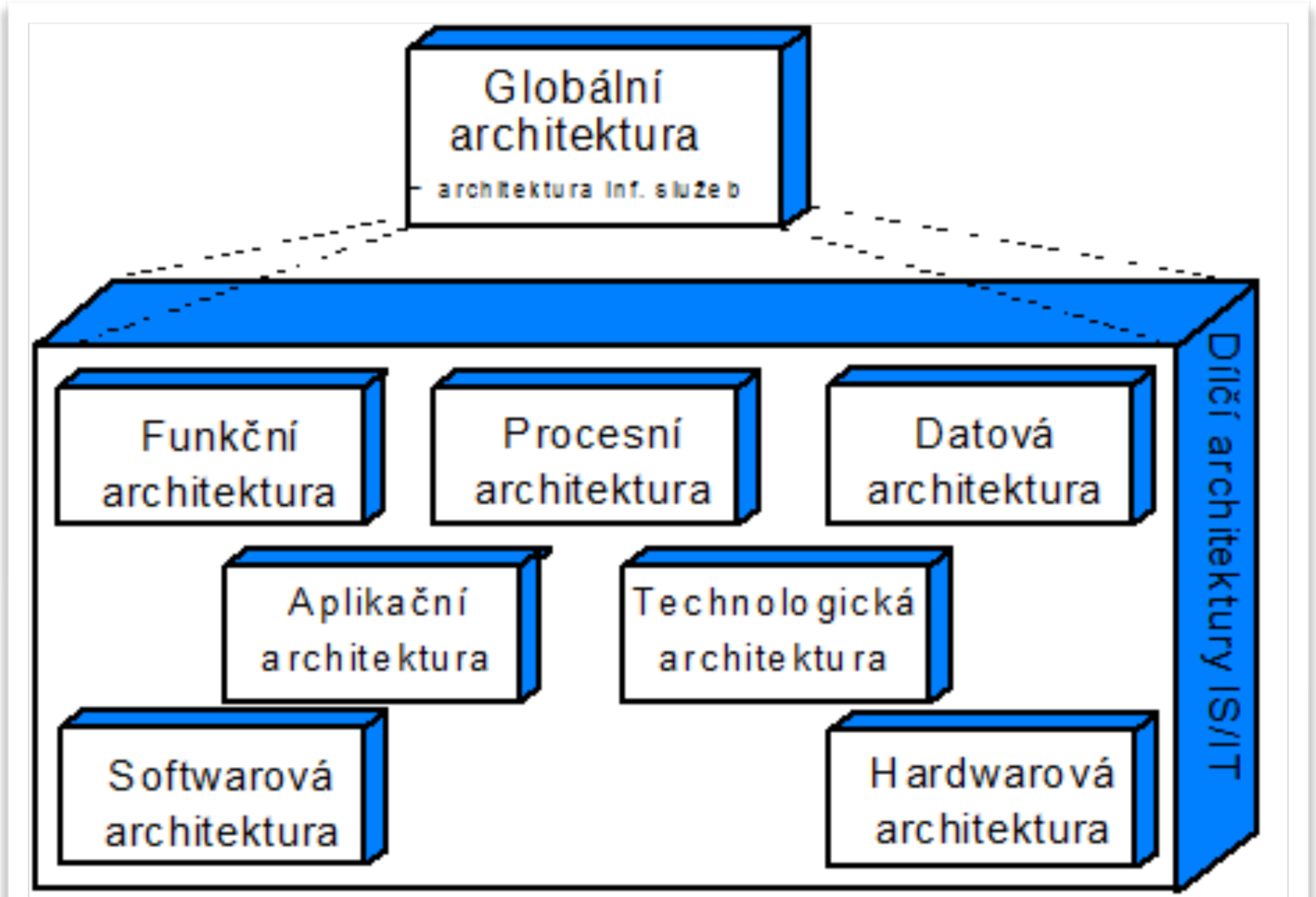
# OBECNÁ GLOBÁLNÍ ARCHITEKTURA

- **EIS** (Executive IS) - strategické řízení podniku
- **DWH** (Data Warehouse) - datový sklad, podpora řízení
- **MIS** (Management IS) - taktická úroveň, účetnictví, personalistika, marketing, legislativa, logistika
- **TPS** (Transaction Processing System) - operační část, nejvíce závislá na charakteru podniku
- **DSS** (Decision Support System) - pro manažerské plánování, grafický reprezentace dat z MIS
- **EDI** (Electronic Data Interchange) - výměna dat s jinými systémy
- **OIS** (Office IS) - podpora kancelářské a týmové práce

# GLOBÁLNÍ ARCHITEKTURA –EXTERNÍ VAZBY



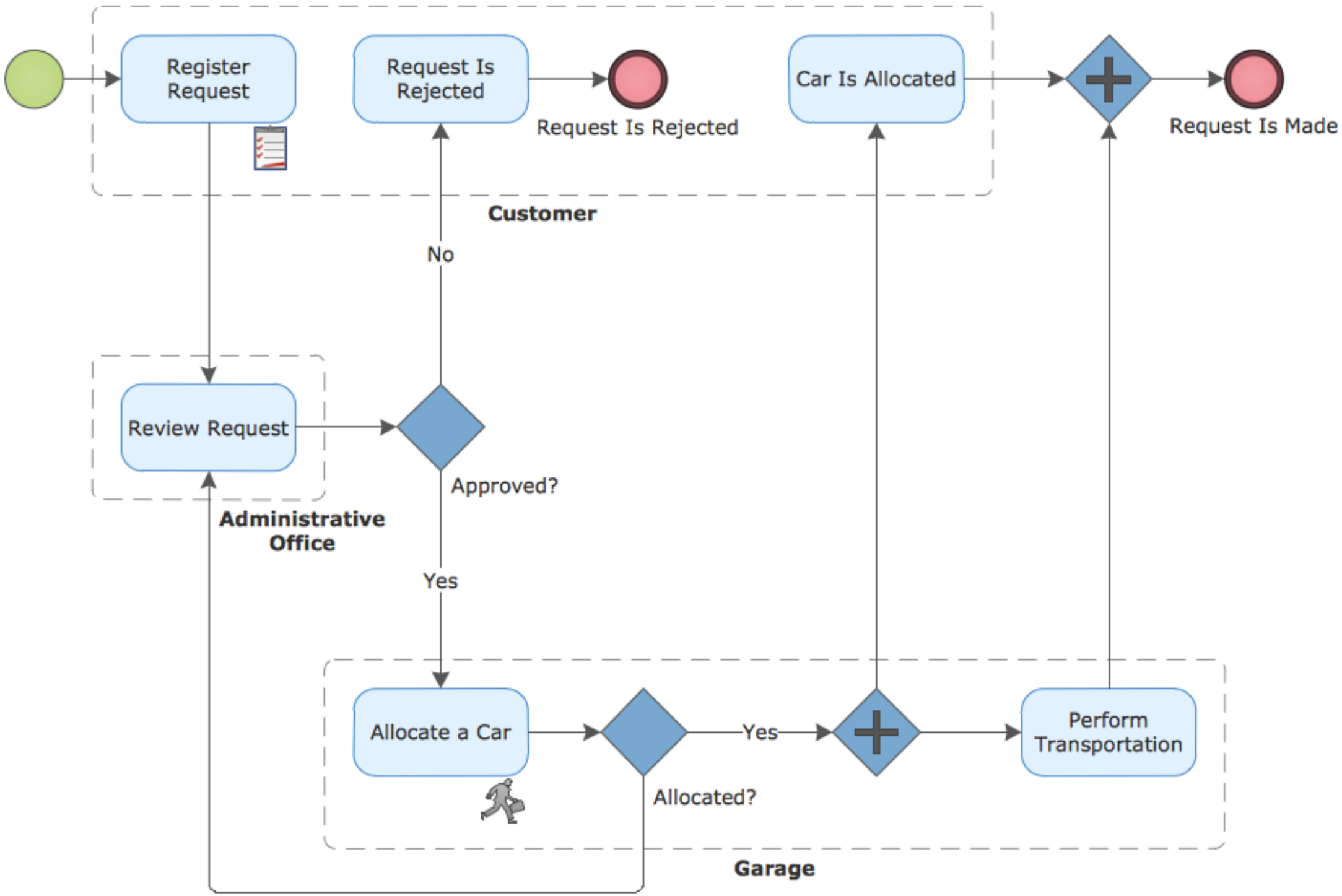
# DÍLČÍ ARCHITEKTURY



# PROCESNÍ

- Navržení vstupů, výstupů
- Cílem je co nejrychlejší reakce podniku na externí události
- Výsledkem návrhu je určení klíčových externích událostí - vazby podniku s okolím
- Nástroje
  - Kontextový diagram
  - Procesní diagram

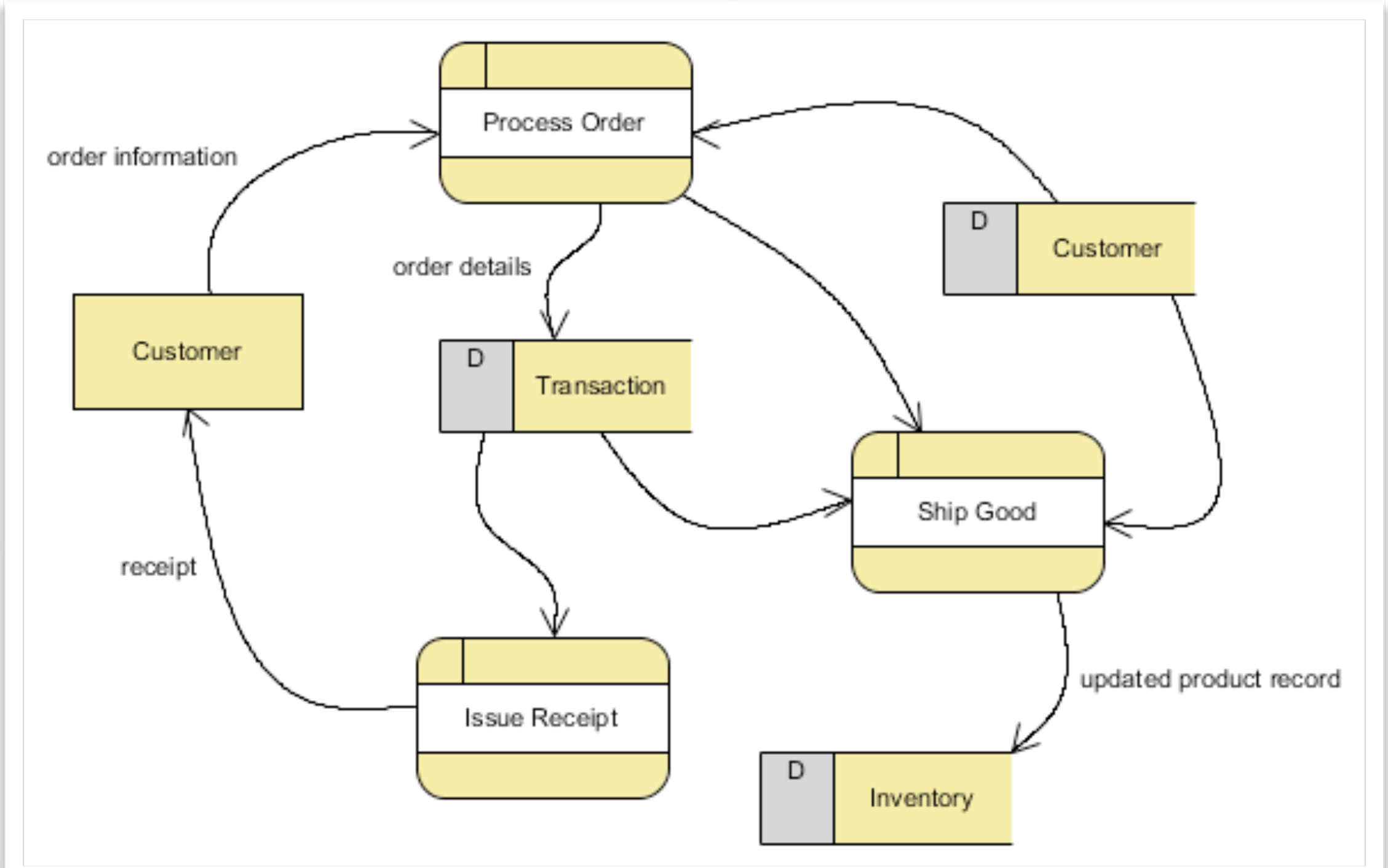
# Taxi Order Process



# FUNKČNÍ

- Navazuje na procesní architekturu
- Hierarchický rozpad požadovaných funkcí a služeb IS
- Nejnižší úroveň z pohledu uživatele - elementární funkce IS (transakce)
- Nástroje
  - DFD (Data Flow Diagram)
  - Slovní popis funkcí

# DFD


Process Order
order information
order details
D
Customer
Customer
D
Transaction
Ship Good
receipt
Issue Receipt
updated product record
D
Inventory

Employee

Hours Worked

D2 Employee Time File

Employee Time Record

D1 Employee Master

Employee Record

Process 1 has no output.

Process 2 has no input. The Gross Pay data flow is going in the wrong direction.

1 Calculate Gross Pay

Gross Pay

2 Calculate Withholding Amount

Withholding

3 Calculate Net Pay

Net Pay

An external entity should not directly connect to a data store.

4 Print Employee Paycheck

Employee Record

D1 Employee Master

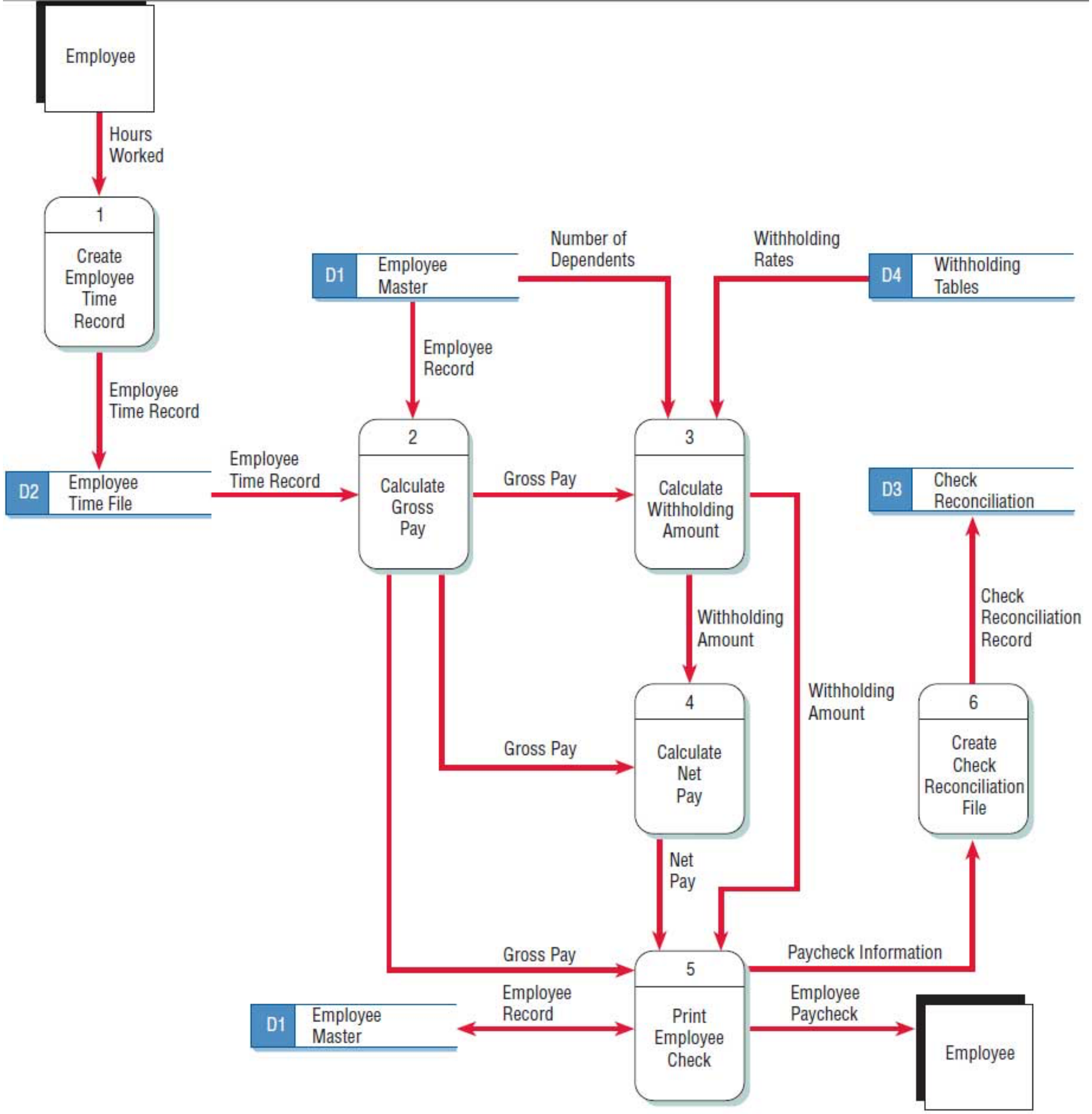
A data store should not directly connect to another data store.

Check Reconciliation Record

Employee Paycheck

D3 Check Reconciliation

Employee

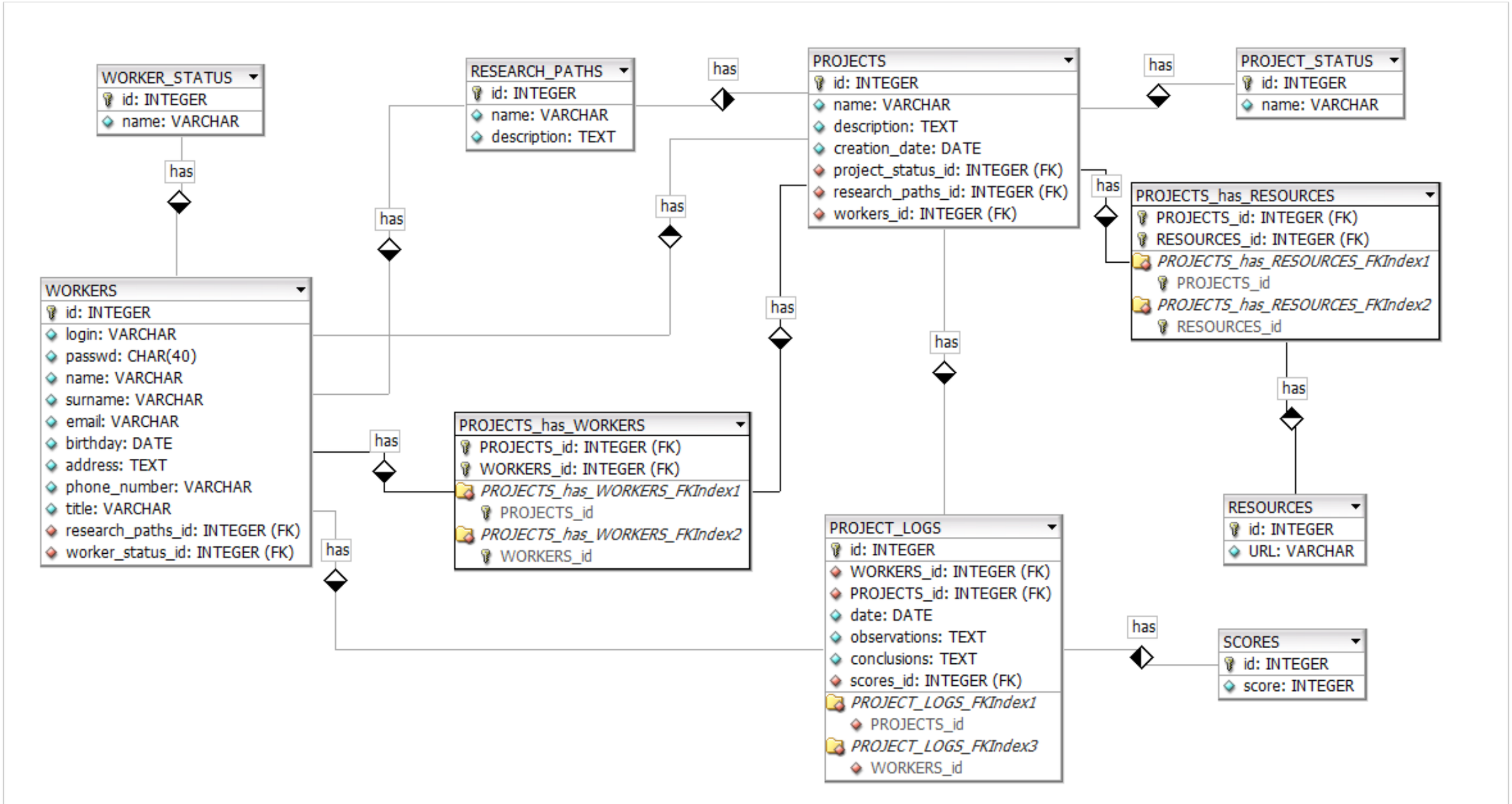

Employee
Hours Worked
1
Create Employee Time Record
Employee Time Record
D2
Employee Time File
D1
Employee Master
Number of Dependents
Withholding Rates
D4
Withholding Tables
Employee Record
Employee Time Record
2
Calculate Gross Pay
Gross Pay
3
Calculate Withholding Amount
D3
Check Reconciliation
Check Reconciliation Record
Withholding Amount
4
Calculate Net Pay
Gross Pay
Withholding Amount
6
Create Check Reconciliation File
Net Pay
Gross Pay
5
Print Employee Check
Paycheck Information
Employee Record
D1
Employee Master
Employee Paycheck
Employee

# DATOVÁ ARCHITEKTURA

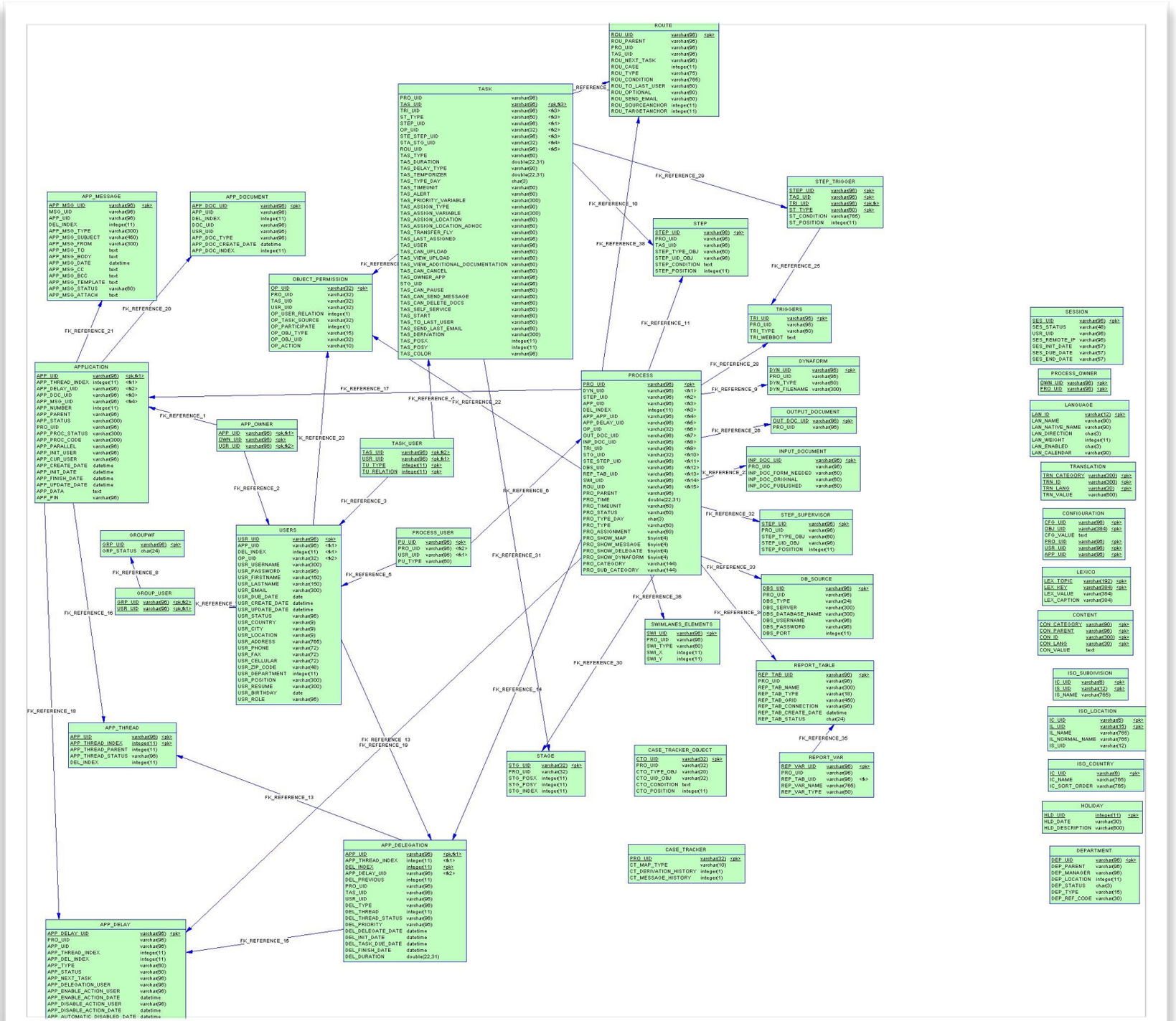
- Vychází z analýzy potřebných datových objektů
- Předchůdcem DA může být konceptuální model
- Na základě datové architektury se navrhují datové entity, vazby, atributy.

- Nástroje
    - E-R diagram

# E-R DIAGRAM

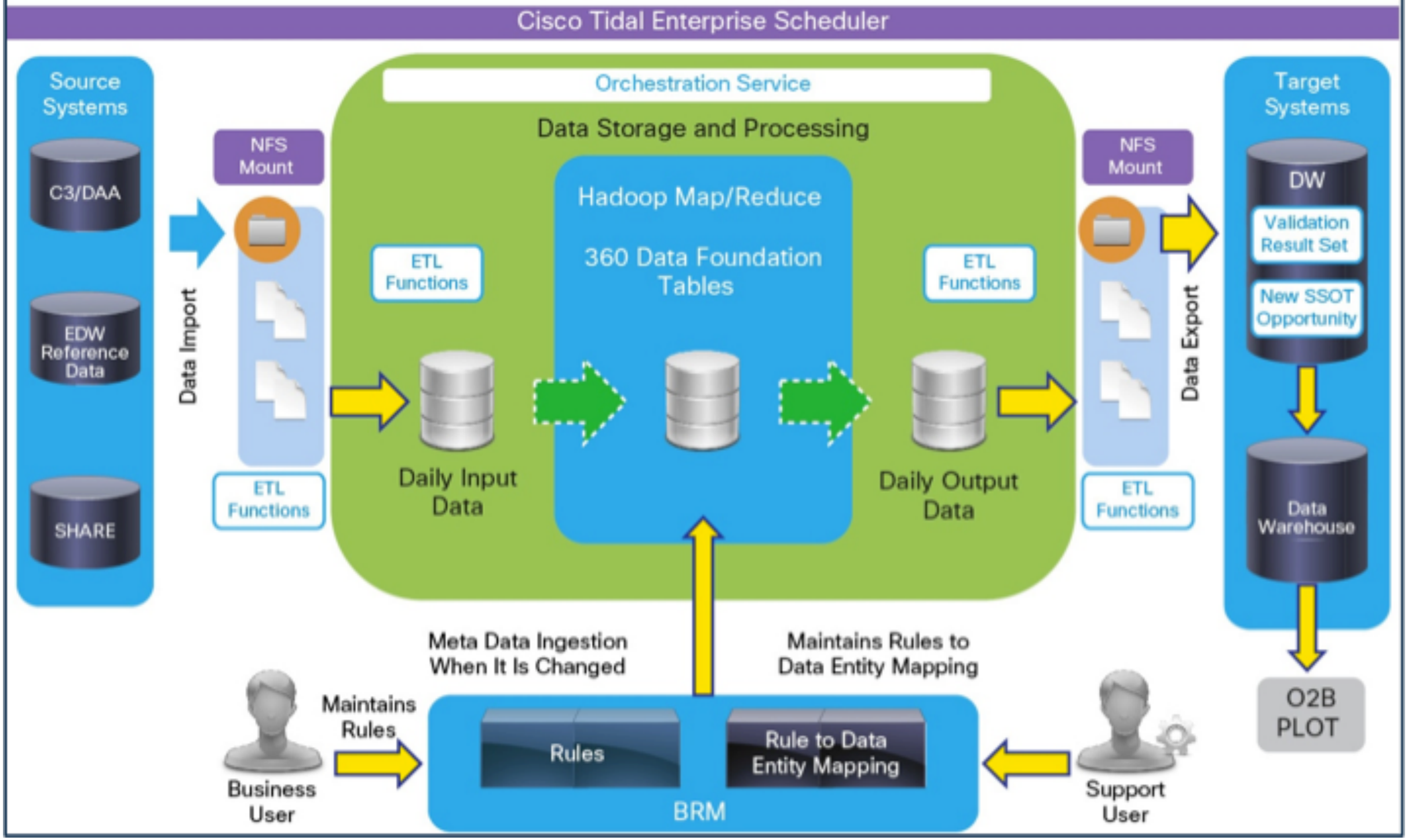

WORKER_STATUS
id: INTEGER
name: VARCHAR
has
WORKERS
id: INTEGER
login: VARCHAR
passwd: CHAR(40)
name: VARCHAR
surname: VARCHAR
email: VARCHAR
birthday: DATE
address: TEXT
phone_number: VARCHAR
title: VARCHAR
research_paths_id: INTEGER (FK)
worker_status_id: INTEGER (FK)
RESEARCH_PATHS
id: INTEGER
name: VARCHAR
description: TEXT
has
PROJECTS
id: INTEGER
name: VARCHAR
description: TEXT
creation_date: DATE
project_status_id: INTEGER (FK)
research_paths_id: INTEGER (FK)
workers_id: INTEGER (FK)
has
PROJECT_STATUS
id: INTEGER
name: VARCHAR
has
PROJECTS_has_RESOURCES
PROJECTS_id: INTEGER (FK)
RESOURCES_id: INTEGER (FK)
PROJECTS_has_RESOURCES_FKIndex1
PROJECTS_id
PROJECTS_has_RESOURCES_FKIndex2
RESOURCES_id
has
RESOURCES
id: INTEGER
URL: VARCHAR
has
PROJECTS_has_WORKERS
PROJECTS_id: INTEGER (FK)
WORKERS_id: INTEGER (FK)
PROJECTS_has_WORKERS_FKIndex1
PROJECTS_id
PROJECTS_has_WORKERS_FKIndex2
WORKERS_id
has
has
has
has
has
PROJECT_LOGS
id: INTEGER
WORKERS_id: INTEGER (FK)
PROJECTS_id: INTEGER (FK)
date: DATE
observations: TEXT
conclusions: TEXT
scores_id: INTEGER (FK)
PROJECT_LOGS_FKIndex1
PROJECTS_id
PROJECT_LOGS_FKIndex3
WORKERS_id
has
SCORES
id: INTEGER
score: INTEGER

# E-R DIAGRAM STŘEDNĚ VELKÉHO SYSTÉMU…



# Architecture at High Level

## 360 Data Foundation



# SOFTWAROVÁ

- Množina programových jednotek (modulů)
- Každý modul je popsán:
    - Funkcemi
    - Vstupem, výstupem, řídícími daty
    - Algoritmem přechodu Vstup-Výstup
    - Vývojové prostředí (prog. jazyk)
    - Provozní prostředí (OS, DB)

# TYPY SW ARCHITEKTUR

- **Lineární** - sekvenční uspořádání funkcí, moc se nepoužívá
- **Hierarchická** - reprezentována stromovým grafem, přehledná ale nákladná
- **Síťová** - otevřená, flexibilní, nedefinuje podřízenost a nadřízenost funkcí
- **Vrstvená** - funkce uspořádány do několika vrstev (podobně jako ISO-OSI)

# JAK TO CHÁPAT JAKO VÝVOJÁŘ?

- SA popisuje jak je systém strukturován do
  - **Komponent** (Components)
  - **Konektorů** (Connectors)
  - **Rozhraní** (Interfaces) → ***Struktura*** *(topologie)*
- a jak spolu tyto komponenty interagují
  - scénáře
  - diagram stavů
  - sekvenční diagram
  - …

→ ***Dynamika*** *(chování)*

# KOMPONENTY

- Komponenta je stavební blok:
  - výpočetní jednotka nebo datový zdroj s definovaným rozhraním pro použití (vstup i výstup)
  - doručitelná část software (např. knihovna)
  - ideálně znovupoužitelná

# KOMPONENTY A OBJEKTY

- Rozdílná úroveň abstrakce
- Velikost
  - Objekty by měly být podstatně menší
  - Komponenta je řádově větší (knihovna objektů, celá aplikace)
- Architektonická komponenta je implementována několika objekty
- Životní cyklus
  - Objekty jsou vytvářeny vždy, když je chceme použít
  - Komponenty často zůstávají v paměti (po prvotním načtení)

# KONEKTORY

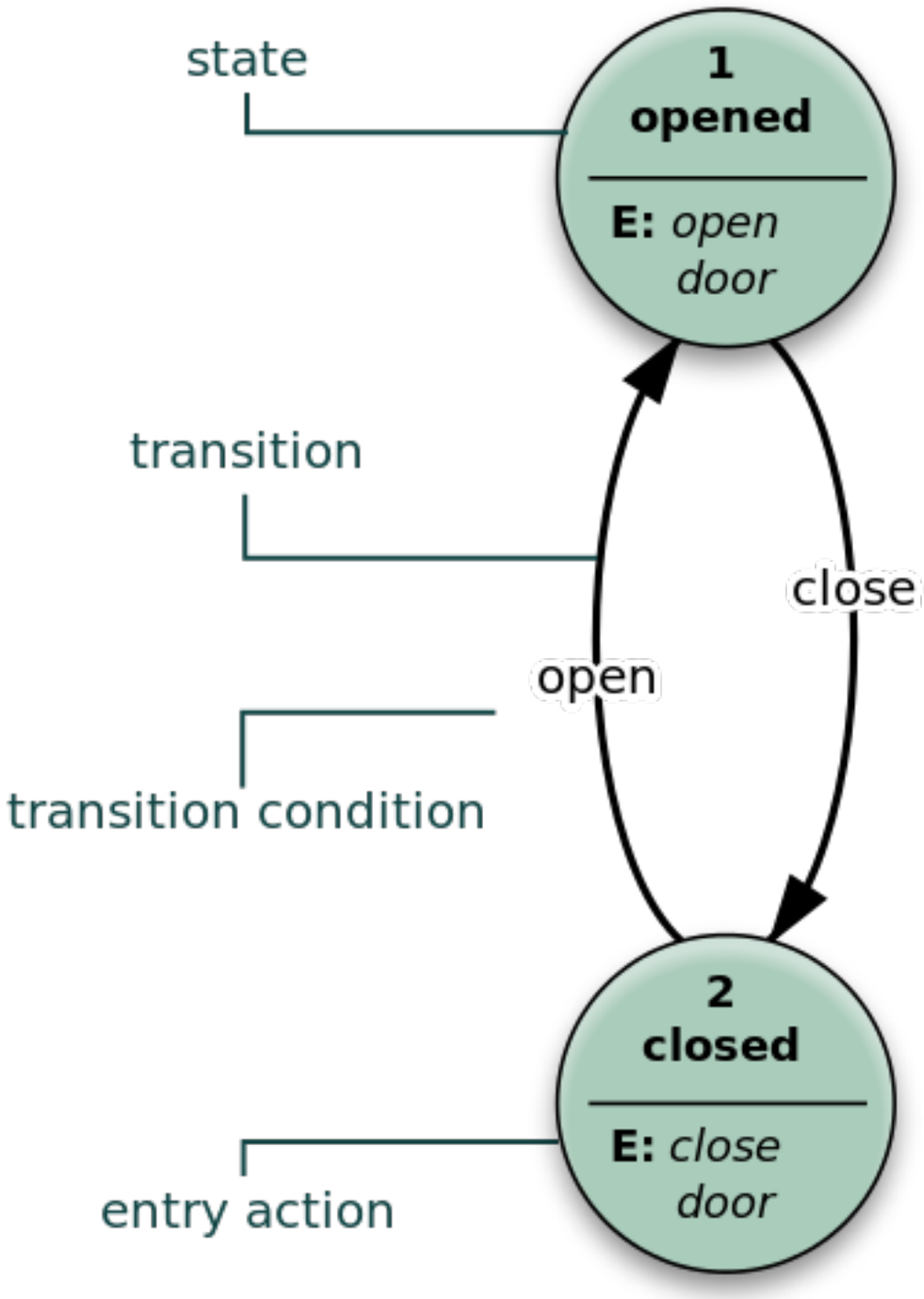
- Konektory je vytvořen pro realizaci interakce mezi komponentami:
  - zasílání událostí
  - rozhraní Klient-Server
  - zprávy a fronty zpráv (model publisher-subscriber)
  - sdílené proměnné
  - volání procedur (lokální i vzdálené)
  - roury (pipes)
- Konektory jsou explicitní a implicitní:
  - slouží pouze jako komunikační kanál mezi komponentami
  - mají svou vlastní logiku (např. filtry)

# ROZHRANÍ

- Rozhraní je externí spojení komponenty (nebo konektoru) s další komponentou a specifikuji jak ji použít
- Jak je lze popsat:
  - velmi volně (co je vstupem a co výstupem)
  - styl API (list funkcí)
  - velmi detailně (včetně protokolu, který je potřeba při komunikaci použít, např. SMTP)

# DYNAMIKA

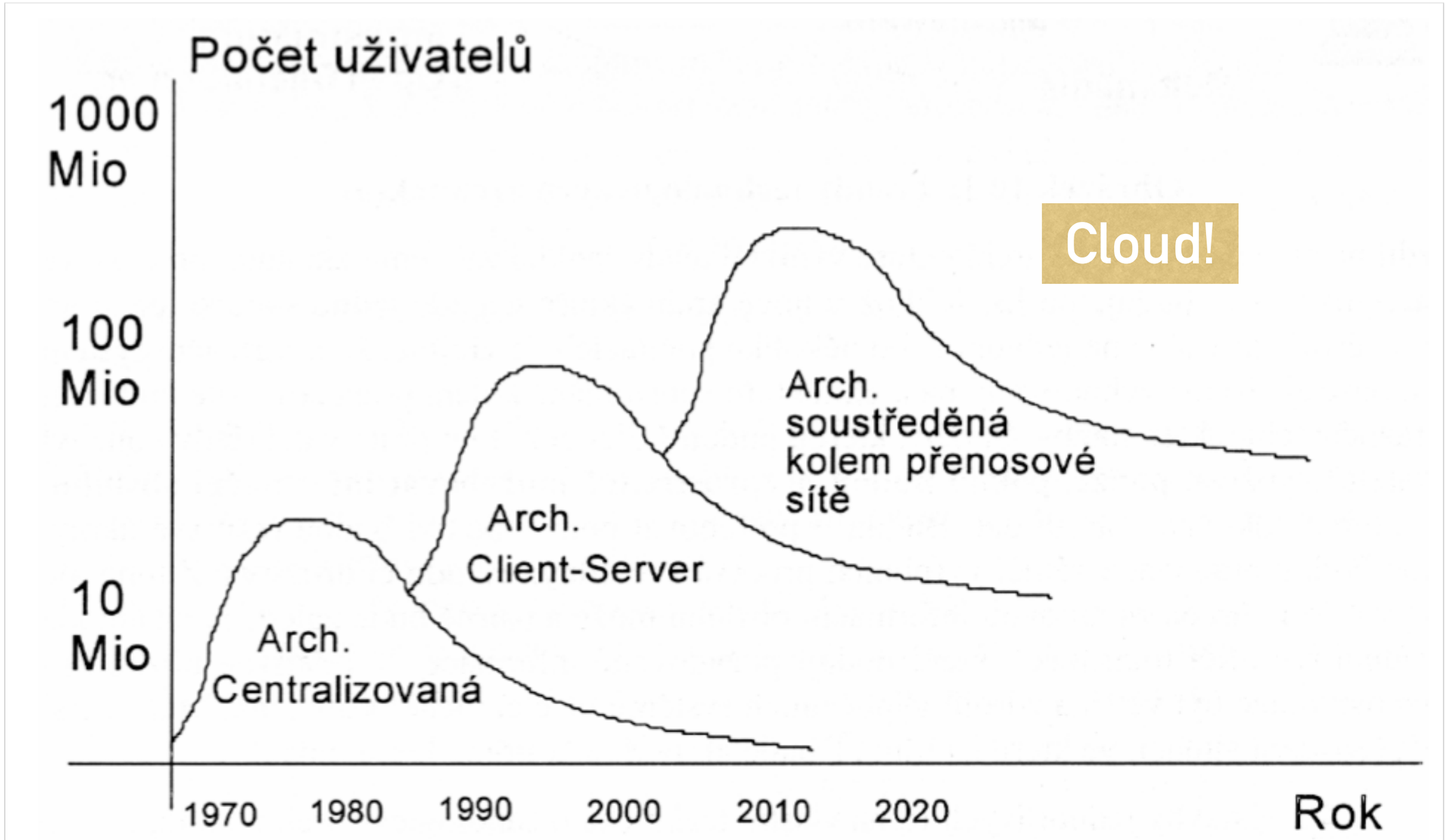
- Popsání interakcí komponent skrze konektory.
- Při popisu můžete využít:
  - Konečný automat
  - UML - diagram stavů, sekvenční diagram, diagram aktivit
  - Message Sequence Chart
  - State Diagram



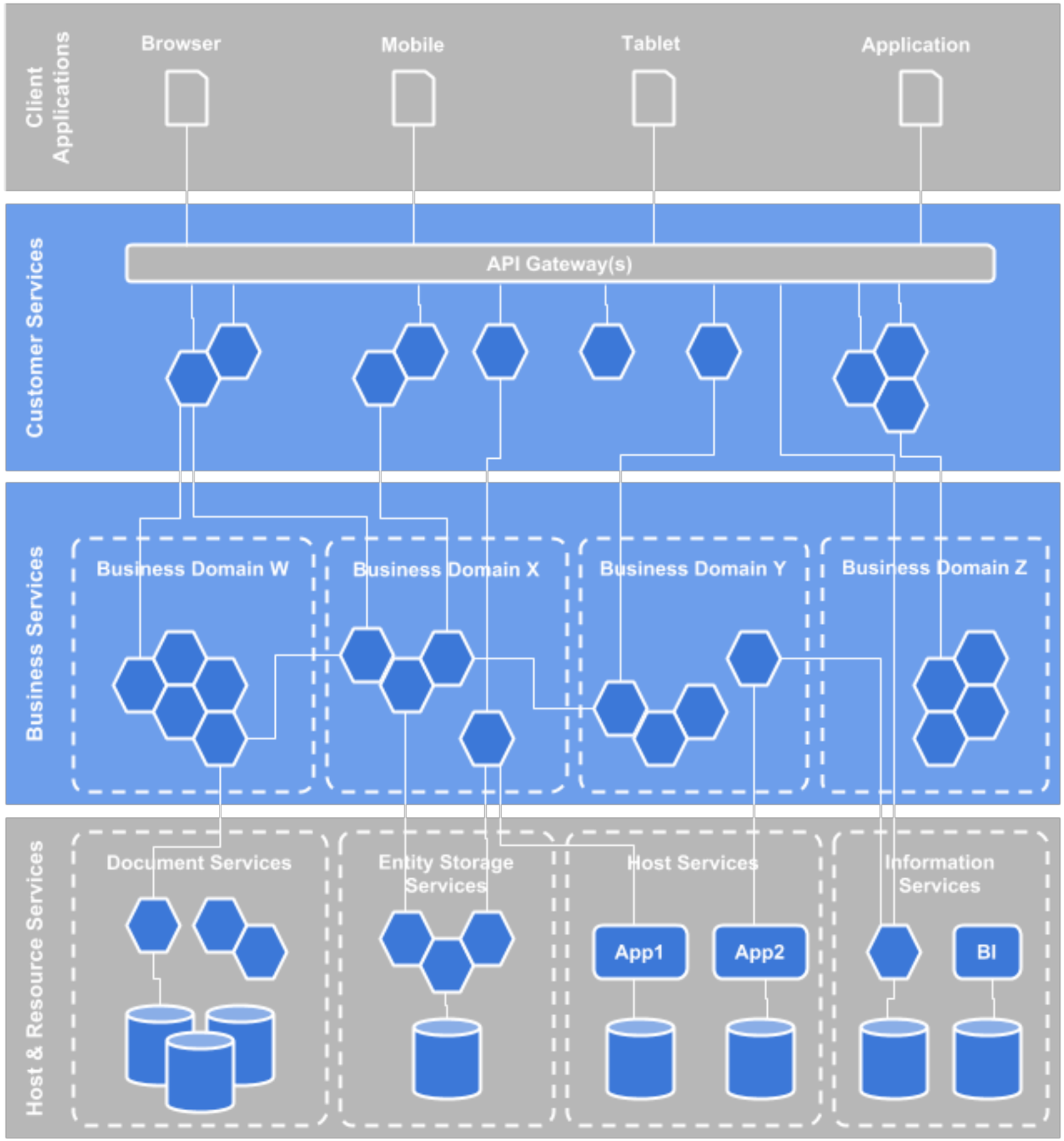
# TECHNOLOGICKÁ

- Dle režimu zpracování: dávkové, interaktivní, řízené událostmi (zprávami), v reálném čase
- Dle uspořádání: centralizovaná, decentralizovaná (bez vazeb), distribuované zpracování (několik serverů s připojenými stanicemi), kooperativní
- Dle vrstev: monolitická, dvouvrstvá, třívrstvá

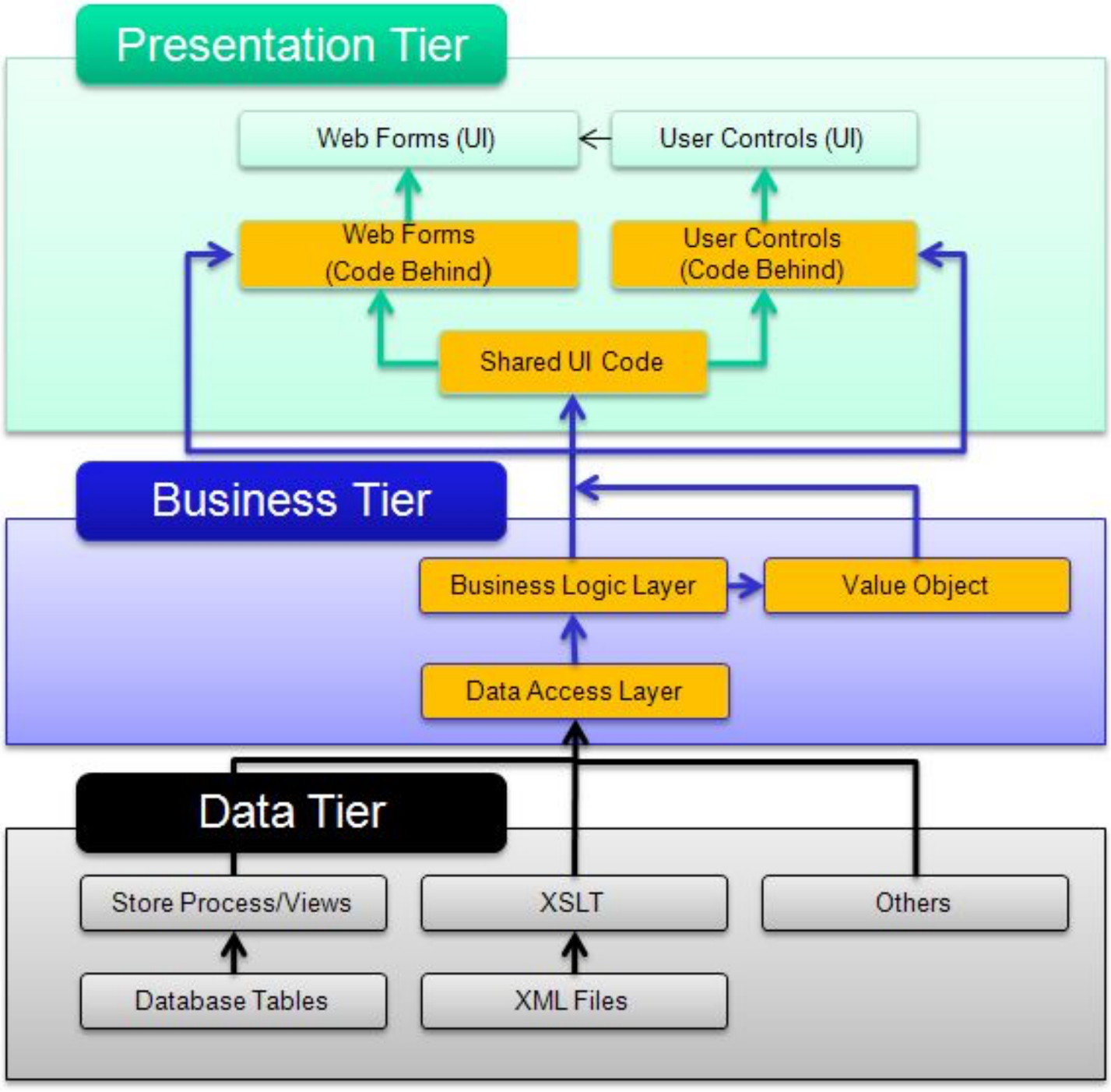
# VÝVOJ TECHNOLOGICKÉ ARCHITEKTURY V ČASE



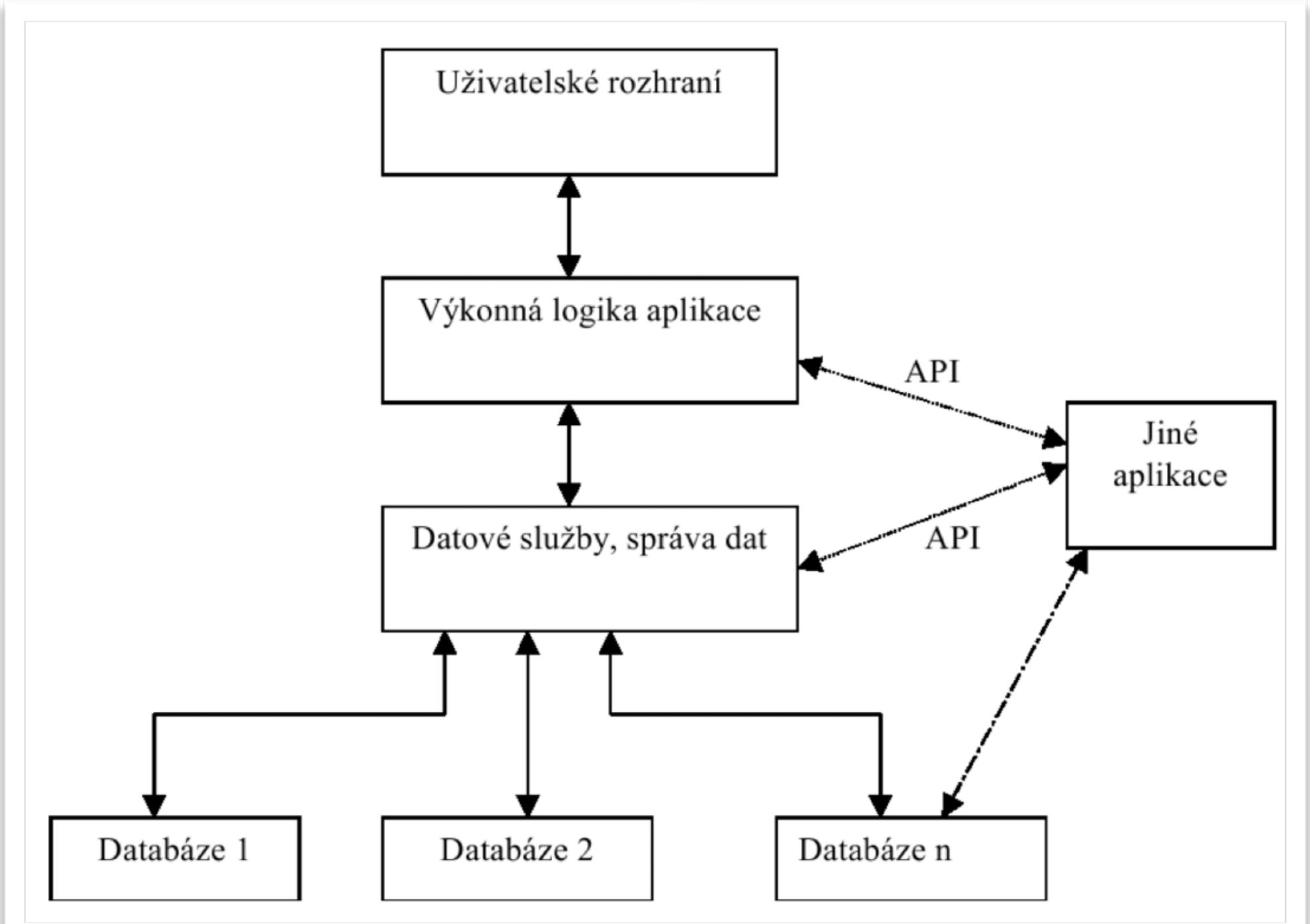
- Logika
- Prezentace
- Funkce
- Data



# DĚLENÍ KÓDU APLIKACE – LAYERS (TIERS)



# TŘÍVRSTVA STRUKTURA APLIKACÍ



# Příklad 3-vrstvé architektury

Prezentační služby
Prezentační logika
(Aplikační logika)

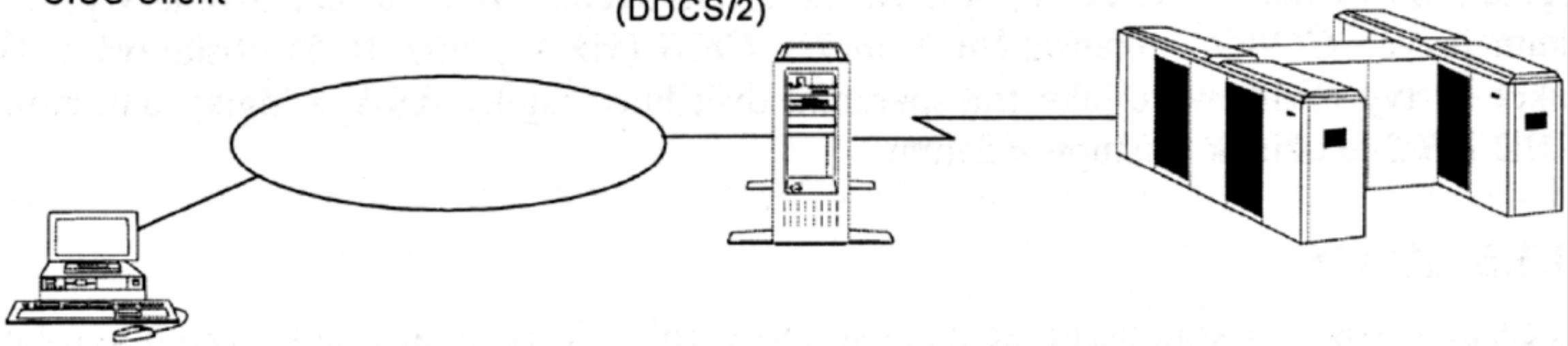
Aplikační logika
Datová logika
(Databázové služby)

Aplikační logika
Databázové služby

VisualAge
CICS Client

CICS OS/2 Server
DB2/2 Server
(DDCS/2)

CICS/ESA
DB2



VisualAge
DB2/2 Client

DB2/2 Server
DDCS/2

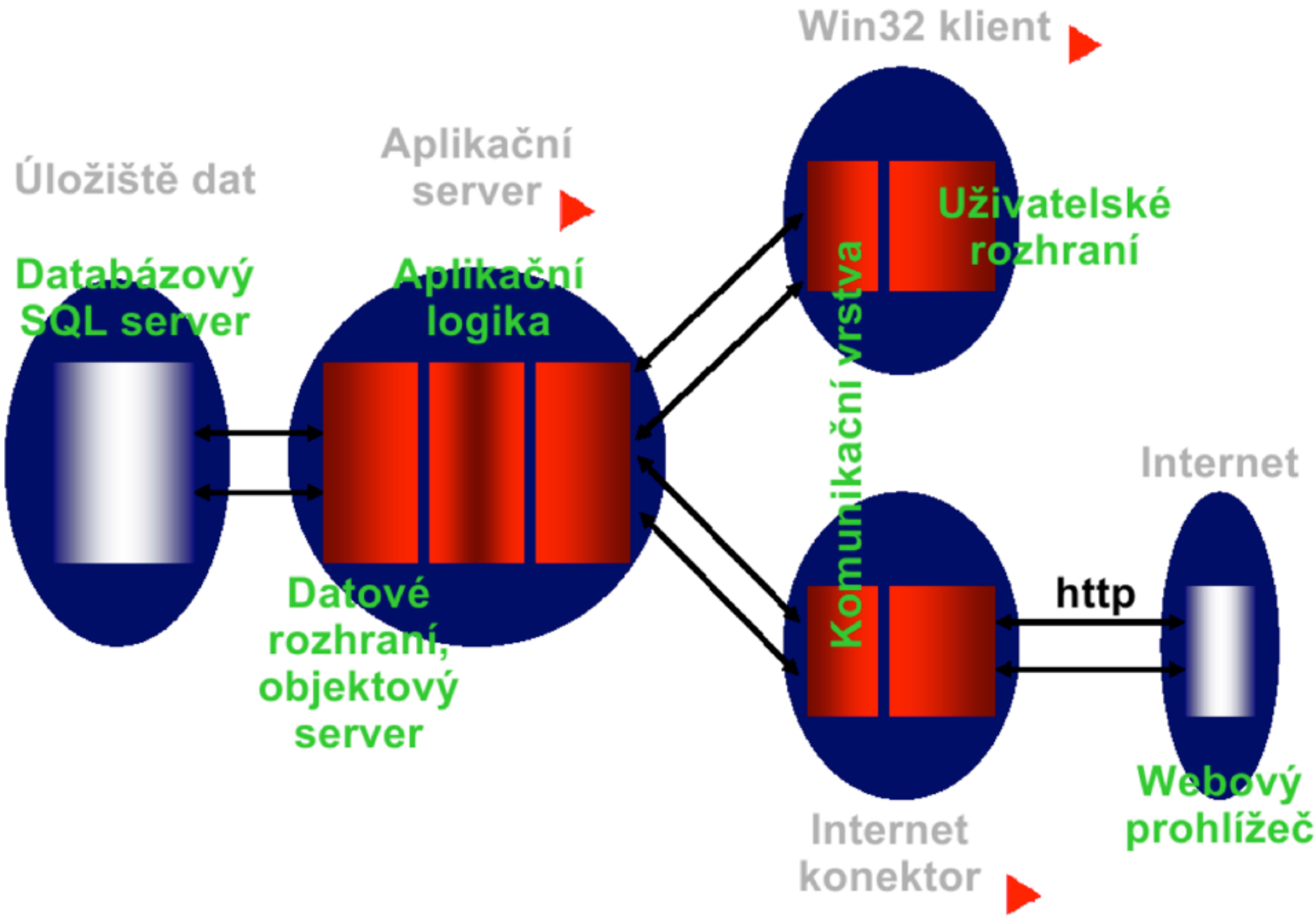
DB2

Prezentační služby
Prezentační logika
Aplikační logika
Datová logika

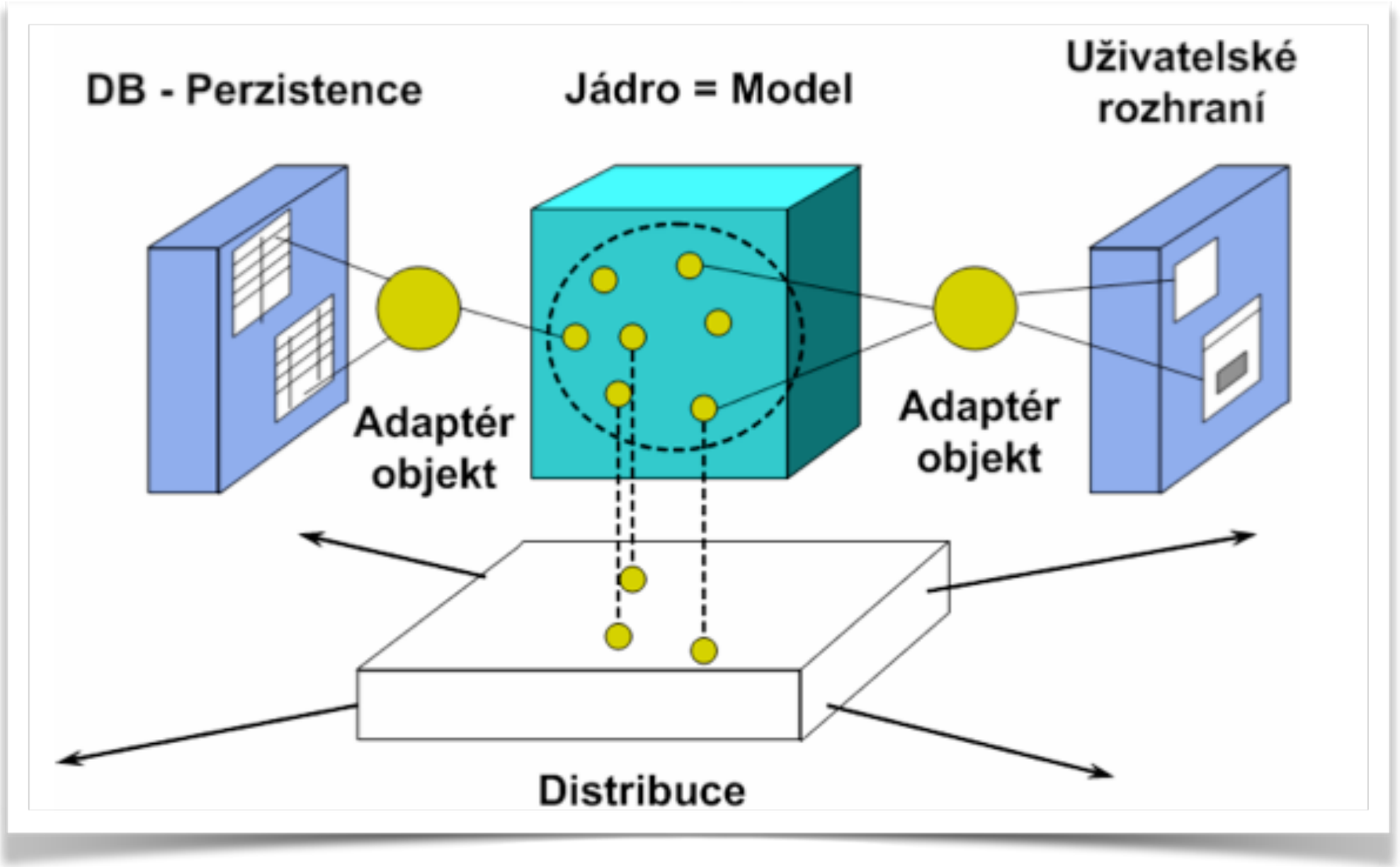
Databázové služby

# Příklad 2-vrstvé architektury

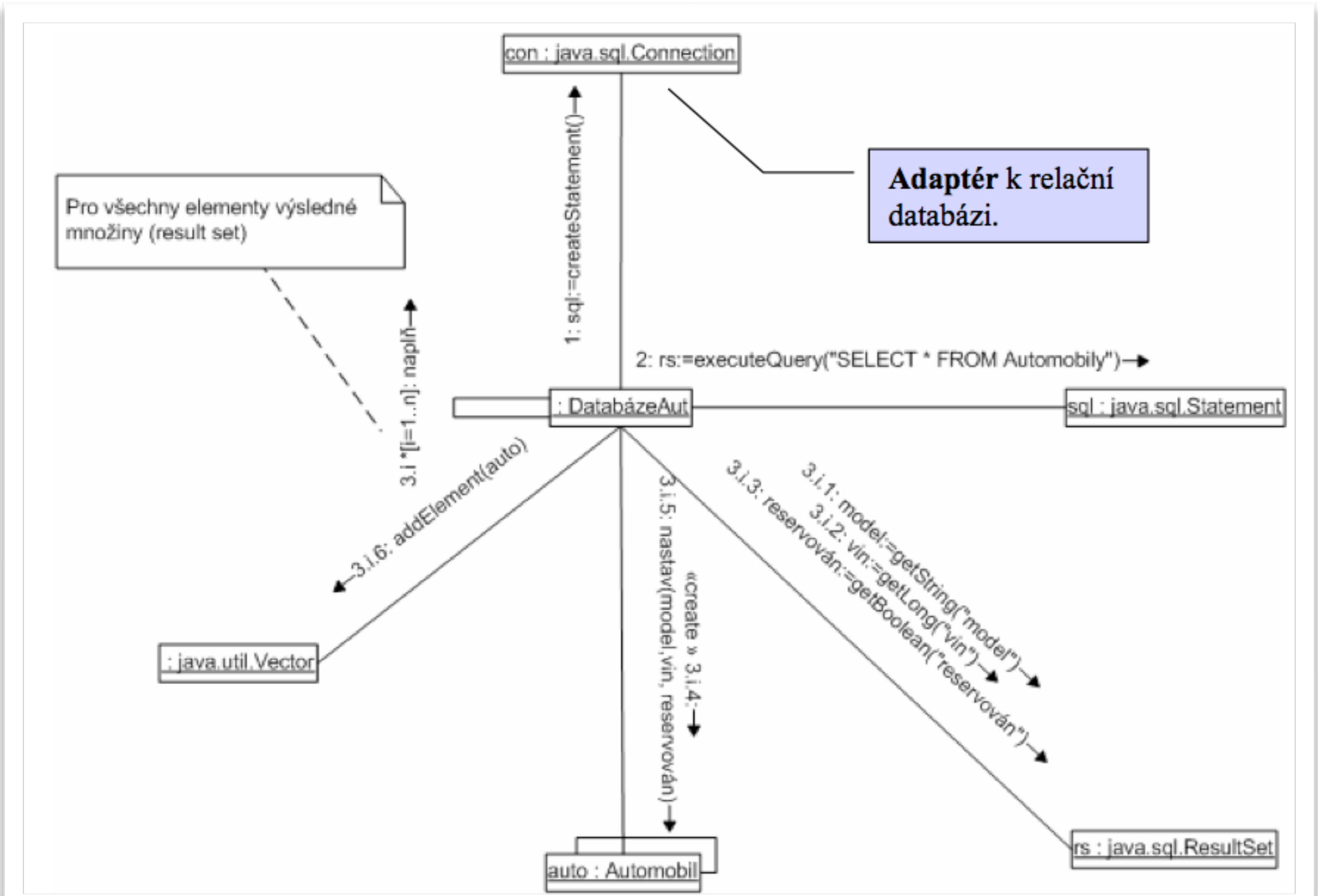
# Příklad vícevrstvé architektury QI



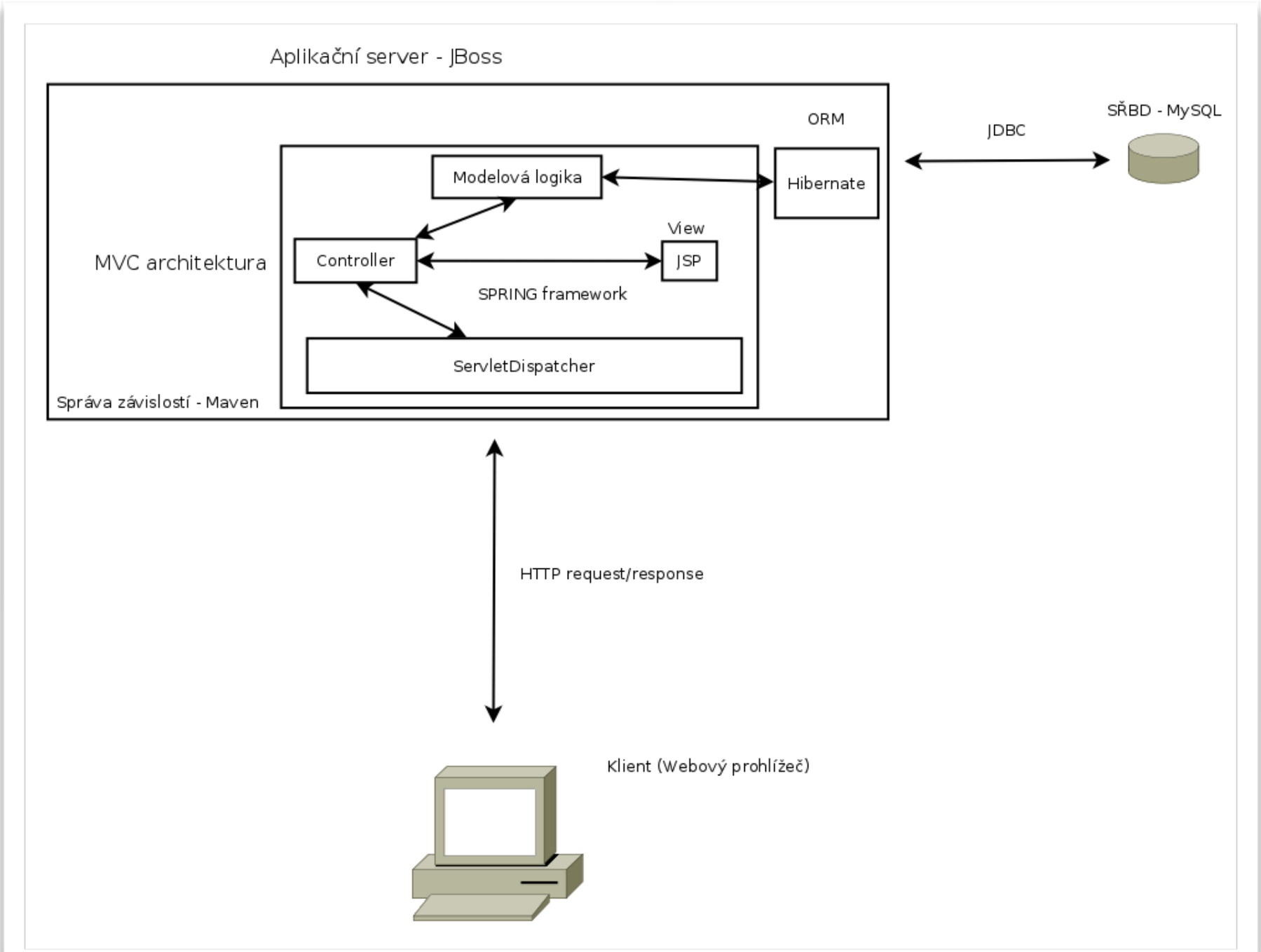
# JÁDRO SYSTÉMU



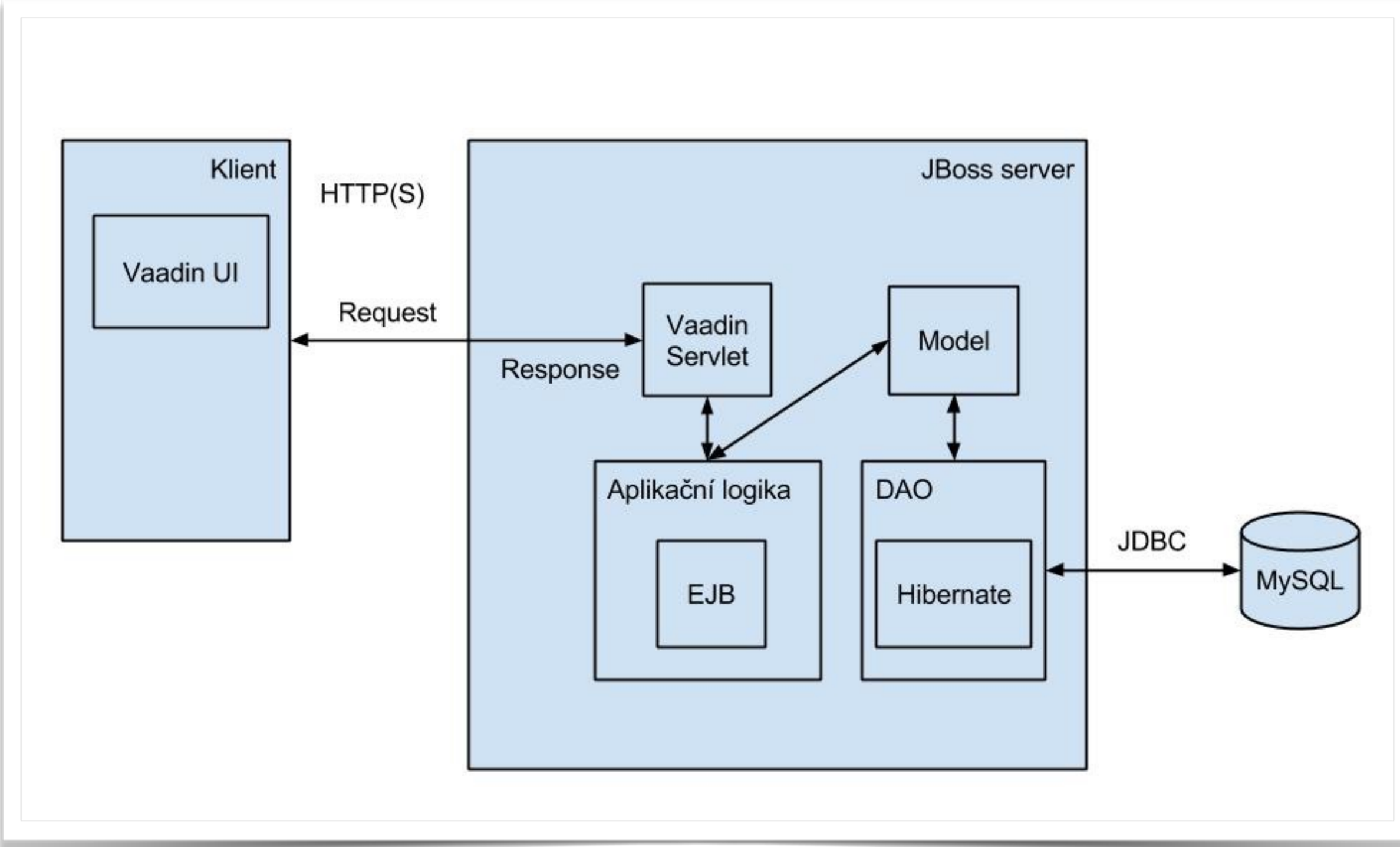
# ADAPTÉRY – PŘÍKLAD



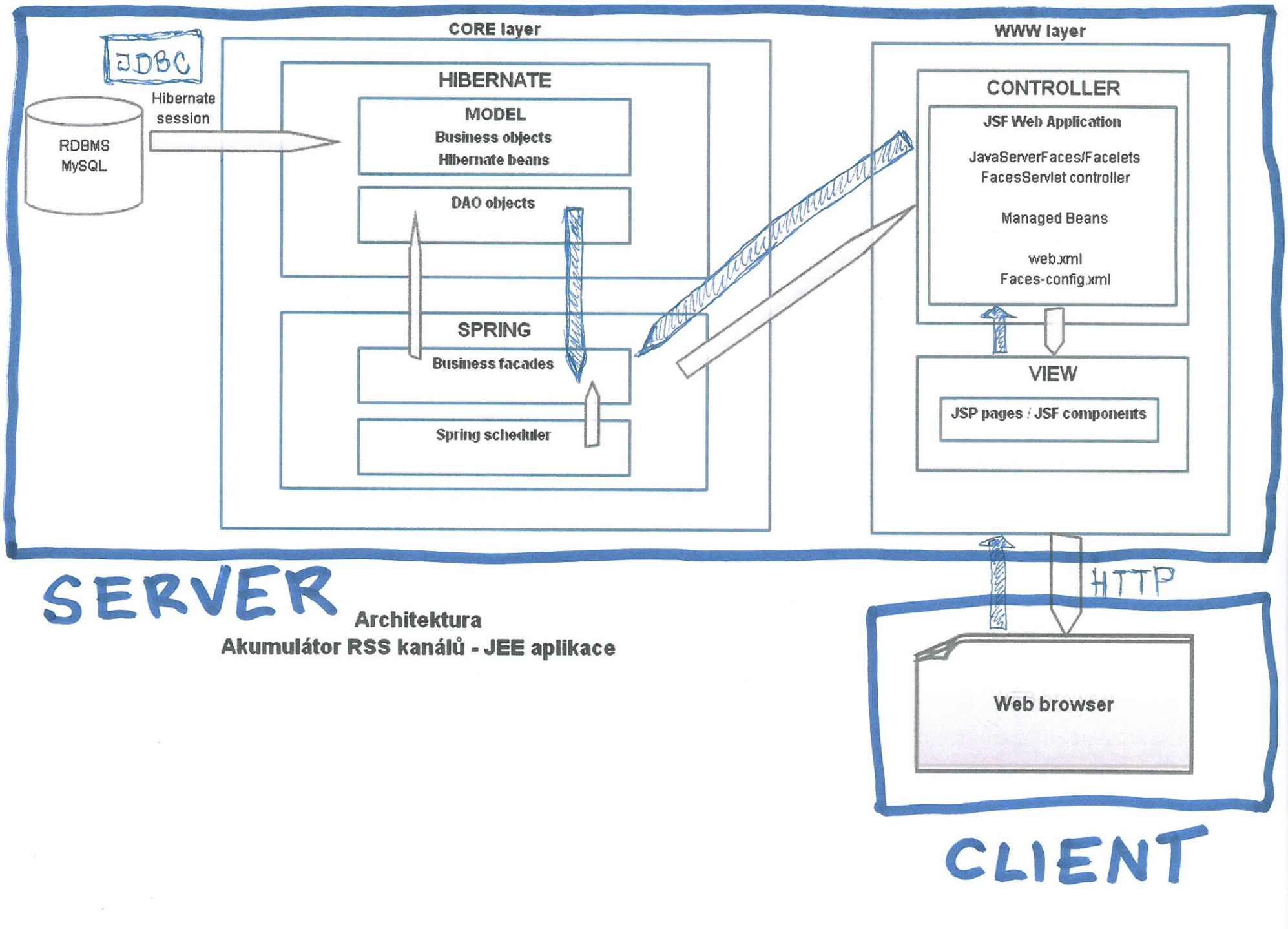
# PŘÍKLAD ARCHITEKTURY



# PŘÍKLAD ARCHITEKTURY



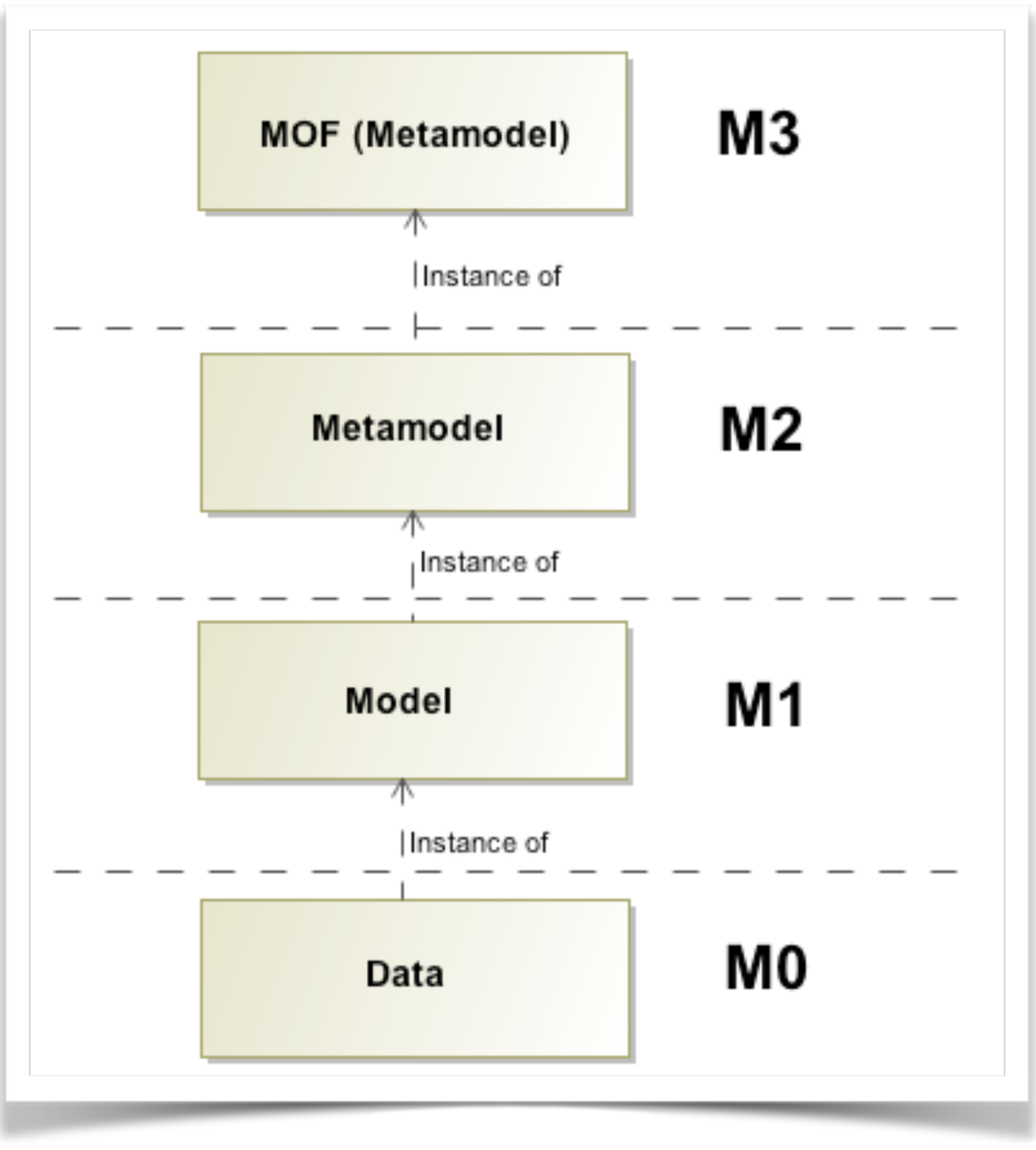
# PŘÍKLAD ARCHITEKTURY



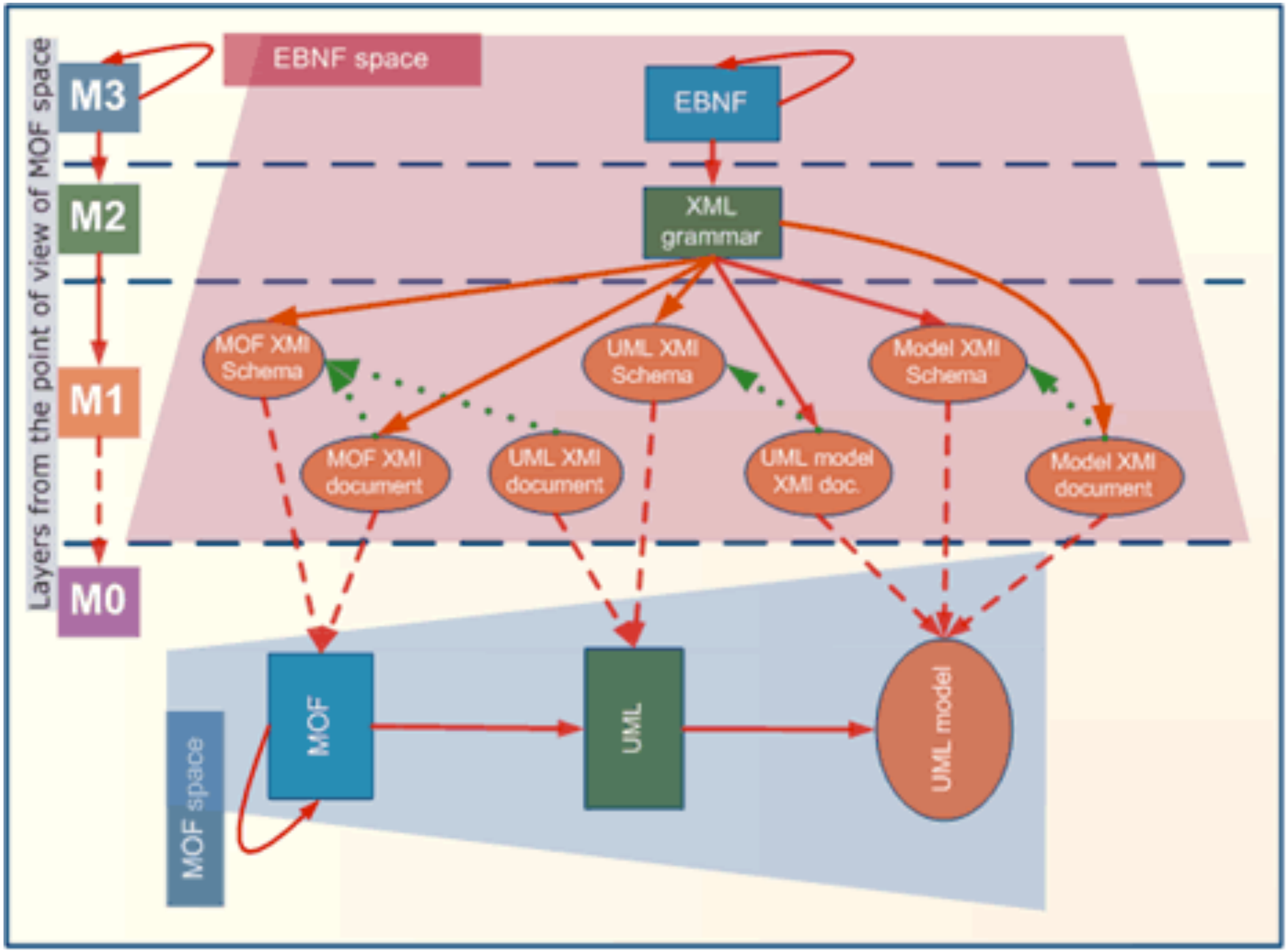

**Architektura**
**Akumulátor RSS kanálů - JEE aplikace**

# VÝBĚR ARCHITEKTURY JE VŽDY KOMPROMIS

# MODEL-DRIVEN ARCHITECTURE

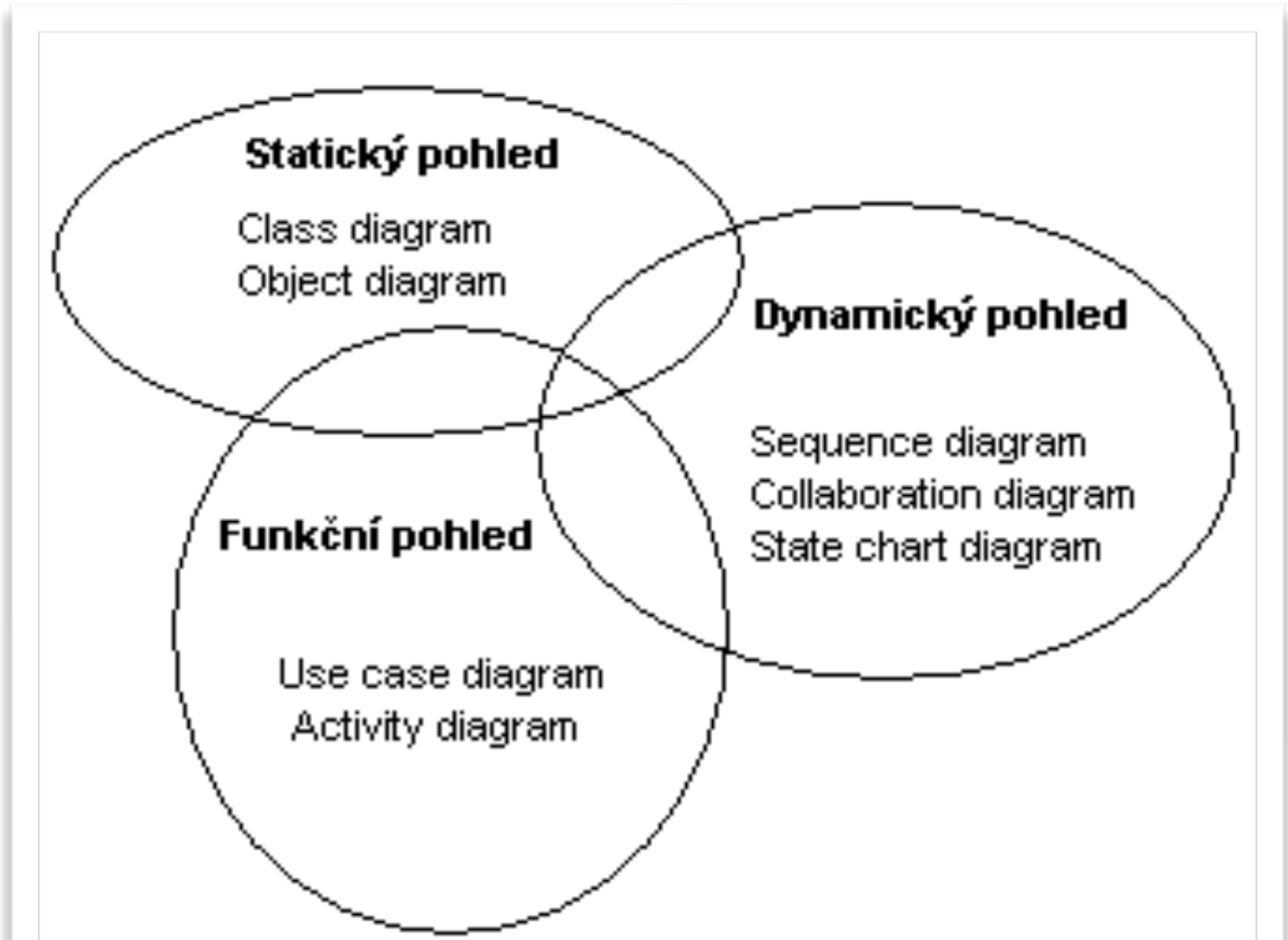


# MODEL-DRIVEN ARCHITECTURE

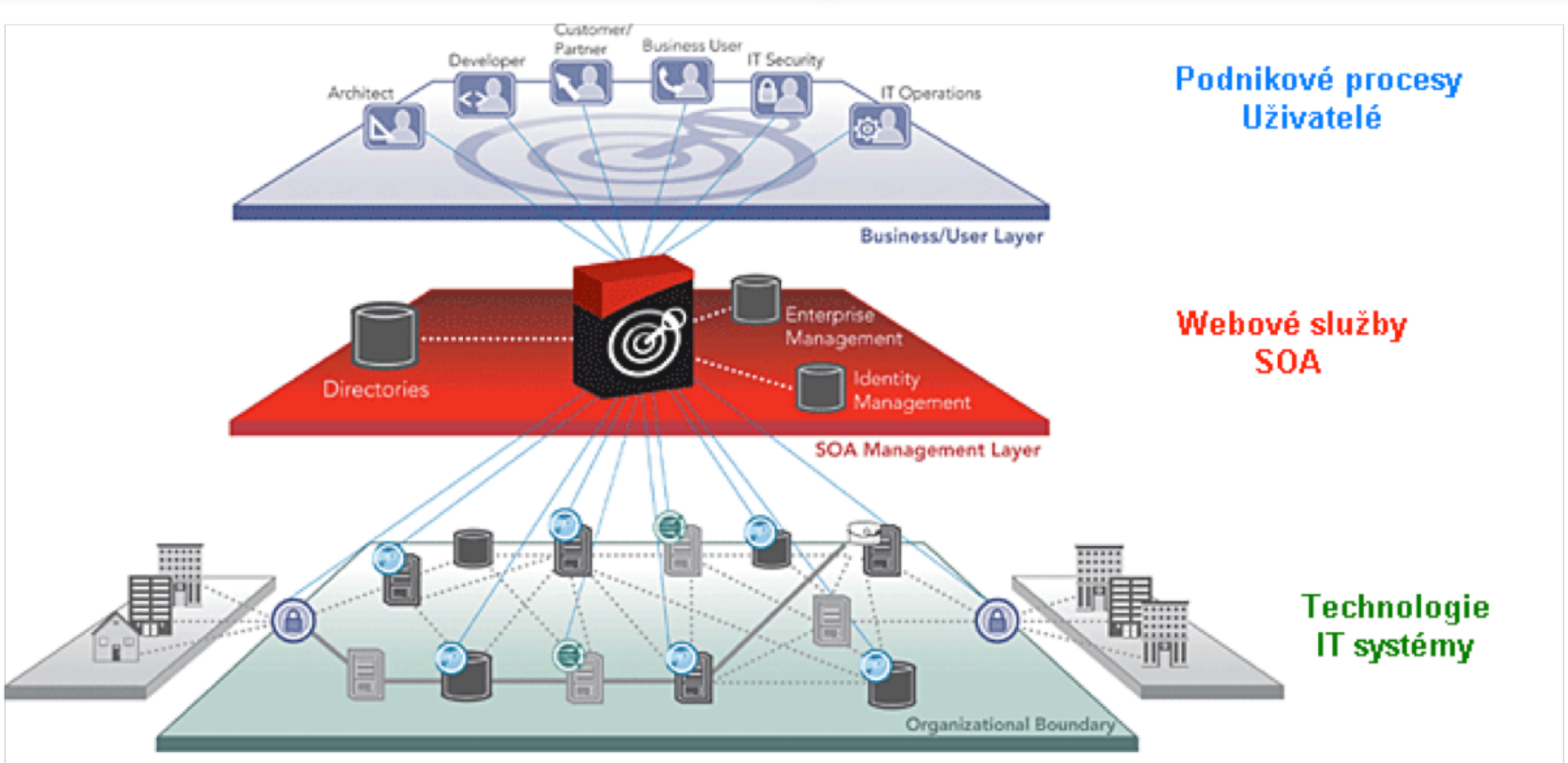


# ARCHITEKTURA 4+1 A UML

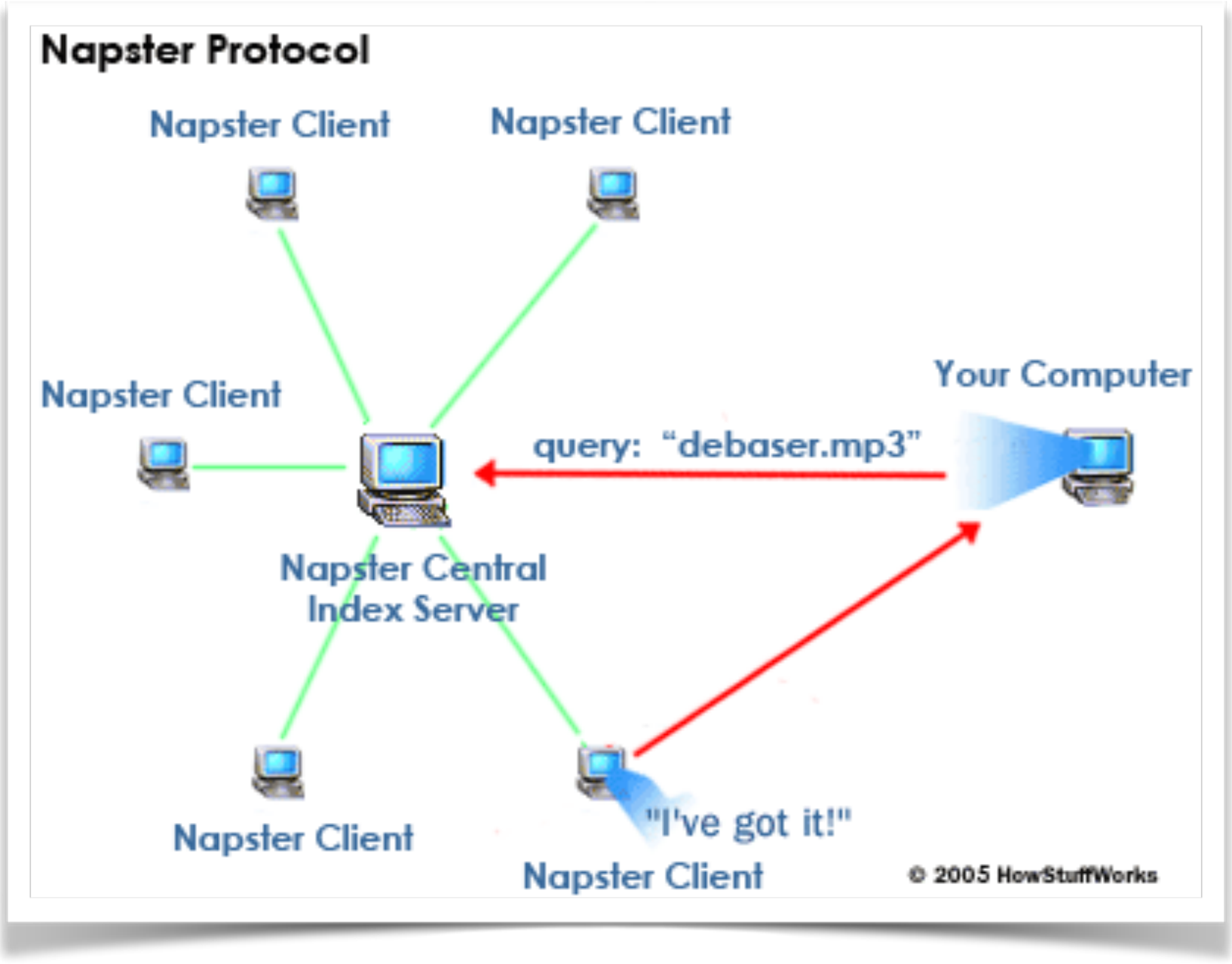
- Statický pohled
- Dynamický pohled
- Funkční pohled



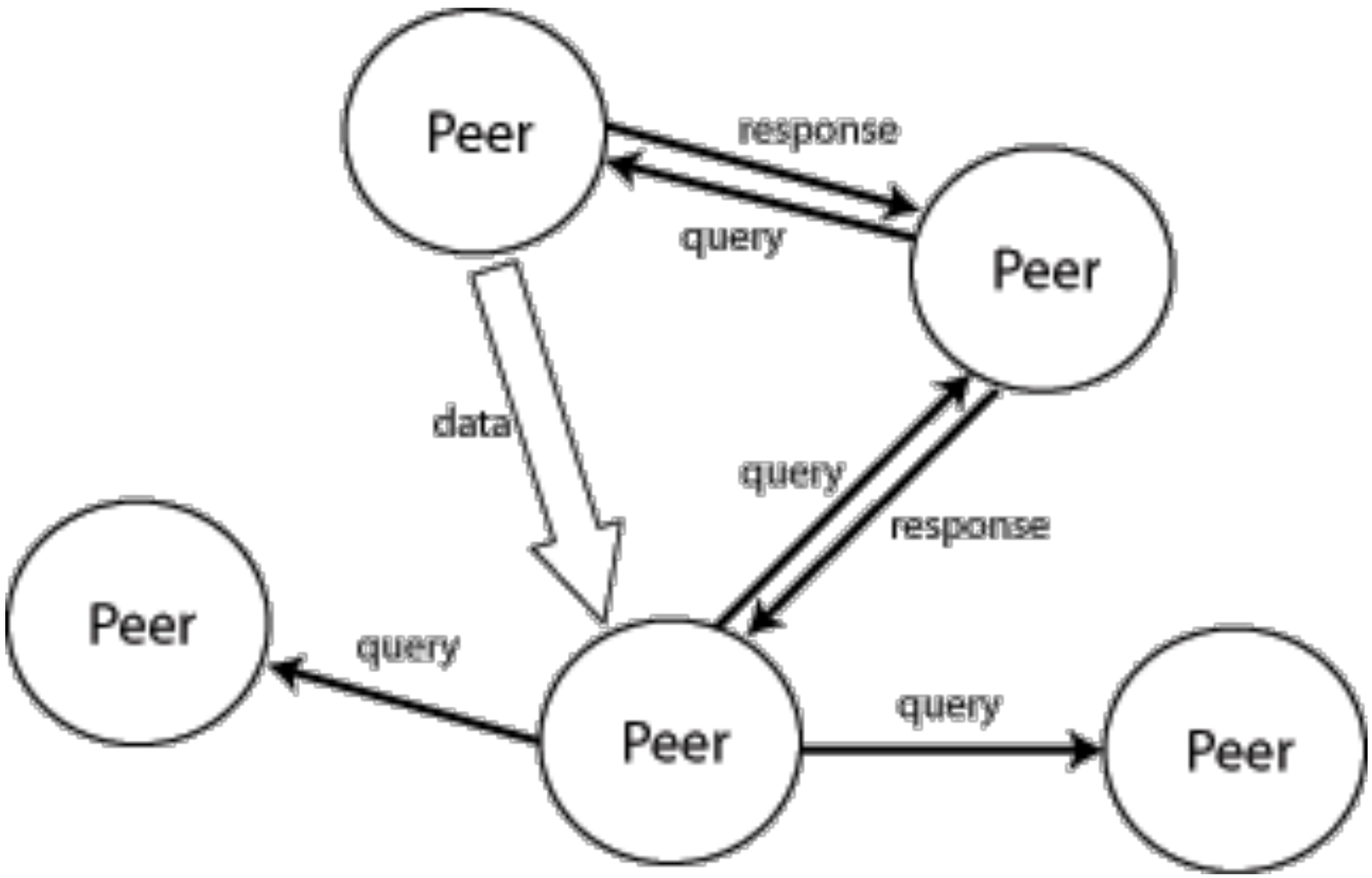
# SERVISNĚ ORIENTOVANÁ ARCHITEKTURA



# P2P ARCHITEKTURA – NAPSTER



# P2P ARCHITEKTURA – GNUTELLA



# P2P ARCHITEKTURA – SUPERNODES