

**MUNI**  
FI

# **Přednáška 3**

## **Principy softwarového vývoje**

### 3. Principy softwarového vývoje

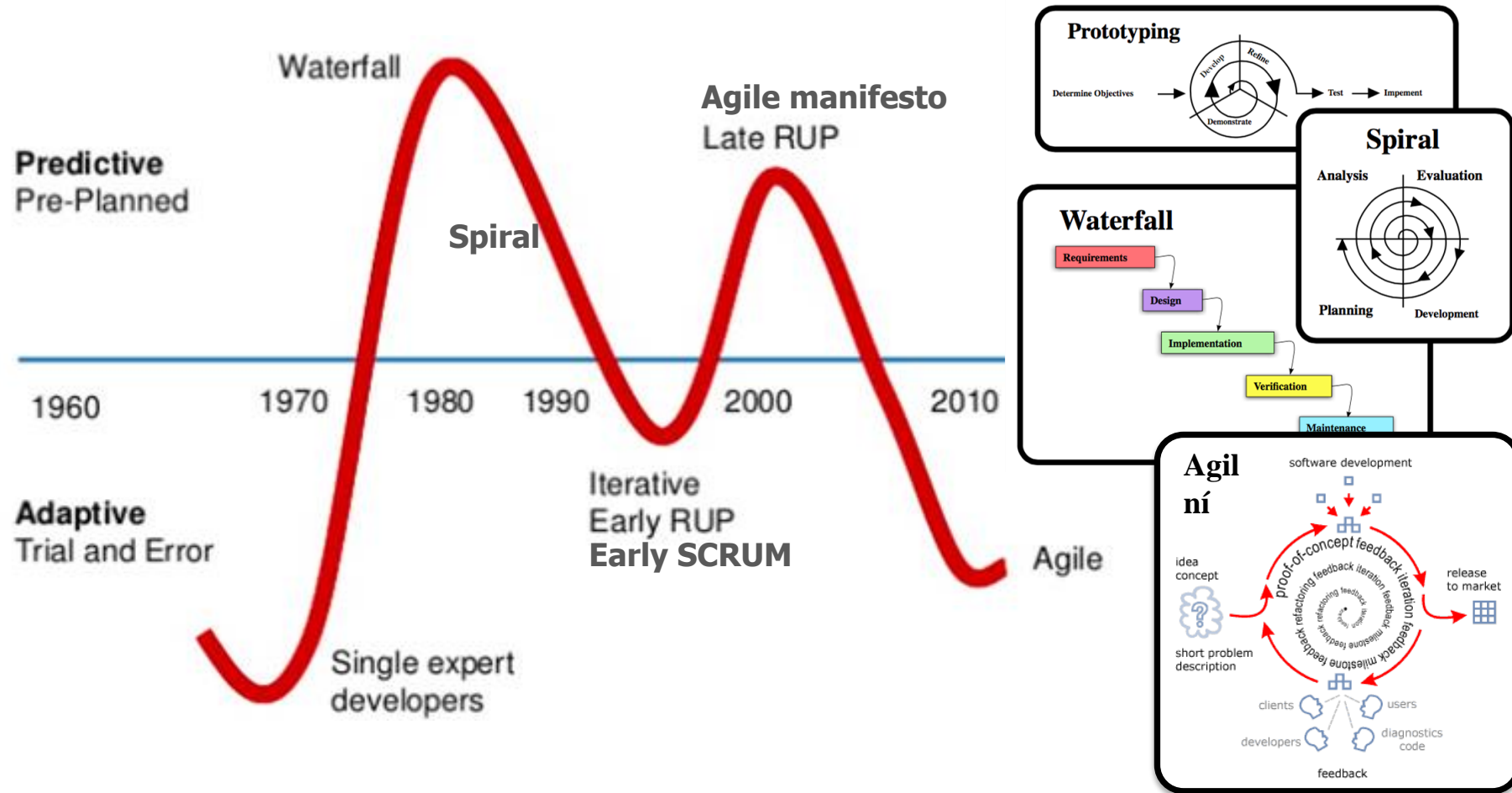
- Metodiky vývoje softwarových systémů
- Agilní vývoj
- Role modelů v softwarovém vývoji

#### **Domácí práce a příprava na tuto přednášku**

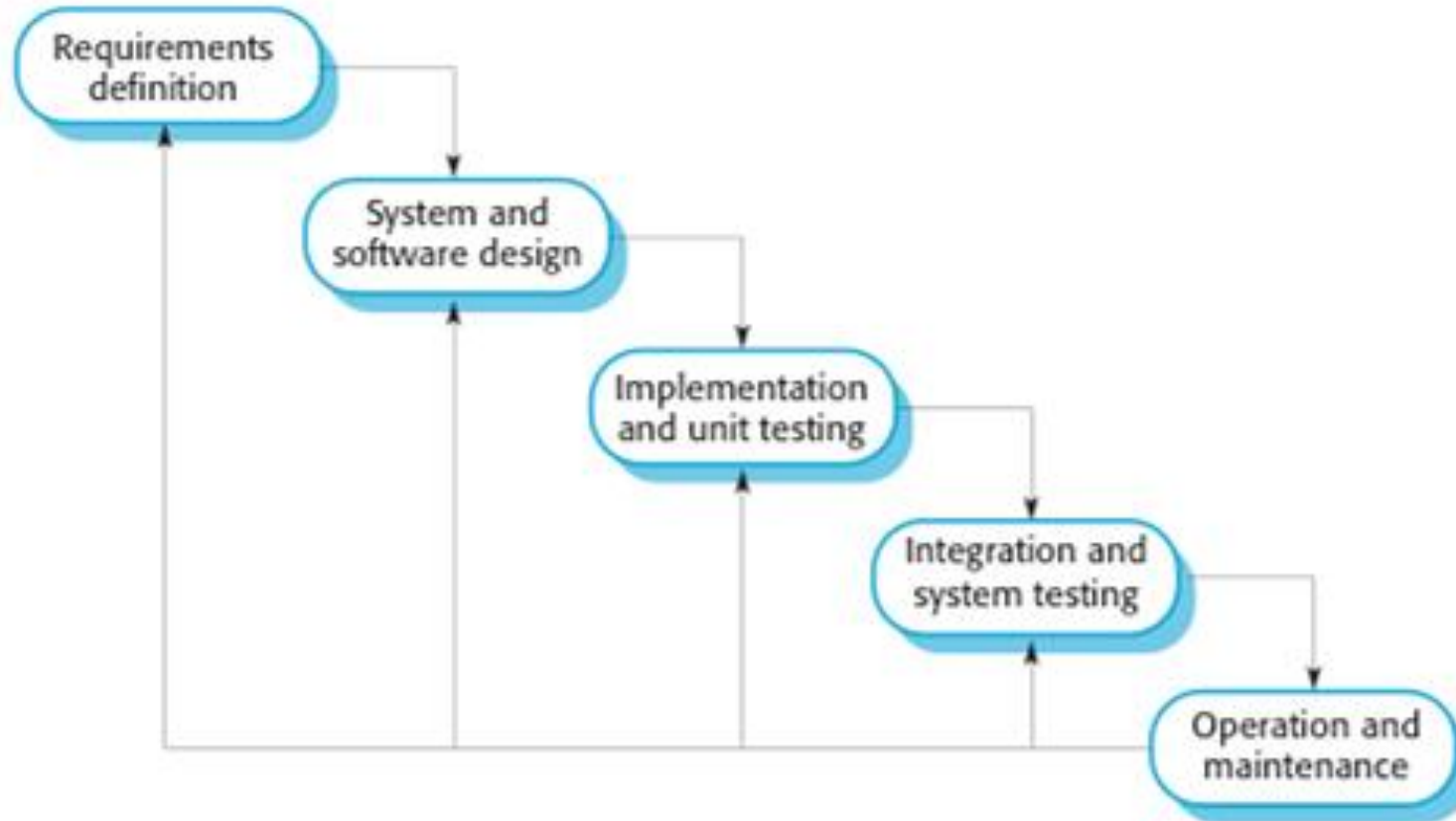
- Pročtěte si 3 z článků přidaných ostatními a přidejte comment v DF
- Vyzkoušejte si [Quiz: Which IT career pathway is right for you?](#)

# METODY SOFTWAREOVÉHO VÝVOJE

# Modely softwarových procesů



# Vodopádový model

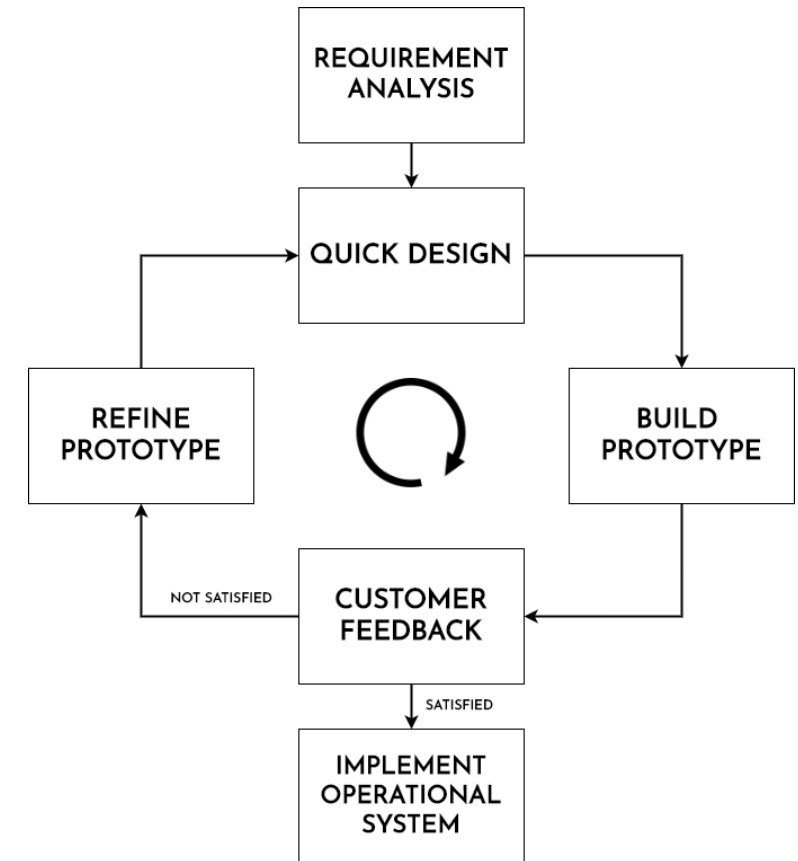


# Výhody a problémy vodopádového modelu

- Vodopádový model se většinou používá u **velkých projektů systémového inženýrství**, kde se systém vyvíjí na několika místech.
  - Za těchto okolností pomáhá koordinovat práci plánovitý charakter vodopádového modelu.
- Vhodné pro nové verze **generických produktů**.
  - Dobře pochopený kontext, stabilní požadavky.
- Tento proces ztěžuje reakci na **měníící se požadavky zákazníků**.
  - Proto je tento model vhodný pouze tehdy, když jsou požadavky dobře pochopeny a změny lze omezit.

# Vytváření prototypů softwaru

- Prototyp je jednoduchá implementace navrhovaného řešení, která slouží k testování nebo ověřování nápadů.
- Poskytuje možnost shromažďovat zpětnou vazbu od zákazníků
- Pomáhá identifikovat problémy a oblasti pro zlepšení
- Prototyp je vždy vyřazen - skutečný vývoj produktu je na základě znalostí získaných při realizaci prototypu.

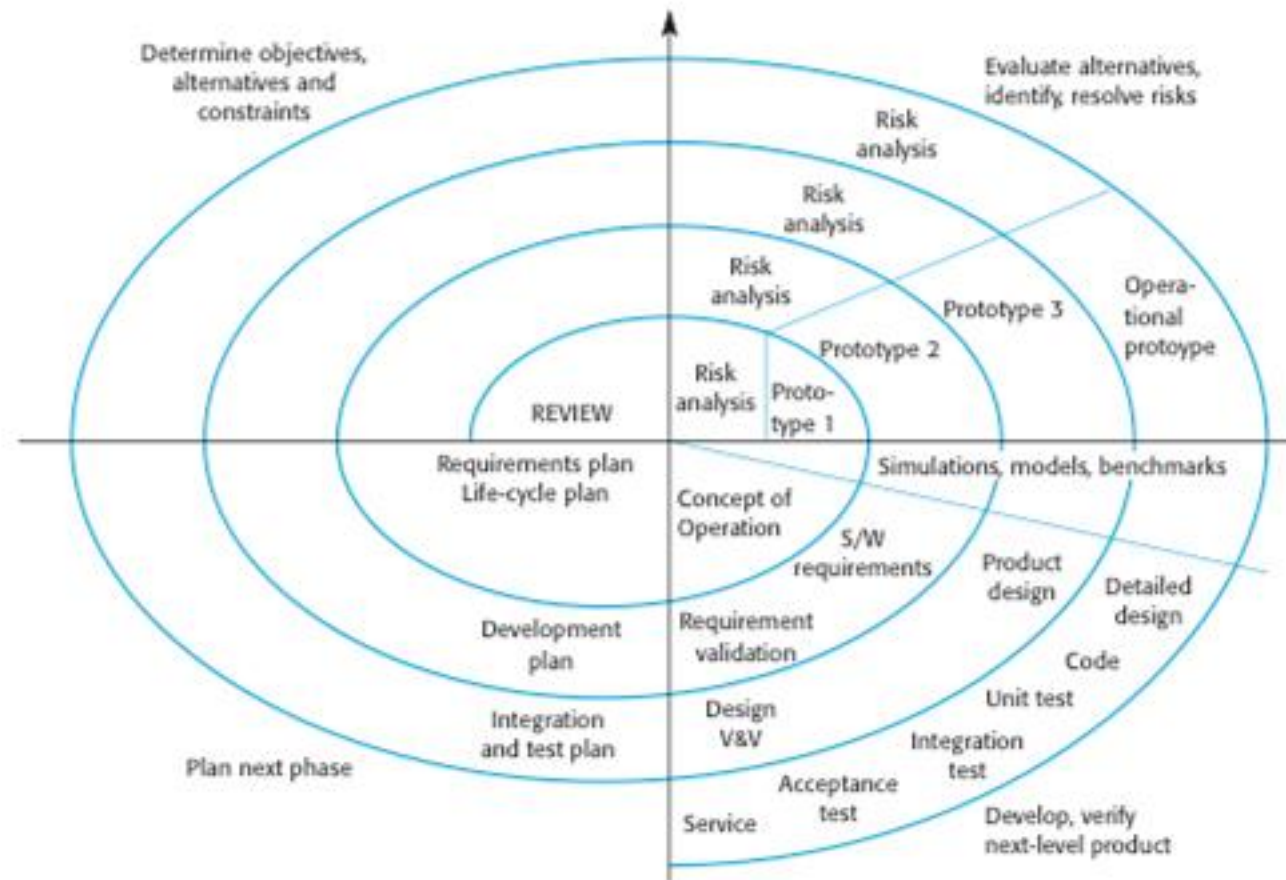


# Boehmův spirálový model

- **Proces** je reprezentován spíše jako **spirála** než jako posloupnost činností se zpětným sledováním.
- Každá smyčka ve spirále představuje jednu fázi procesu.
- **Žádné pevně stanovené fáze**, jako je specifikace nebo návrh - smyčky ve spirále se volí podle toho, co je požadováno.
- **Rizika** jsou v průběhu celého procesu jednoznačně vyhodnocována a řešena.

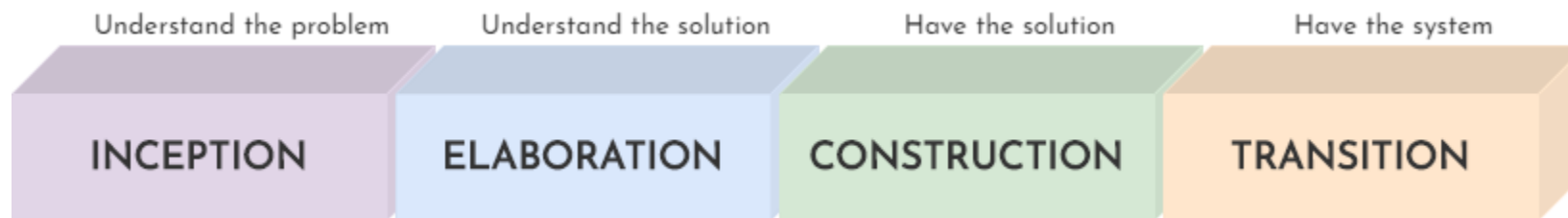


# Boehmův spirálový model softwarového procesu

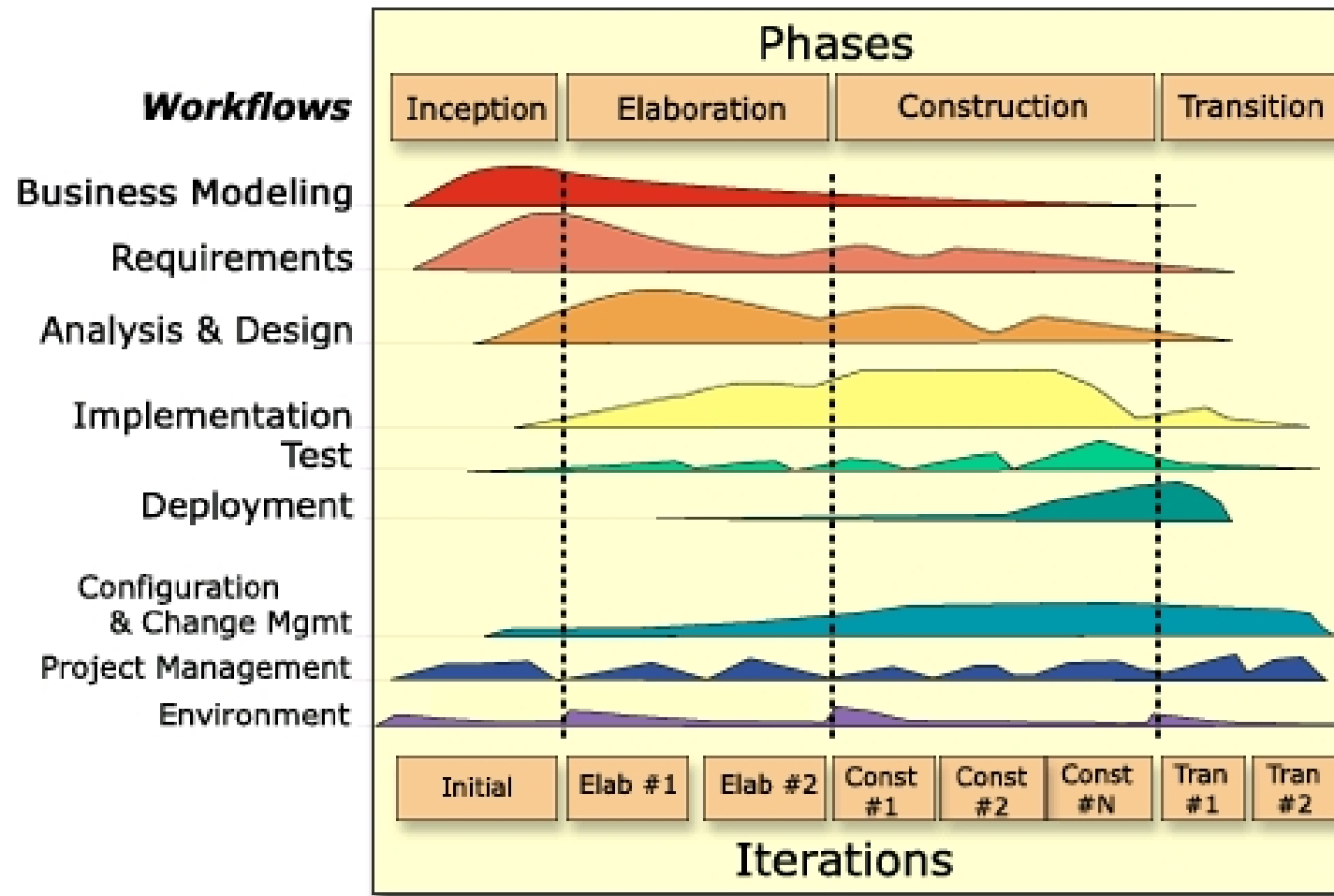


# Rational Unified Process (RUP)

- Moderní obecný proces běžně spojovaný s unifikovaným modelovacím jazykem (UML).
- Obvykle se popisuje ze 3 hledisek:
  - Dynamická perspektiva, která zobrazuje fáze v čase
  - Statická perspektiva, která zobrazuje činnosti procesu
  - Pohled na praxi, který navrhuje správné postupy, jež by se měly používat během procesu.

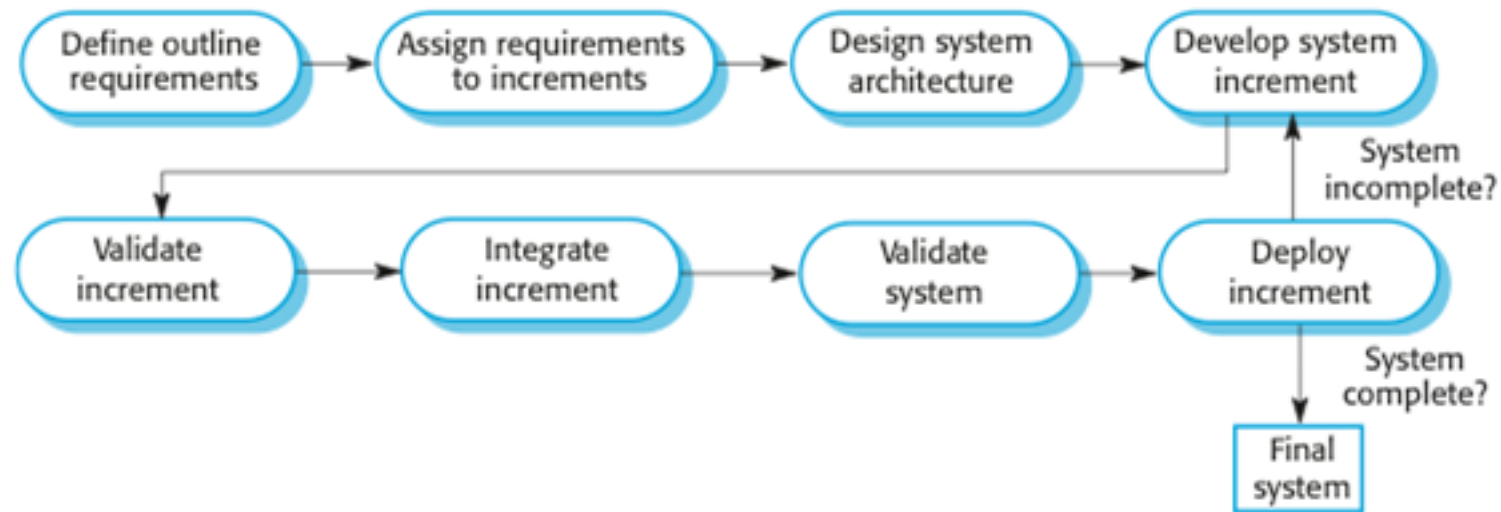


# Rational Unified Process (RUP)



# Iterativní a inkrementální vývoj

- Jaký je mezi nimi rozdíl?



# Inkrementální vývoj

- Namísto jednorázového dodání systému je vývoj a nasazení rozděleno do jednotlivých částí, přičemž **každá část dodává část požadované funkce**.
- Požadavky uživatelů jsou **seřazeny podle důležitosti** a požadavky s nejvyšší prioritou jsou zahrnuty do prvních přírůstků.
- Po zahájení vývoje přírůstku jsou požadavky zmrazeny, ačkoli požadavky na pozdější přírůstky se mohou dále vyvíjet.

# Výhody inkrementálního vývoje

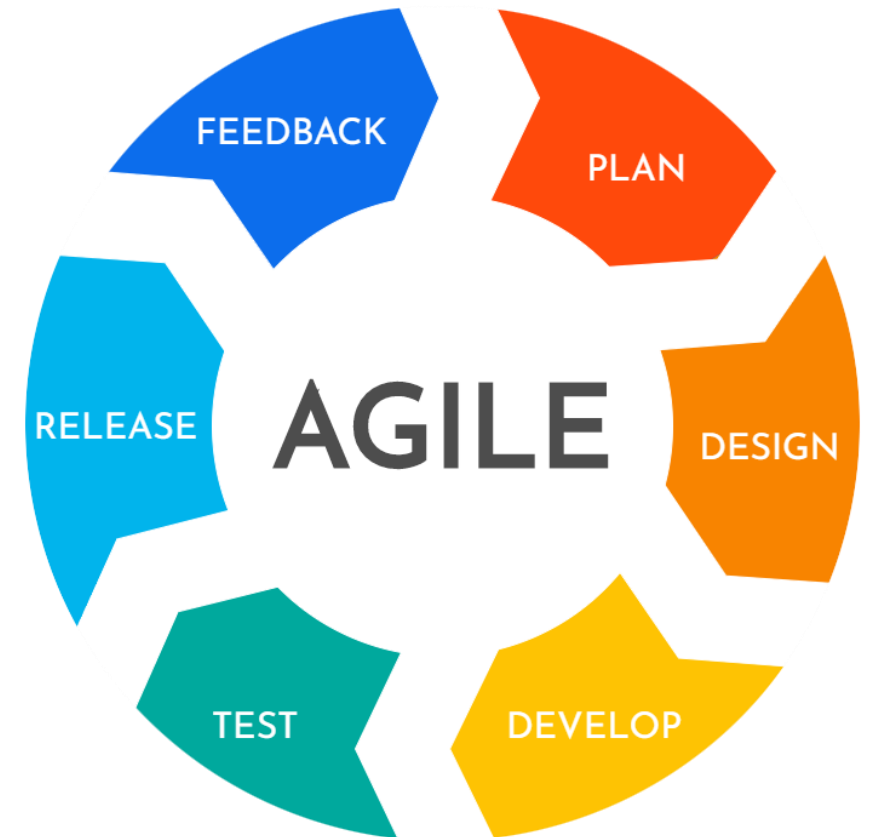
- S každým přírůstkem může být dodána **hodnota pro zákazníka**, takže funkce systému jsou k dispozici dříve.
- Rané přírůstky slouží jako **prototyp**, který pomáhá zjistit požadavky na pozdější přírůstky.
- **Snížení rizika** celkového selhání projektu.
- Systémovým službám s **nejvyšší prioritou** bývá věnována největší pozornost (návrh, testování atd.).

# Problémy s inkrementálním vývojem

- **Úplnou specifikaci** je těžké předvídat.
  - To se stává problematickým, pokud je při vyjednávání o smlouvě vyžadována úplná specifikace.
- **Struktura systému má tendenci se s přidáváním nových přírůstků zhoršovat.**
  - Pokud se nevynaloží čas a peníze na rozsáhlý **refaktoring**, pravidelné změny mají tendenci **narušovat strukturu systému** a zvyšovat náklady na začlenění dalších změn.
- Je obtížné identifikovat a efektivně navrhnout základní **zařízení sdílená** různými částmi systému.
- Proces není viditelný, **pokrok je těžké vysledovat.**

# Agilní metodiky

- Časově omezený, iterativní a inkrementální přístup k vývoji software.
- Umožňuje týmům rychleji dodávat a reagovat na změny.
- Nejpoužívanější agilní metodiky:
  - SCRUM
  - KANBAN
  - Extrémní programování (XP)
  - Vývoj řízený funkcemi (FDD)



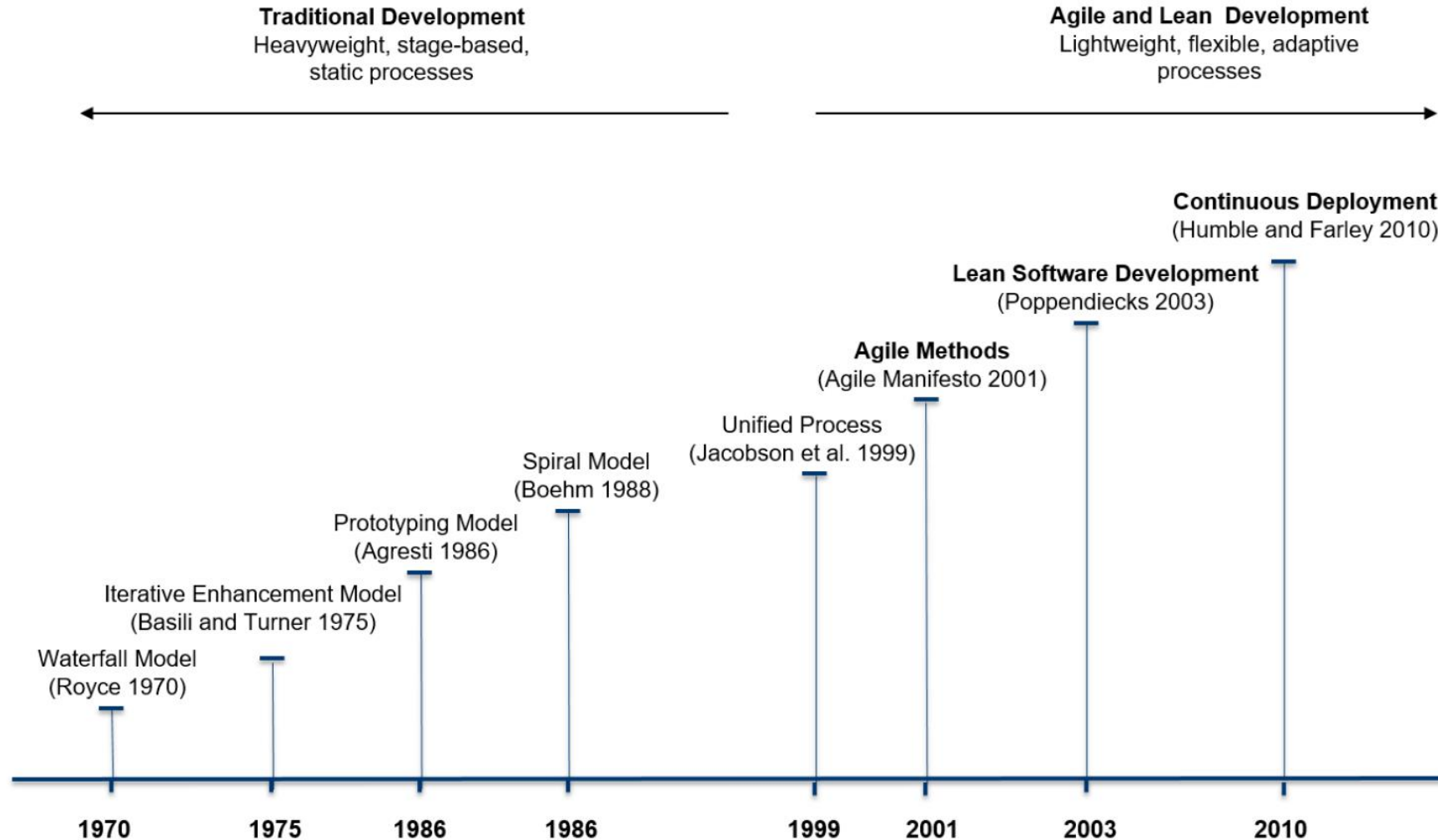


# Skládání systémů z existujících částí

- Na základě **systematického opětovného použití**, kdy jsou systémy integrovány ze stávajících komponent nebo systémů COTS (Commercial-off-the-shelf).
  - Fáze procesu
  - Analýza složek;
  - Úprava požadavků;
  - Návrh systému s opakovaným použitím;
  - Vývoj a integrace.
- Opakované použití je dnes standardním přístupem k budování mnoha typů podnikových systémů.

# AGILNÍ VÝVOJ

# Historie agilního přístupu



# Agilní manifest

## The Agile Manifesto

<b>Individuals and Interactions</b>	over	Processes and Tools
<b>Working Product</b>	over	Comprehensive Documentation
<b>Customer Collaboration</b>	over	Contract Negotiation
<b>Responding to Change</b>	over	Following a Plan

*That is, while there is value in the items on the right,  
we value the items on the left more.*

[www.agilemanifesto.org](http://www.agilemanifesto.org)

# Principy agilních metod

Princip	Popis
<b>Zapojení zákazníků</b>	Zákazníci by měli být úzce zapojeni do celého procesu vývoje. Jejich úlohou je poskytovat nové požadavky a určovat jejich priority a vyhodnocovat iterace systému.
<b>Přírůstkové dodávky</b>	Software je vyvíjen v přírůstcích, přičemž zákazník určí požadavky, které mají být v každém přírůstku zahrnuty.
<b>Lidé, ne proces</b>	Schopnosti vývojového týmu by měly být rozpoznány a využity. Členové týmu by měli mít možnost rozvíjet své vlastní způsoby práce bez předepsaných postupů.
<b>Přijměte změnu</b>	Očekávejte, že se požadavky na systém budou měnit, a navrhnete systém tak, aby se těmto změnám přizpůsobil.
<b>Zachování jednoduchosti</b>	Zaměřte se na jednoduchost vyvíjeného softwaru i procesu vývoje. Kdekoli je to možné, aktivně pracujte na odstranění složitosti systému.

# Agilní

- Být agilní znamená **reagovat na změny**.
- Podporuje **týmovou práci, sebeorganizaci a odpovědnost**.
- Hlavní aspekty:
  - Flexibilita
  - Rozdělení prací
  - Hodnota týmové práce
  - Iterativní vylepšení
  - Spolupráce s klientem

# Výhody agilního vývoje

- **Spokojenost zákazníků** díky průběžnému dodávání softwaru
- Pracovní software je **dodáván často**
- Větší **flexibilita a přizpůsobivost** změnám
- Zvýšená frekvence **spolupráce** a zpětná vazba
- Úzká **spolupráce** mezi zúčastněnými stranami a vývojáři
- Zaměření na **obchodní hodnotu**
- Zvýšená **kontrola** projektu

# Problémy a výzvy v agilním přístupu

- Projekt může snadno sejít z cesty, pokud zúčastněné strany **nemají jasno** v tom, jaký konečný **výsledek** chtějí.
- Může být obtížné **udržet zájem zákazníků**, kteří jsou do procesu zapojeni.
- Úroveň **spolupráce** může být **obtížné** udržet
- Riziko **ztráty dlouhodobé vize**, protože není jasný konec projektu.
- Problémem mohou být **smlouvy**, stejně jako u jiných přístupů k iterativnímu vývoji.

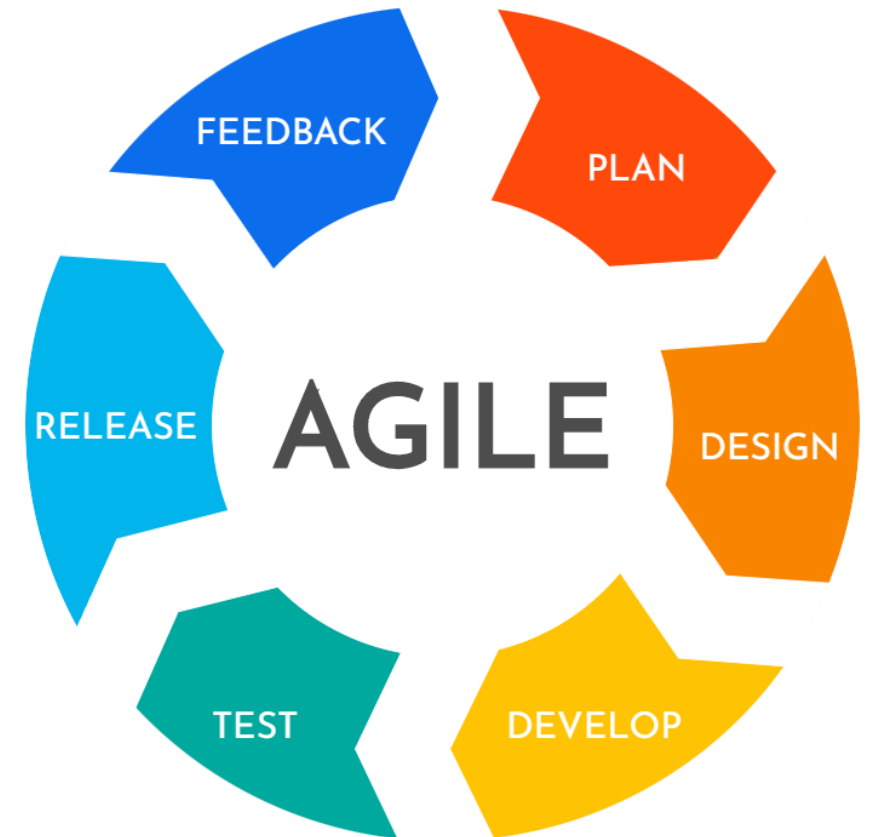


# Problémy a výzvy v agilním přístupu

- **Dokumentace** má tendenci **odbíhat na vedlejší kolej**
- **Obtížné měření pokroku**
- Vzhledem k tomu, že se **agilní metody** zaměřují na malé, úzce integrované týmy, je třeba být opatrný při **jejich rozšiřování** na velké systémy.
- **Stanovení priorit změn** může být obtížné, pokud existuje **více zúčastněných stran**.
- Zachování **jednoduchosti** vyžaduje další práci

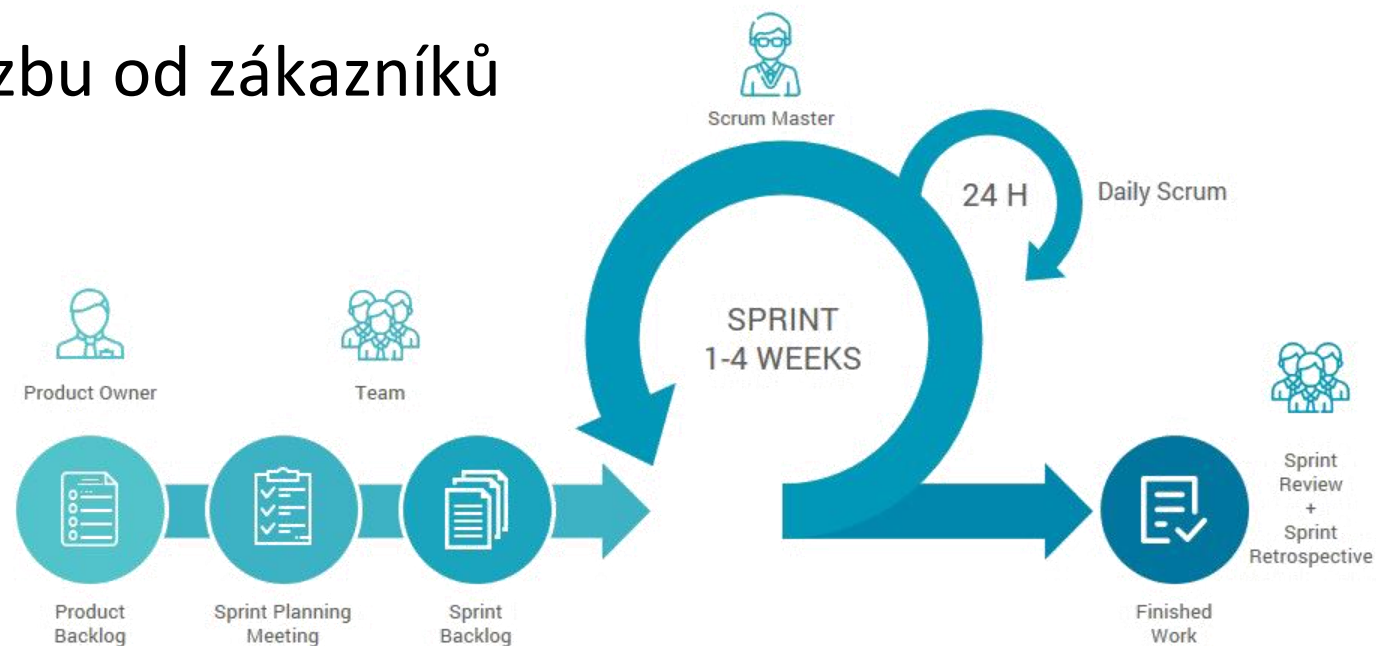
# Agilní metodiky

- Velká část vlastností je společná
  - Iterativní proces návrhu
  - Efektivní komunikace a zapojení zúčastněných stran
  - Snaha o kvalitní a spolehlivý software
  - Krátký vývojový cyklus umožňující pravidelné dodávky softwaru
- Nejpoužívanější agilní metodiky:
  - SCRUM
  - KANBAN
  - Extrémní programování (XP)
  - Vývoj řízený funkcemi (FDD)



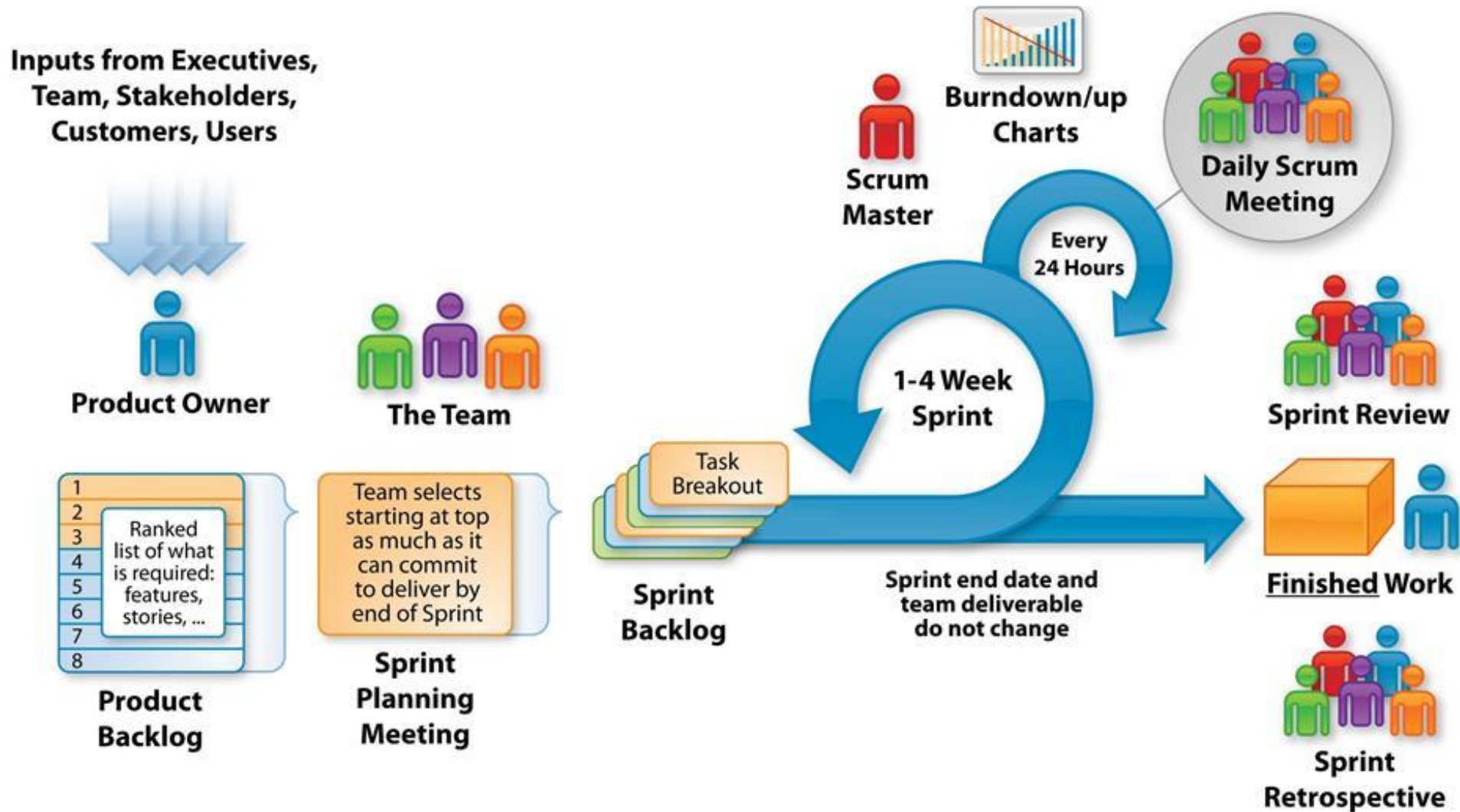
# Agilní metodika Scrum

- Soubor schůzek, nástrojů, postupů a rolí, které pomáhají týmům strukturovat a řídit jejich práci.
- Týmy dodávají produkty v iteracích zvaných sprinty.
- Průběžně vytvářet části funkcí s nejvyšší prioritou a pravidelně získávat zpětnou vazbu od zákazníků



<https://media.vpl.us/images/katanazero86/post/dcab8baba-9834-4276-ae85-870753abb0dd/%EB%8B%A4%EC%69A%B4%EB%A1%9C%EB%93%9C.jpg?w=768>

# The Agile - Scrum Framework



# Scrum ceremonie

- Sprint
  - Základní jednotka vývoje ve Scrumu, pevně stanovená doba trvání 1-4 týdny.
- Plánování sprintu
  - Projedná se a odsouhlasí rozsah práce, která má být během sprintu provedena.
- Denní Scrum
  - Každý den během sprintu tým pořádá denní scrum (nebo stand-up), aby všichni řekli, co včera dokončili, co plánují dokončit dnes a s jakými překážkami se potýkají.
- Recenze Sprint
  - Tým přezkoumá dokončenou práci a naplánuje práci, která nebyla dokončena.
- Retrospektiva sprintu
  - Tým se zamyslí nad uplynulým sprintem a určí a odsouhlasí opatření pro neustálé zlepšování procesů.

# Denní scrum (Stand-up)

- 15minutová časově omezená událost pro vývojový tým k synchronizaci činností a vytvoření plánu na příštích 24 hodin.
- Optimalizuje týmovou spolupráci a výkonnost sdílením práce provedené od posledního denního scrumu a předvídáním práce v nadcházejícím sprintu.
- Denní schůzka se koná každý den ve stejnou dobu a na stejném místě.
- Na tyto otázky by měl odpovědět každý člen týmu:
  - Na čem jsem včera pracoval?
  - Na čem dnes pracuji?
  - Jaké problémy mě blokují?

# Projektové role

## – Vlastník produktu

- Odborník na produkt a potřeby a priority zákazníka. Denně spolupracuje s vývojovým týmem, aby pomohl vyjasnit požadavky a učinil obchodní rozhodnutí.

## – Scrum Master

- Týmová role zodpovědná za to, že tým bude dodržovat agilní hodnoty a principy a bude se řídit procesy a postupy, na kterých se tým dohodl.

## – Člen týmu

- Lidé, kteří produkt vytvářejí. Programátoři, testeři, designéři, autoři, datoví inženýři a všichni ostatní, kteří se podílejí na vývoji produktu.

## – Zúčastněné strany

- Každý, kdo má o projekt zájem. Poskytuje pravidelnou zpětnou vazbu a je ovlivněn výsledkem projektu.

# Projektové role



<https://wac-cdn.atlassian.com/dam/jcr:98d24faa-dd1e-4563-b389-85dc58b13b77/Scrum-Master-revised.png?cdnVersion=1027>

MANAGING THE SCRUM BACKLOG



<https://wac-cdn.atlassian.com/dam/jcr:e65c7b56-55e3-47e5-8e6b-ba80afa95a0f/Scrum-Product%20Owner-revised.png?cdnVersion=1027>

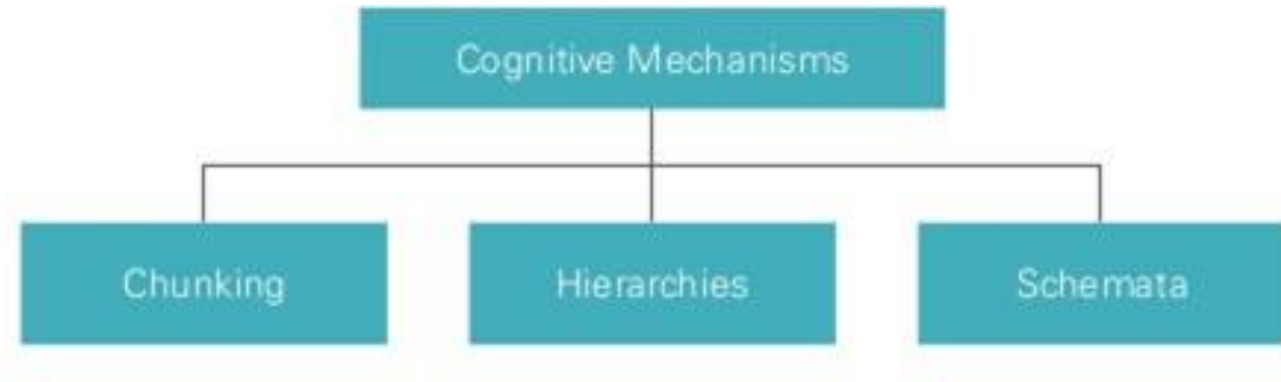


<https://wac-cdn.atlassian.com/dam/jcr:f085fea0-5149-4b9a-9fe1-7e9fd32dc0da/Scrum-Development%20team-revised.png?cdnVersion=1027>



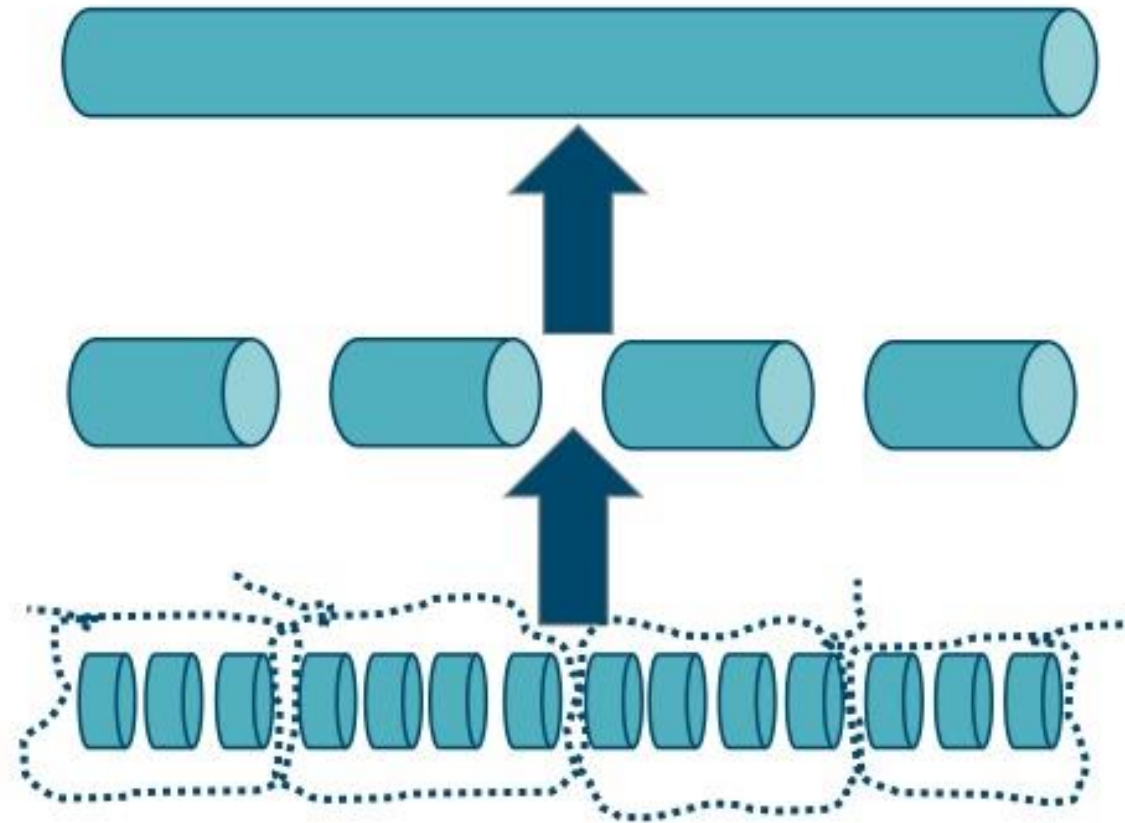
# ÚLOHA MODELŮ V SOFTWAREM INŽENÝRSTVÍ

# Vizualizace usnadňuje porozumění



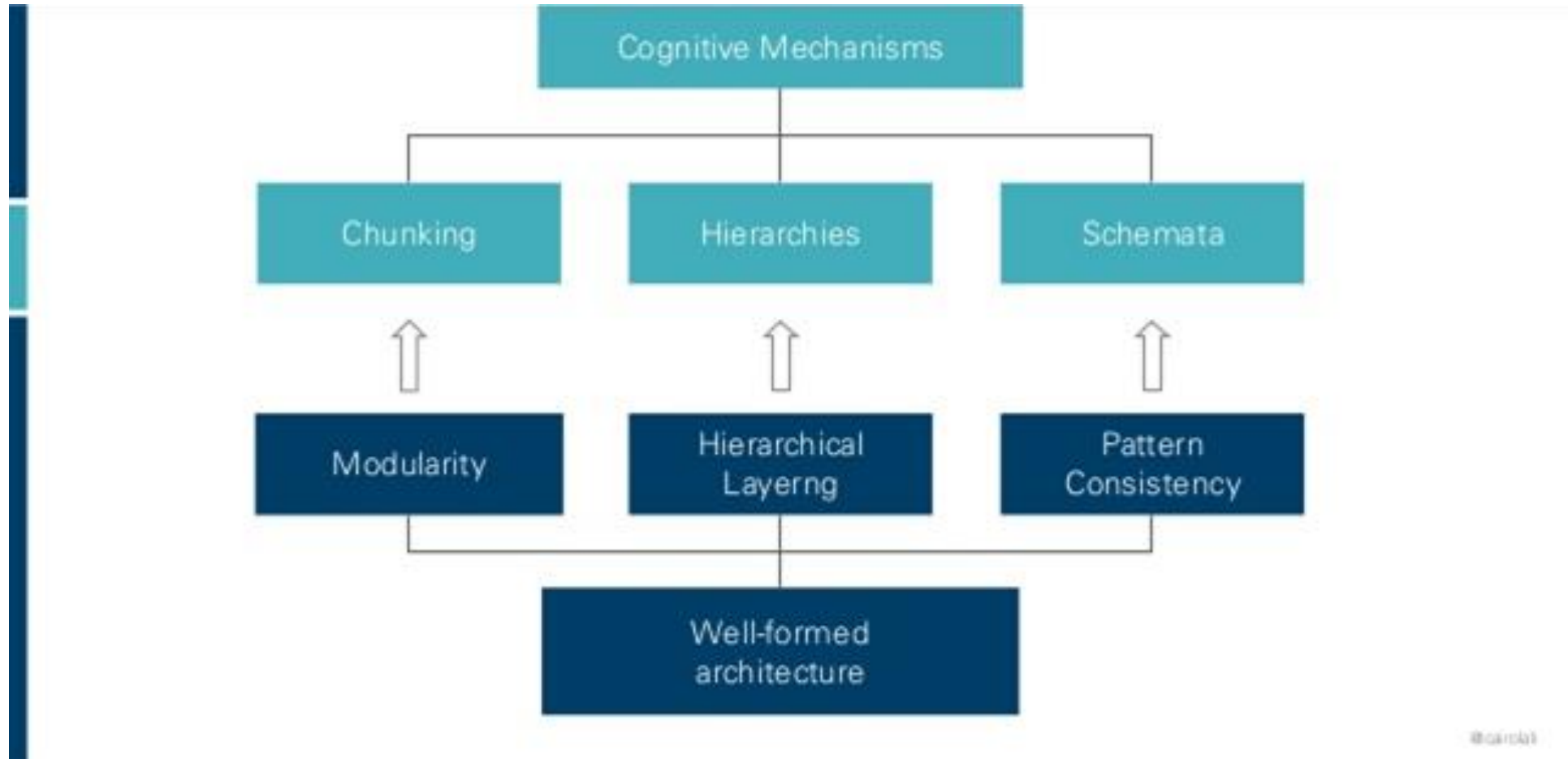
@caiolali

# Chunking



#caiolali

# Vizualizace usnadňuje porozumění



# Základní pohledy na softwarové systémy

- Funkčně orientovaný pohled
  - Systém jako soubor vzájemně se ovlivňujících funkcí. Funkční transformace založené na procesech, propojené s datovými a řídicími toky.
- Datově orientovaný pohled
  - Vyhledává základní datové struktury v systému. Funkční aspekt systému (tj. transformace dat) je méně významný.
- Objektově orientovaný pohled
  - Systém jako soubor vzájemně se ovlivňujících objektů, které obsahují jak data, tak operace prováděné s daty.

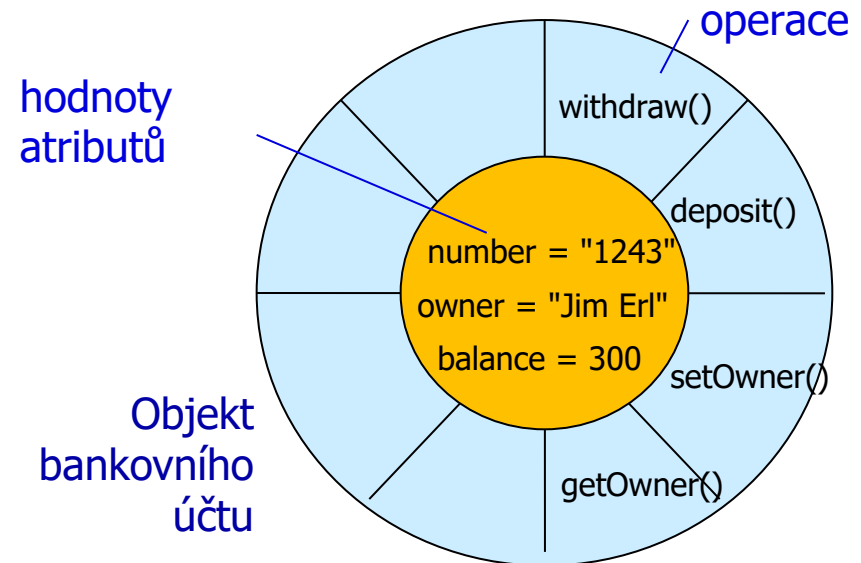
# Objektově orientovaná analýza a návrh

- Přístup softwarového inženýrství, který modeluje systém jako skupinu **vzájemně se ovlivňujících objektů**.
- Každý objekt představuje určitou entitu, která je předmětem zájmu modelovaného systému, a je charakterizován svou **třídou**, **stavem** (datovými prvky) a **chováním**.
- Pro zobrazení statické struktury, dynamického chování a nasazení těchto spolupracujících objektů za běhu lze vytvořit **různé modely**.
- Existuje řada **různých metod**, které určují pořadí modelovacích činností.  
**Modelovací notace** se používá jednotná (UML).

# Objekty a třídy analýzy

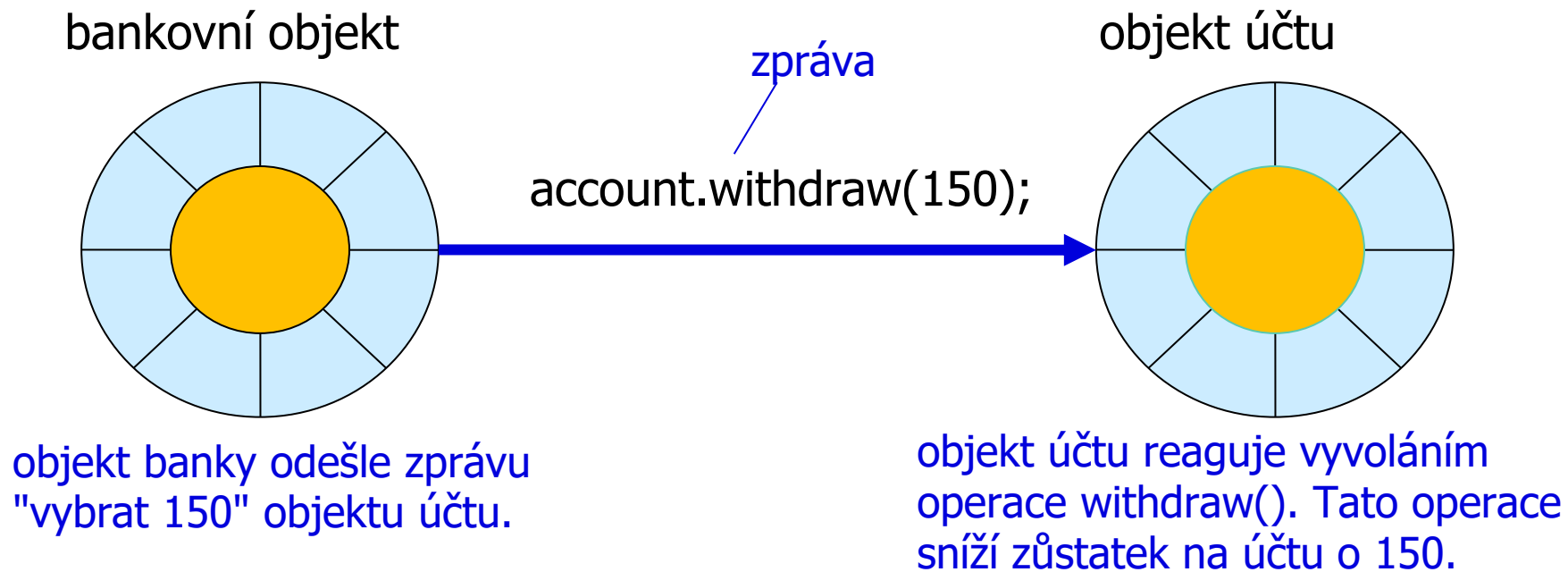
## Co jsou to objekty?

- Objekty se skládají z dat a funkcí zabalených do opakovaně použitelného celku. Objekty **zapouzdřují** data.
- Každý objekt je instancí nějaké **třídy**, která definuje společnou množinu **vlastností** (atributů a operací) sdílených všemi jejími instancemi.
- Objekty mají:
  - **Hodnoty atributů** - datová část
  - **Operace** - část chování



# Zasílání zpráv

- V OO systémech si objekty navzájem posílají zprávy přes odkazy.
- Tyto zprávy způsobí, že objekt vyvolá operaci.





# Co jsou to třídy?

- Každý objekt je instancí jedné třídy - třída popisuje "typ" objektu.
- Třídy umožňují modelovat množiny objektů, které mají stejnou sadu vlastností - třída funguje jako šablona pro objekty:
  - Třída určuje strukturu (soubor vlastností) všech objektů této třídy.
  - Všechny objekty třídy musí mít stejnou sadu operací, musí mít stejné atributy, ale mohou mít různé hodnoty atributů.
- Klasifikace je jedním z nejdůležitějších způsobů, jak uspořádat náš pohled na svět.
- Představte si třídy jako:
  - Razítka
  - Vykrajovátka na sušenky



# Kolik tříd je na obrázku?



# Modelování systému

- Modelování systému je proces vytváření **abstraktních modelů systému**, přičemž každý model představuje jiný pohled nebo **perspektivu** tohoto systému.
- Modelování systému nyní znamená znázornění systému pomocí určitého druhu grafického zápisu, který je velmi často založen na **Unified Modeling Language (UML)**.
- Modelování systému pomáhá analytikovi **pochopit funkčnost** systému a modely se používají při **komunikaci s kolegy a zákazníky**.

# Systemové perspektivy

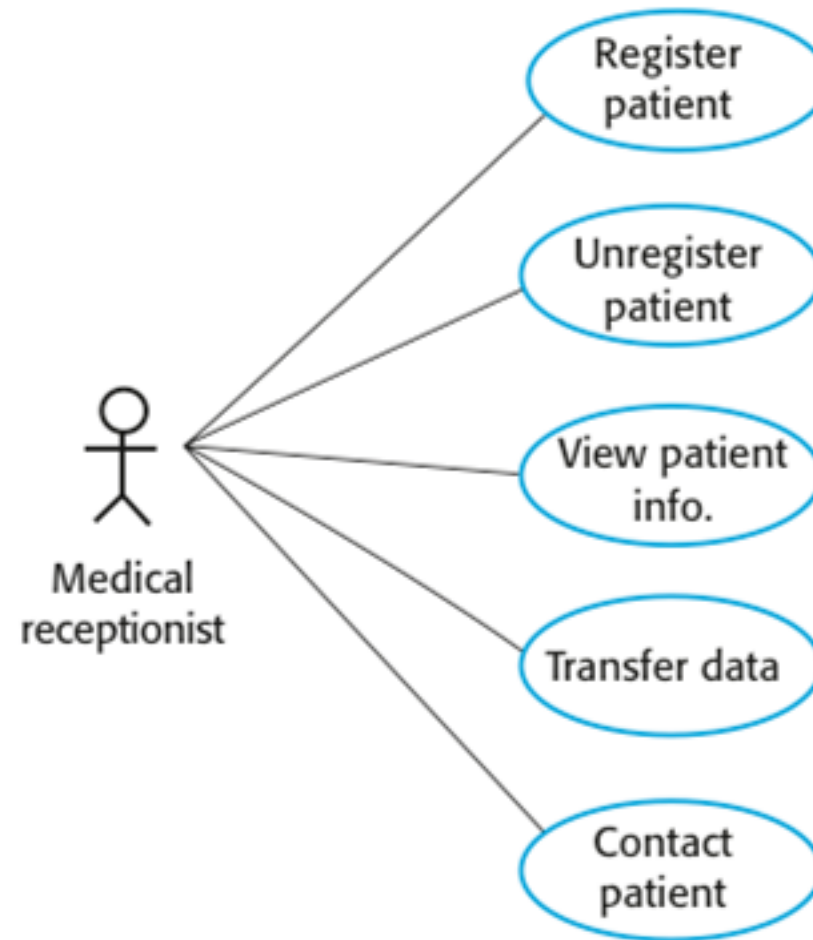
- **Vnější pohled**, kde se modelují hranice systému, kontext a/nebo prostředí systému.
- **Strukturální hledisko**, kde se modeluje organizace systému nebo struktura dat, která jsou systémem zpracovávána.
- **Interakční perspektiva**, při níž se modelují interakce mezi systémem a jeho prostředím nebo mezi součástmi systému.
- **Perspektiva chování**, kdy se modeluje dynamické chování systému nebo jeho jednotlivých prvků a jejich reakce na události.

# Oblíbené diagramy UML

- **Diagramy případů užití**, které zobrazují interakce mezi systémem a jeho prostředím.
- **Diagramy tříd**, které zobrazují třídy objektů a asociace mezi nimi.
- **Sekvenční diagramy**, které zobrazují interakce mezi aktéry a systémem a mezi komponentami systému.
- **Diagramy aktivit**, které znázorňují činnosti zapojené do procesu nebo zpracování dat.

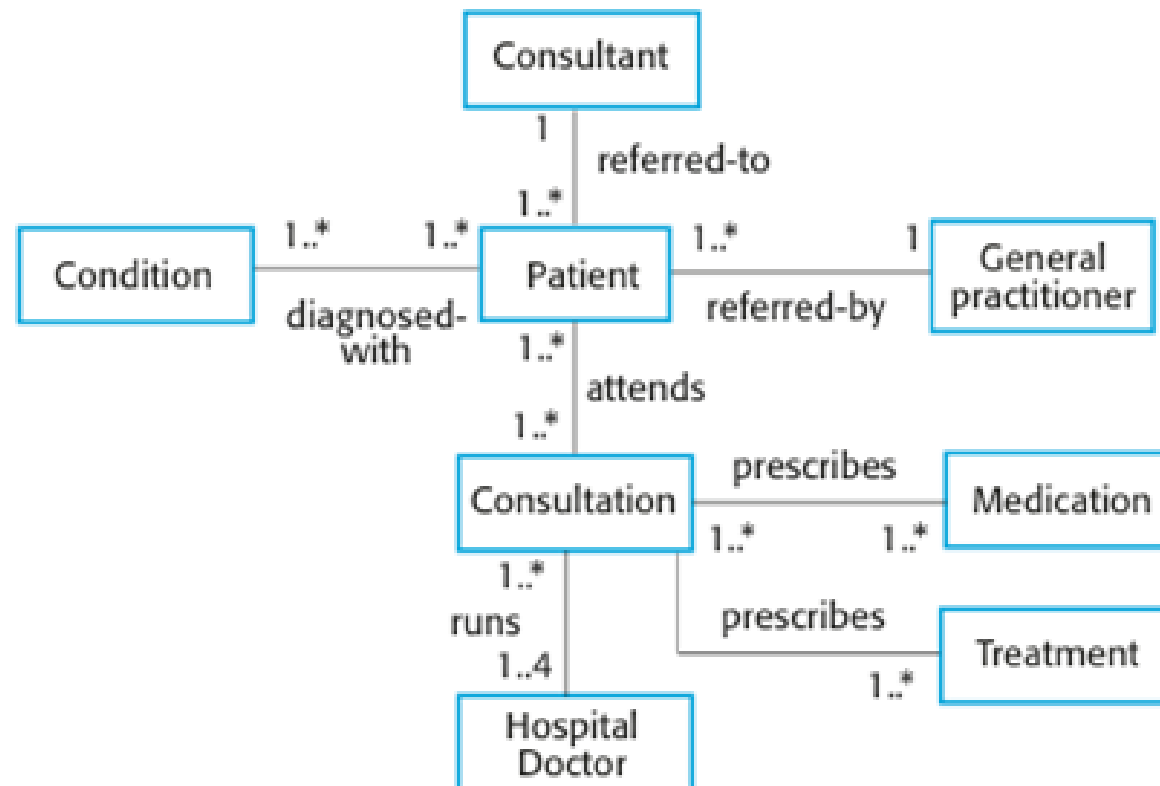
# Oblíbené diagramy UML

- **Diagramy případů užití,** které zobrazují interakce mezi systémem a jeho prostředím.



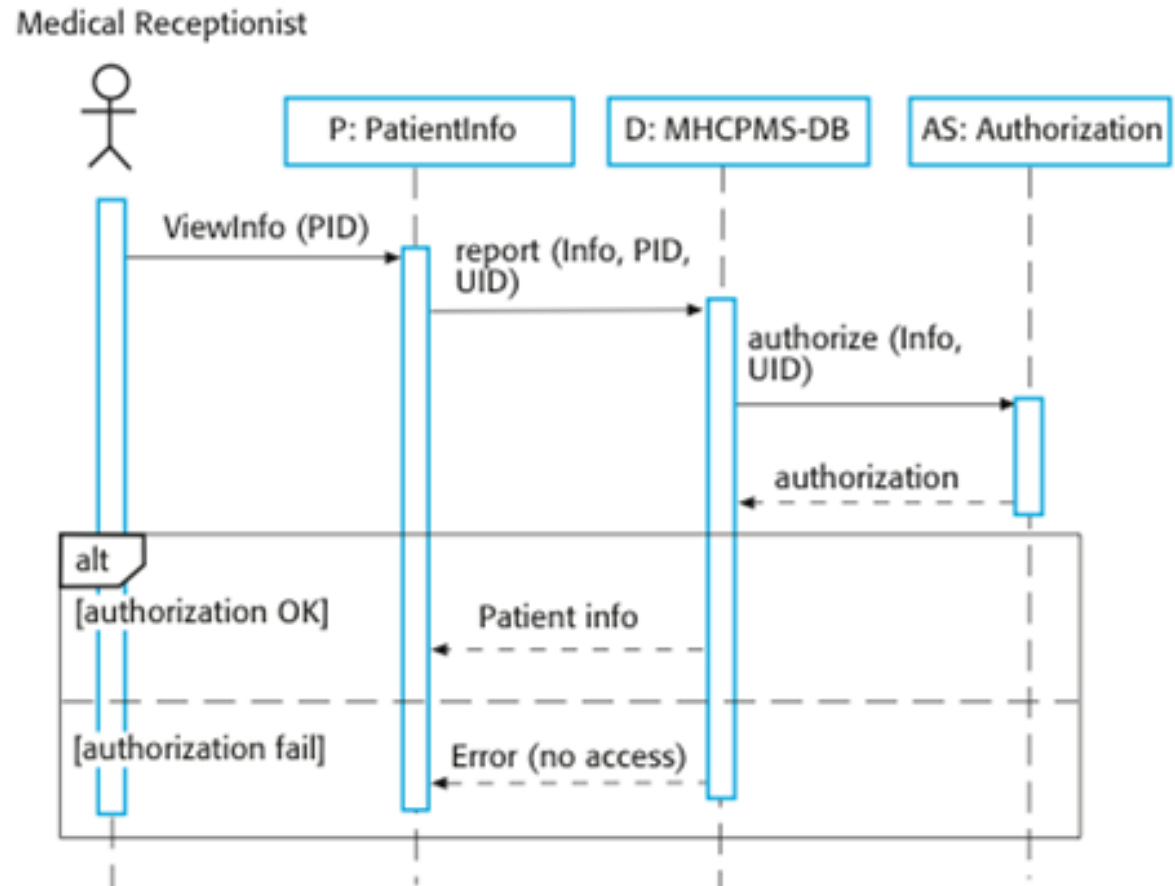
# Oblíbené diagramy UML

- **Diagramy tříd**, které zobrazují třídy objektů a asociace mezi těmito třídami.



# Oblíbené diagramy UML

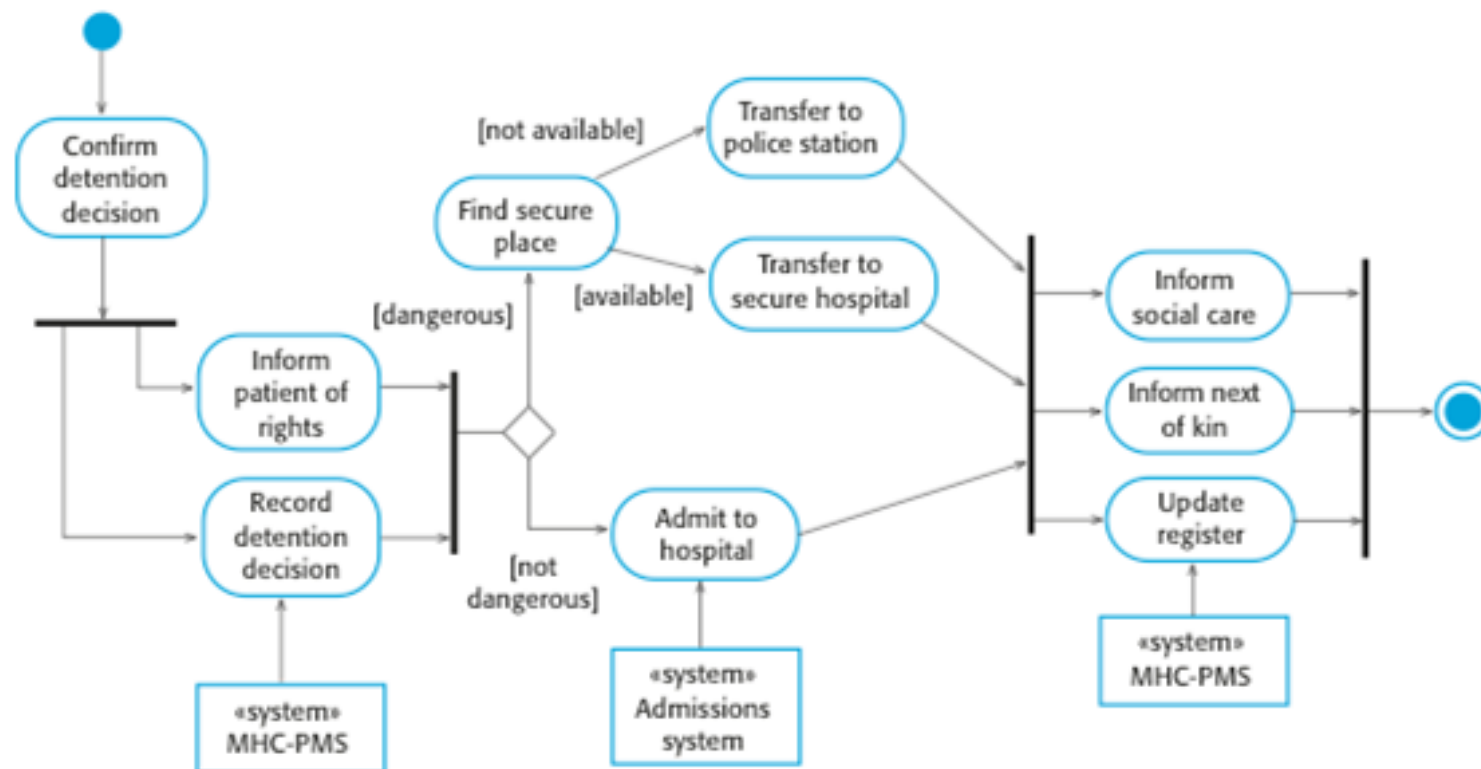
- **Sekvenční diagramy**, které zobrazují interakce mezi aktéry a systémem a mezi komponentami systému.





# Oblíbené diagramy UML

- **Diagramy aktivit**, které znázorňují činnosti zapojené do procesu nebo zpracování dat.



# Oblíbené diagramy UML

- **Diagramy případů užití**, které zobrazují interakce mezi systémem a jeho prostředím.
- **Diagramy tříd**, které zobrazují třídy objektů a asociace mezi nimi.
- **Sekvenční diagramy**, které zobrazují interakce mezi aktéry a systémem a mezi komponentami systému.
- **Diagramy aktivit**, které znázorňují činnosti zapojené do procesu nebo zpracování dat.

# CO NÁS ČEKÁ PŘÍŠTĚ

## 4. Algoritmické myšlení

- Algoritmické myšlení
- Programování jako koncept
- Jak vypadá práce programátora
- Programování prakticky

### **Domácí práce a příprava na příští přednášku**

- Proklikajte si [portál iMyšlení](#)
- Proklikajte si [portál CS Unplugged](#)