… je inženýrská (technická) disciplína zabývající se praktickými problémy spojenými s vývojem rozsáhlých SW systémů od počátečních fází specifikace systému po jeho údržbu.

Softwarové inženýrství

Definice (Fritz Bauer, 1968):

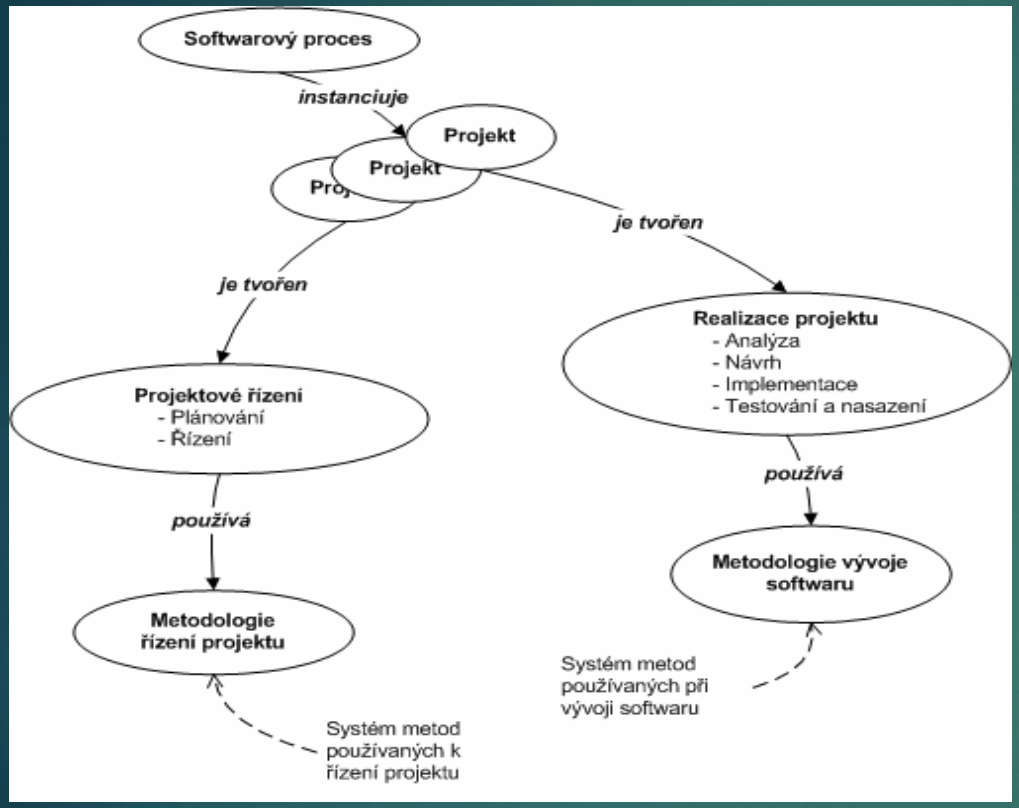
Softwarové inženýrství je zavedení a používání řádných inženýrských principů tak, abychom dosáhli ekonomické tvorby softwaru, který je spolehlivý a pracuje účinně na dostupných výpočetních prostředcích.

Ekonomická tvorba SW:

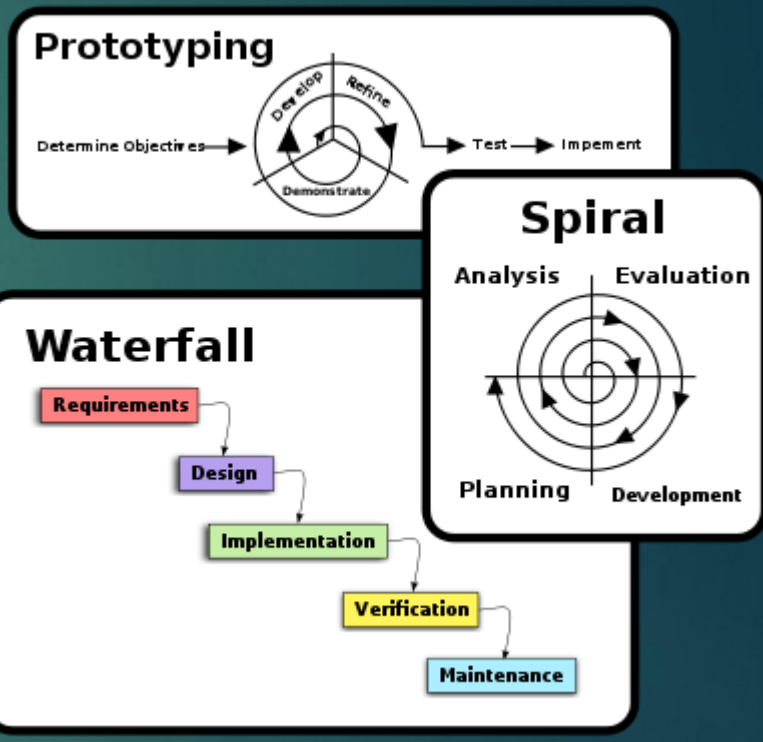
Zahrnuje vhodné sestavení vývojového týmu, vhodnou volbu správného vývojového nástroje, rozvahu, zda „vyvinout nebo koupit“ knihovnu, komponentu, je nutné nalézt společnou řeč se zadavatelem, zvážit budoucí údržbu a rozšiřování programu.

- Vznik v 70. létech, první aplikace umožňující interakci s uživatelem, produkty šité na míru, první chyby, nedokončené projekty,…
- 80. léta masivní rozvoj softwarového inženýrství, silný nástup metodik, OO přístupu, standardizace, rozvoj produktových řad, komponent, architektur a modelů
- 1997 je softwarové inženýrství uznáno jako **obor s certifikátem** v USA.

Vývoj SW produktu obecně



Tři standardní modely



https://www.softwareengineerinsider.com/articles/what-is-software-engineering.html
Co je SW inženýrství?

https://www.careerexplorer.com/careers/software-engineer/
Kdo je SW inženýr?

Životní cyklus softwarového díla

- Začíná prvotním nápadem něco řešit nebo vylepšit pomocí IS/IT v souladu s informační strategií podniku, končí likvidací produktu a výměnou za jiný.

- ŽC SW obsahuje zhruba tyto etapy:
  - **Specifikace problému**
  - **Analýza**
  - **Návrh**
  - **Implementace**
  - **Zavedení a testování produktu, dokumentace**
  - **Provoz, údržba a rozvoj produktu**

Náročnost fází životního cyklu

- Každý produkt by měl projít všemi fázemi životního cyklu.
- Ze statistik (pro velké systémy – řádově stovky tisíc řádků kódu) vyplývá, že úsilí věnované těmto fázím by mělo být rozděleno zhruba v poměru:

– analýza 40%

– návrh 40%

– implementace 20%

- Pokud se tedy nevěnuje počátečním fázím dostatečné úsilí, projeví se to zvýšenými nároky při implementaci či údržbě systému.

Modely ŽC

Modely určují základní schéma postupu při tvorbě SW a zvolená podoba modelu je závislá na osobnosti manažera projektu a charakteru celého týmu. Typové modely jsou jádrem metodik a určují posloupnosti aktivit v metodikách.

- Model **vodopád**, Model **spirálový**, Model **prototyp**, Model **iterativní**.

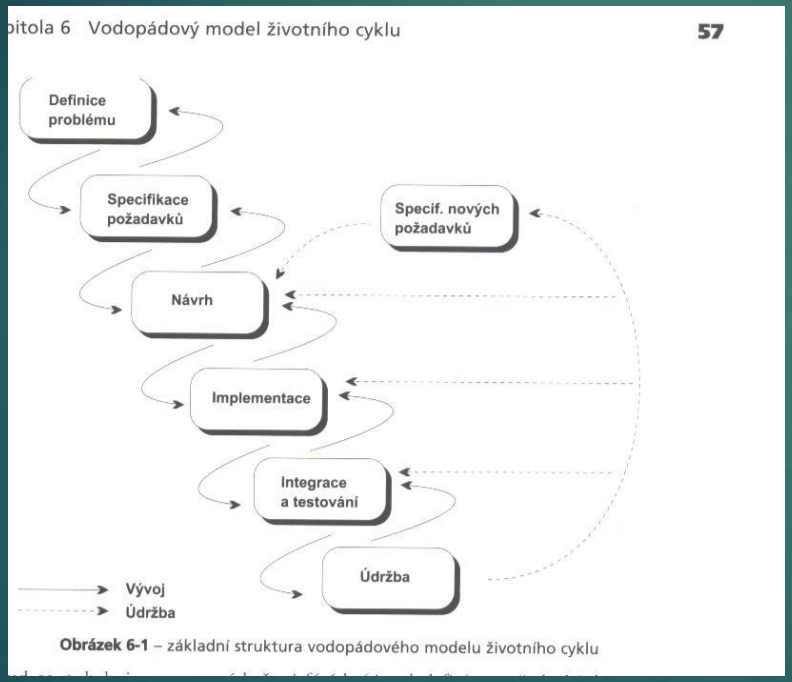
- **Vodopádový model** životního cyklu (princip nikdy se nevracet zpět), pro rozsáhlé projekty je nevhodný, přinesl však základní členění softwarového procesu a jejich logickou posloupnost, ve své době převratný, dnes má stále zvuk.

Všechny podstatné fáze jsou prováděny ve stanoveném pořadí s žádnými nebo minimálními iteracemi.

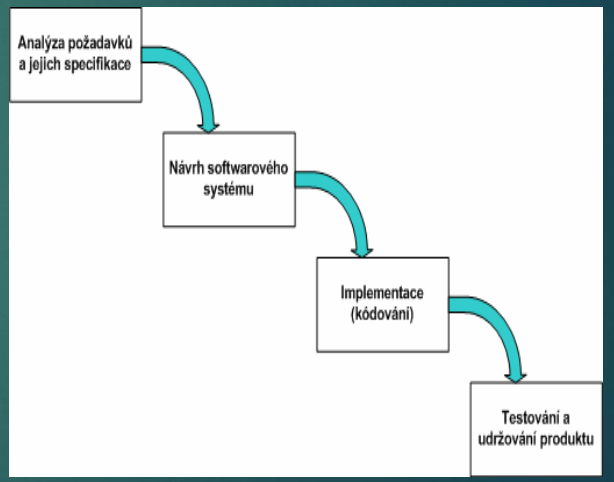
Viz obr.

Vodopádový model životního cyklu

- Novější zjednodušený vodopád (Vondrák)



Obrázek 6-1 – základní struktura vodopádového modelu životního cyklu



Fáze vodopádového životního cyklu

- **Definice problému** – (výsledkem je dokument Úvodní studie , tj. hrubý, základní, přibližný popis požadavků na systém)
- **Analýza a specifikace požadavků** – (výsledkem je dokument Specifikace požadavků, existuje standard pro specifikaci požadavků IEEE 830-1998, dokument je odsouhlasen a podepsán zákazníkem)
- **Návrh a tvorba architektury** – obsahuje tyto podfáze
  - Určení implementačního prostředí, vývojového nástroje, programovacího jazyka, technologií
  - Vytvoření architektury systému, logické rozdělení na subsystémy a další funkční celky
  - Namapování logického návrhu do fyzické struktury
  - Definice chování modulů, specifikace práce s daty
- **Implementace** (naprogramování návrhu, realizace jako SaaS, …)
- **Integrace a testování**
- **Provoz a údržba a rozvoj** (nekončící proces úprav, oprav, vylepšování a ladění aplikace).

Spirálový model životního cyklu

Barry Boehm (1985)

- Do procesu vývoje zavádí klíčové koncepty, iterativní přístup a opakovanou a důslednou analýzu rizik.
- Model náleží do skupiny **riziky řízených přístupů** (v závislosti na výsledcích rizikových analýz se rozhoduje o dalším směru vývoje, na konkrétním postupu).
- Spirála ukazuje jakými kroky vyvíjený produkt prochází, velikost spirály (radiální rozměr) ukazuje časové a finanční náklady.
- Základní pojmy jsou rizika, prototyp, plánování a cyklický iterativní vývoj.

Spirálový model životního cyklu

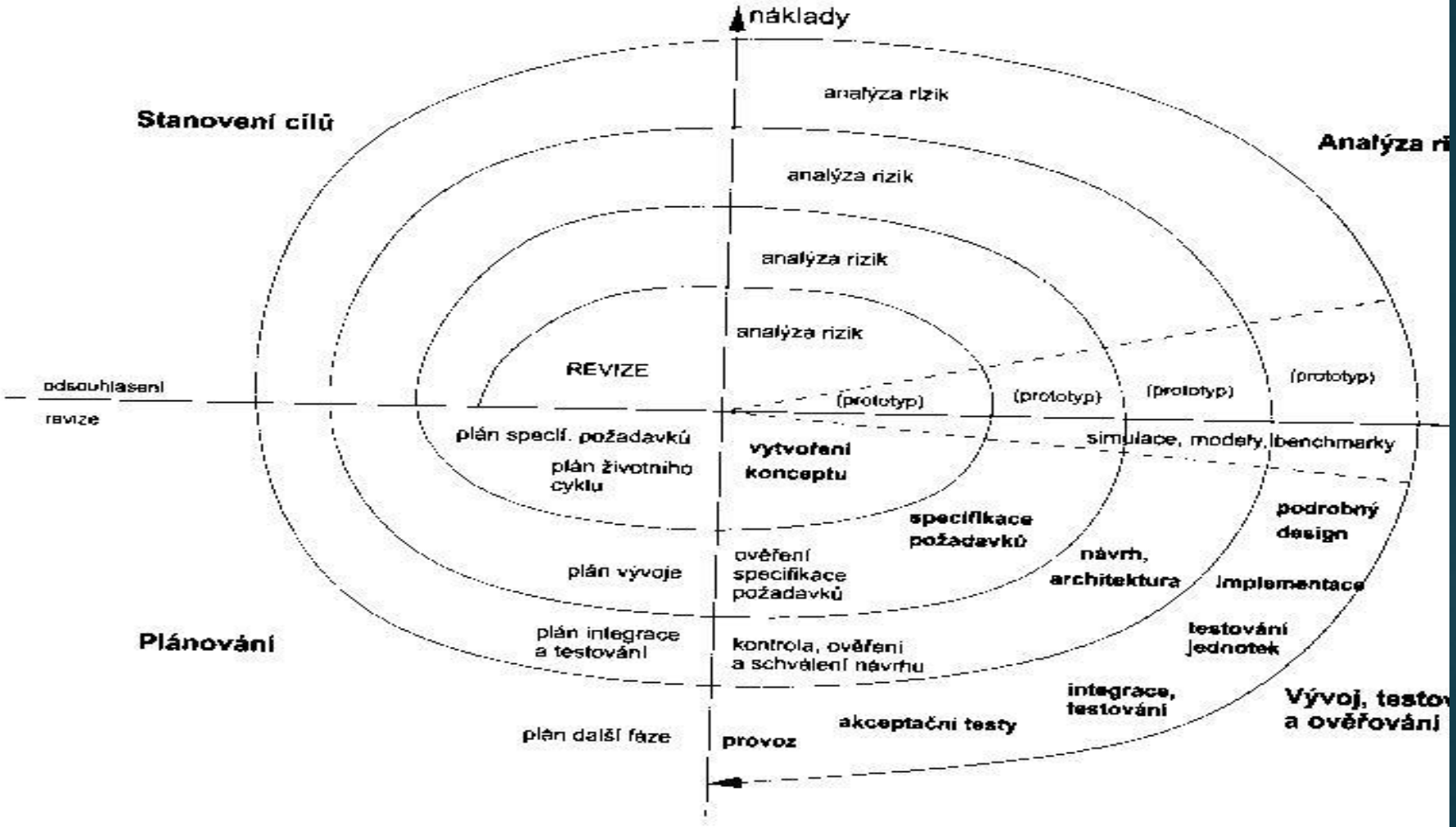
- Vývoj složený z jednotlivých cyklů (všechny obsahují povinné plánování, ověření a analýzu rizik)
  - 1. cyklus - nalezení globálních rizik, zpracování základního konceptu a rozhodnutí o použitých metodách
  - 2. cyklus - konstrukce a ověření specifikace požadavků
  - 3. cyklus - vytvoření a ověření detailního designu
  - 4. cyklus - implementace, testování a integrace

  Viz obr.

- Z tohoto modelu se vyvinuly další modely např. model Win-Win 1994 (je detailnější, obsahuje konkrétní náplň).

- V současnosti se již spirálový model nevyvíjí, na základě těchto principů však vzniká mnoho dalších propracovaných metodologií, např. Rational Unified Process, obecně iterativní metoda vývoje.

Spirálový model životního cyklu



Obrázek 7-1 – schéma spirálového modelu životního cyklu

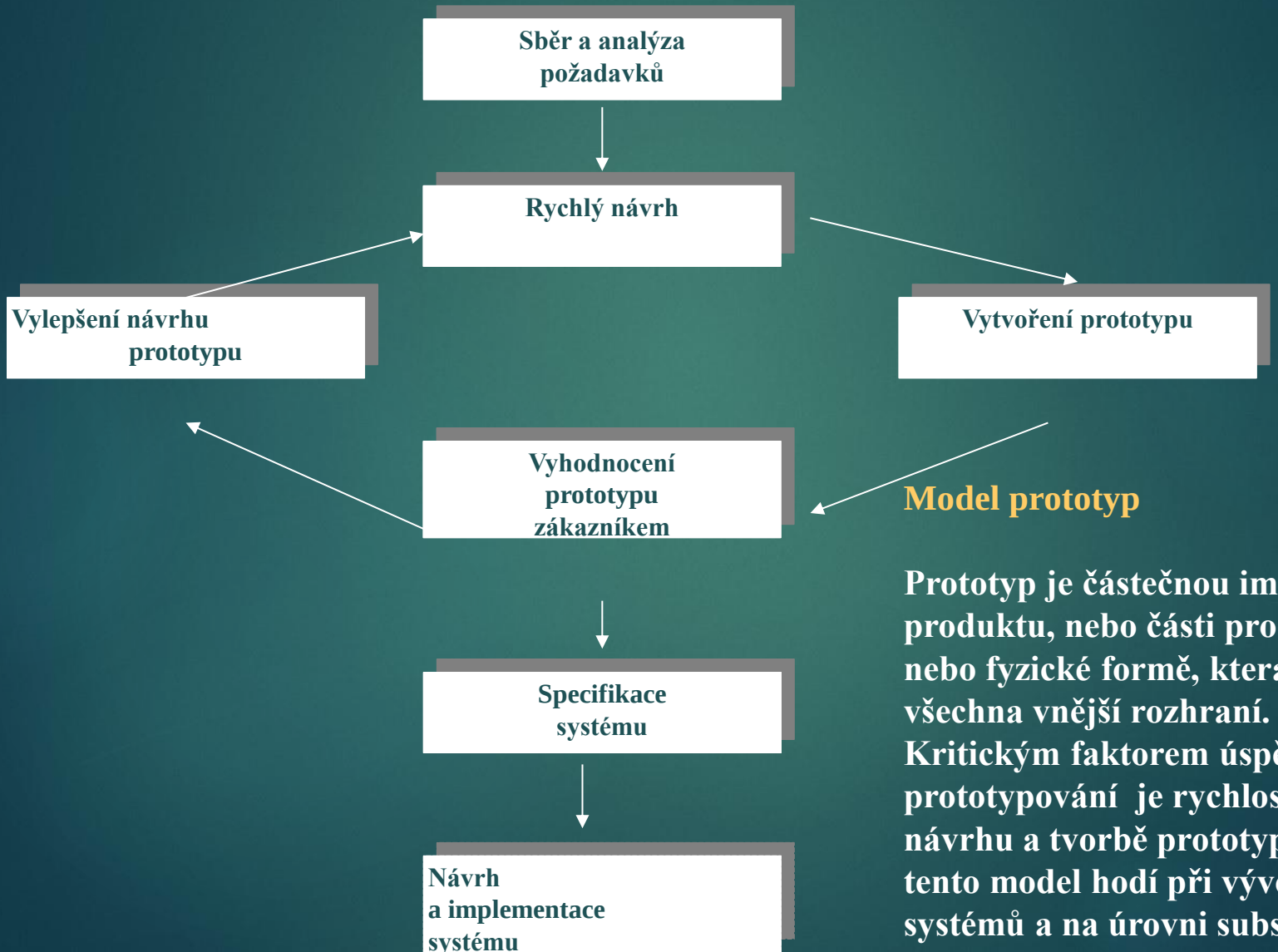
Model prototypování

- Prototyp IS je **dočasná verze systému**, která ukazuje základní rysy systému, který bude později implementován
- Prototyp musí být vytvořen rychle - speciální prostředky
- Na prototypu se odzkouší schůdnost, účinnost řešení, obrazovky atd.
- Používají se RAD nástroje (Rapid Application Development)

Typy prototypu

- ***Ilustrativní prototyp*** (důraz je kladen na vzhled a uživatelské rozhraní, vhodný při dialogu se zákazníkem, RAD nástroje, vyhazovací prototyp)
- ***Funkční prototyp*** (implementuje se jádro systému, minimální počet funkcí a postupně se přidávají, tzv. vývojový prototyp)
- ***Ověřovací prototyp*** (implementuje se jen část systému aby se ověřilo, zda vyhovuje požadavkům nebo technologii)

Model prototypování



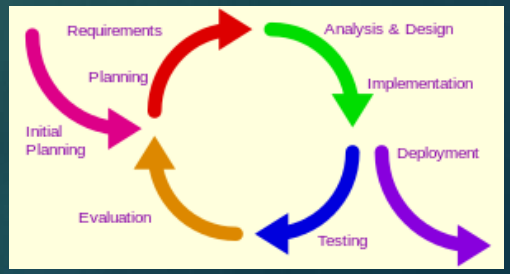
Model prototyp

Prototyp je částečnou implementací produktu, nebo části produktu, v logické nebo fyzické formě, která prezentuje všechna vnější rozhraní.
Kritickým faktorem úspěšného prototypování je rychlost obrátky při návrhu a tvorbě prototypů. Většinou se tento model hodí při vývoji menších systémů a na úrovni subsystémů.

Iterativní metoda vývoje

- **Vodopádový životní cyklus** vývoje SW platí **pouze** za těchto předpokladů:
  - požadavky budou stálé
  - systém lze navrhnout na „papíru”
  - integrace se dá provést v krátkém čase
  - integrace nevnese nové chyby do systému

- Pro současné velké projekty v období neustálých změn je vhodnější **iterativní metoda**, kde iterace = malý vodopád. První hrubé rozdělení životního cyklu je na **fáze**, každá fáze má **několik iterací**, a každá fáze končí **milníkem**.



Iterativní metoda vývoje Fáze

- **počátek (asi 10% celkového času)**
  - vize projektu
  - definice obchodního případu
  - definice rozsahu projektu
  - milník **Rozsah projektu (souhlas zadavatele s rozsahem systému a odhadem nákladů, porozumnění požadavkům, definice rizik a priorit)**
- **elaborace (asi 30% celkového času)**
  - funkční požadavky
  - základní linie architektury
  - plán pro další fáze a další iterace
  - milník **Architektura (stabilní architektura, její popis, model use case, dodatečné požadavky, revidovaný seznam rizik a priorit, plán pro zbytek projektu).**

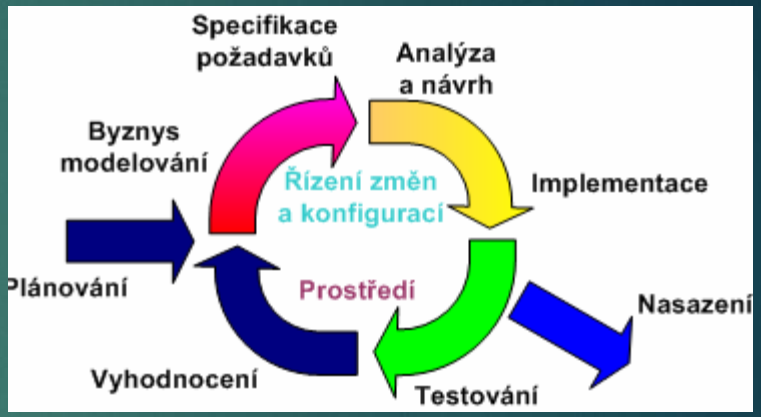
Iterativní metoda vývoje
Fáze

- **konstrukce (asi 50% celkového času)**
  - výroba produktu tzv. beta verze
  - příprava nasazení
  - milník **Úvodní funkčnost (funkční a stabilní verze systému, pokrytí požadavků, eliminace rizik, uživatelská dokumentace)**
- **zavedení (asi 10% celkového času)**
  - nasazení produktu do rutinního provozu
  - výroba médií a dokumentace
  - instalace
  - školení a podpora
  - milník **Nasazení produktu (systém v rutinním provozu, vyškolení uživatelé, fungující podpora, vyladěný výkon, vyhodnocení projektu a doporučení, jak dál)**

Iterativní metoda vývoje

Každá iterace obsahuje

- **Základní pracovní postupy:**
  - správa požadavků
  - analýza a návrh
  - implementace
  - testování
  - vyhodnocení plánu.
- **Podpůrné pracovní postupy**
  - řízení projektu
  - konfigurace prostředí
  - konfigurační řízení.
- **Další pracovní postupy (pouze v první nebo poslední iteraci)**
  - obchodní modelování
  - úvodní plánování
  - dodání.



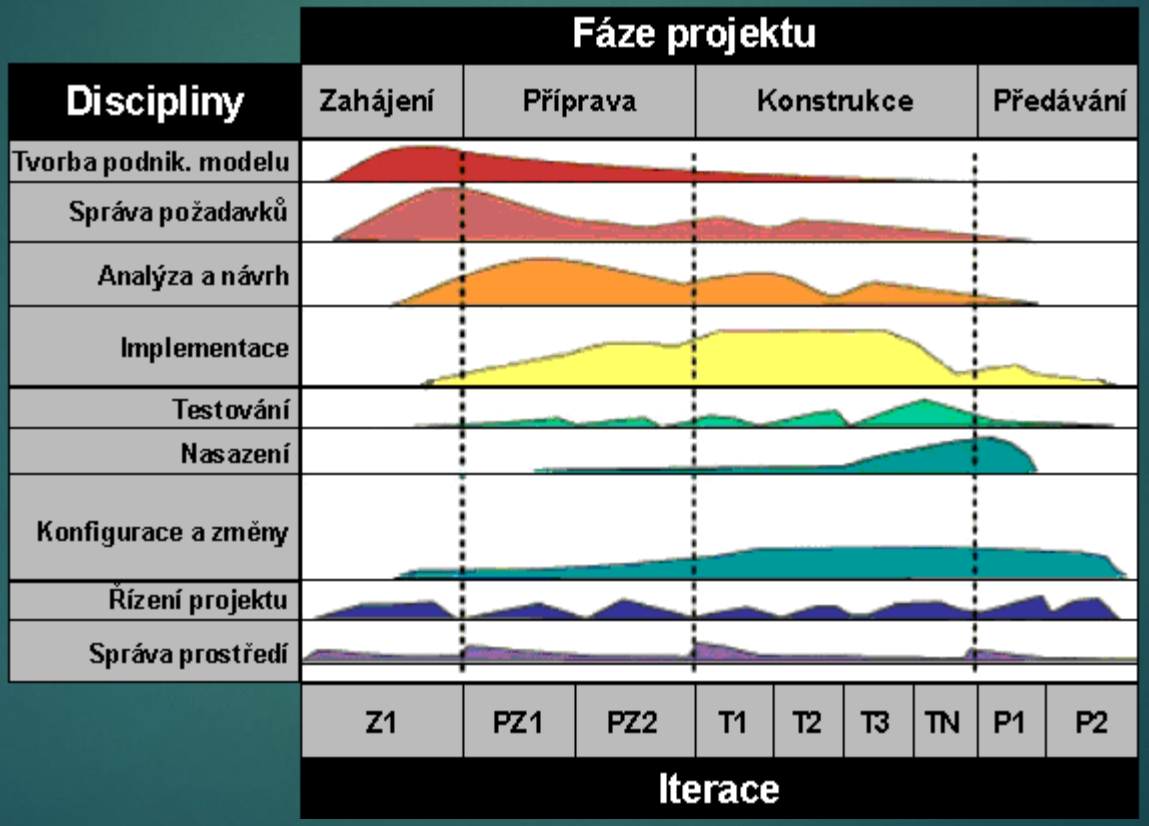
Iterace vývoje IS

RUP

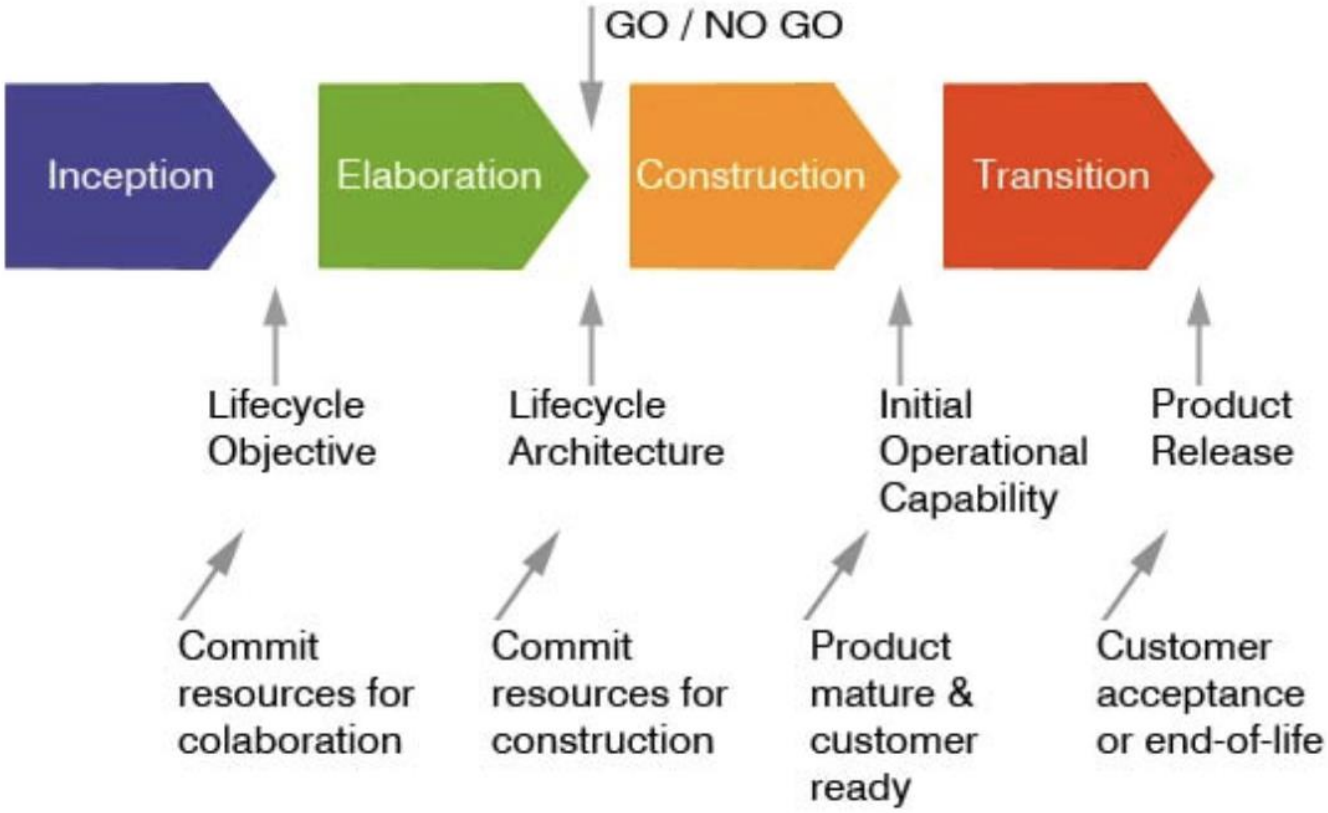
- **Rational Unified Process** - RUP je konkrétní metodologie založená na iterativní metodě.
- **Klíčové pojmy**
  - Iterativní vývoj SW
  - Správa a řízení požadavků
  - Použití komponentové architektury
  - Vizuální modelování
  - Průběžné zjišťování a ověřování kvality SW
  - Řízení změn
- **Elementy metodiky**
  - Pracovníci a role
  - Aktivity
  - Artefakty
  - Pracovní procesy

- Pozn. Unified Software Development Process je podobná RUP, ale je bezplatně dostupná.

RUP

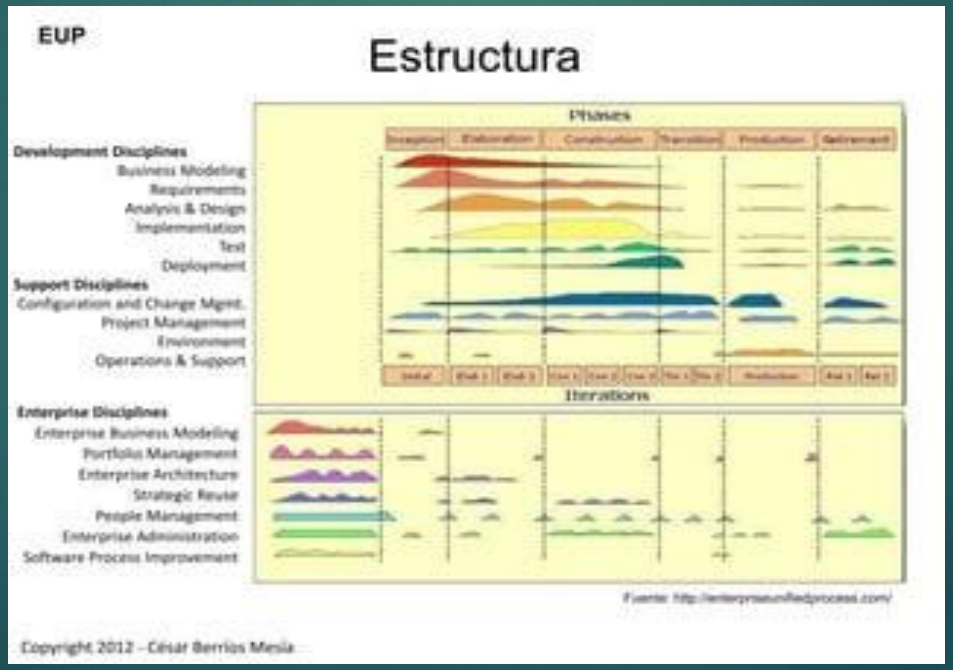

Fáze projektu
Disciplíny
Zahájení
Příprava
Konstrukce
Předávání
Tvorba podnik. modelu
Správa požadavků
Analýza a návrh
Implementace
Testování
Nasazení
Konfigurace a změny
Řízení projektu
Správa prostředí
Z1
PZ1
PZ2
T1
T2
T3
TN
P1
P2
Iterace

Rational Unified Process (RUP)



Poznámka : EUP

- **The Enterprise Unified Process (EUP)** is a software development methodology that is based on the principles of the Rational Unified Process (RUP).
- Jedná se o komplexní a přizpůsobitelný přístup k vývoji softwaru a poskytuje rámec pro organizaci a koordinaci práce vývojového týmu, oproti RUP obsahuje pracovní postupy podnikových procesů.



Jiné modely životního cyklu

- **Model RAD (Rapid Application Development)**

– model určený pro dobře srozumitelné a dobře vymezené problémy, s malými riziky, využívající krátký vývojový cyklus (cca do 3 měsíců), problém je rozdělen na samostatné moduly – model založen na rychlé tvorbě prototypů

- **Evoluční model**

– využívá skládání komponent, které mohou být vyvíjeny současně, či zakoupeny a upraveny

- **Formální metody**

– využívají specifikací řízený styl vývoje, tj. generování programů ze

Specifikací pomocí CASE

- **Extrémní programování** a podobné agilní metodiky a techniky viz dále
- **Testy řízený vývoj** (spirála s důrazem na testy)

Agilní metodiky vývoje SW

http://blog.czm-cvut.cz/agile-software-development-cast-prvni-manifest-agilniho-vyvoje-software

- Cílem těchto metodik je zajistit vytvoření SW produktu rychleji a efektivněji a snáze tak splnit požadavky dnešní doby.

- **Základní principy**
  - Iterativní a inkrementální vývoj s krátkými iteracemi
  - Důraz na přímou, osobní komunikaci v týmu
  - Nepřetržité sepětí se zadavatelem resp. uživatelem
  - Rigorózní, opakované průběžné automatizované testování.

- Nejsilnější důraz je zde kladen na zdrojový kód.
- Vhodné pro projekty s nejasným, nečistým nebo často se měnícím zadáním.

Agilní metodiky vývoje SW

- Skupina moderních metodik, které vycházejí z toho, že jedinou cestou, jak prověřit správnost navrženého systému, je **rychle ho (nebo jeho část) navrhnout, předložit zákazníkovi k vyzkoušení a mít zpětnou vazbu**. Usnadnit změnu je mnohem efektivnější než se snažit jí zabránit a je třeba se učit reagovat na nepředvídatelné události.

Dává se přednost:

- individualitám a komunikaci před procesy a nástroji,
- provozuschopnému SW před obsažnou dokumentací,
- spolupráci se zákazníkem před sjednáváním kontraktu,
- reakci na změnu před plněním plánu.

Agilní metodiky vývoje SW

- **Manifest agilního vývoje**
  - Dokument se základními principy

https://www.smartsheet.com/comprehensive-guide-values-principles-agile-manifesto
Manifest agilního vývoje

- **Konkrétní metodiky**
  - Extrémní programování (XP)
  - SCRUM Developmet Process
  - Vývoj řízený vlastnostmi (FDD - Feature Driven Development)
  - Vývoj řízený testy (TDD - Test Driven Development)
  - Crystal metodiky

https://www.rascasone.com/cs/blog/co-je-scrum-jak-funguje

Agilní metodiky vývoje SW

Extrémní programování (XP)

- 5 základních hodnot XP (komunikace, jednoduchost, zpětná vazba, odvaha a respekt).
- 12 základních postupů (např. malé verze, jednoduchý návrh, párové programování, nepřetržitá integrace, atd.)
- 4 základní činnosti (testování, psaní kódu, poslouchání a navrhování).

- Vhodná metodika pro menší týmy, kteří pracují na měnících se nebo nejasných zadáních, nebo tam, kde je jednoduchá zpětná vazba.

Agilní metodika SCRUM

Slovník pojmů

- **Release** - **o**bdobí vývoje systému, na jehož konci se uživateli dodává nová verze/aktualizace produktu.
- **Sprint** - **o**bdobí trvající 2-4 týdny (v praxi nejčastěji 3 týdny). Sprint obsahuje několik User Stories.
- **User Story (US, uživatelský příběh)** - **s**lovní popis požadavku zákazníka. US jsou vytvářeny Product Ownerem. Po zahrnutí US do konkrétního Sprintu by se zadání US nemělo měnit.
- **Backlog Item (BLI)** - **p**opis problému na technické úrovni = pro programátora. Existují dva typy Backlog Itemu - Functional a Technical. Funkční BLI popisuje více do podrobna zadání User Story, zatímco technický BLI říká, co je potřeba pro vyřešení zadání (např. instalace serveru, konfigurace, refactoring,…).
- **Task** - Samotný úkol, který vykonává člen Scrum týmu.
- **Daily meeting** - Denní střetnutí celého týmu. Tato střetnutí jsou pro metodiku klíčová. Na střetnutí každý ze členů prezentuje co dělal předešlý den, co dělá dnes a s jakými problémy se setkal. Pokud se vyskytnou nějaké problémy, které zabraňují dalšímu pokračování v práci, Scrummaster nebo ostatní členové týmu udělají potřebné kroky k jejich odstranění.

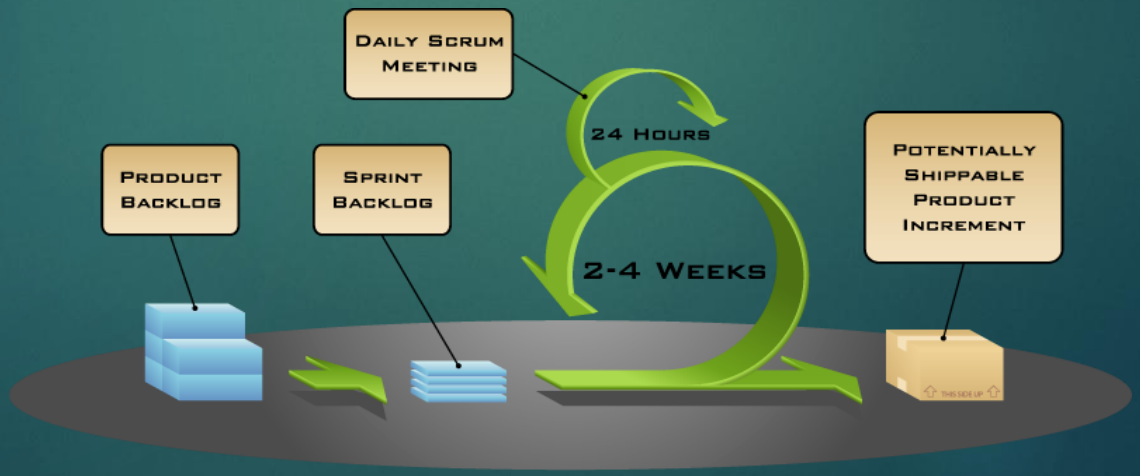
Role

- **Product Owner (PO)** - **t**ato osoba má za úkol komunikaci se zákazníkem.
- **Scrummaster (SM)** - Hlavní postava metodiky Scrum. Jedná se o člověka, který zajišťuje správné fungování vývojového týmu, odstraňuje překážky při vývoji. Společně s Product Ownerem se podílí na plánování Sprintů.
- **Scrum Team Member (STM)** - Člen vývojového týmu. Může se jednat o vývojáře, testera, analytika, dokumentaristu,…

SCRUM

Vývojový cyklus

- V první fázi vykomunikuje **PO** se zákazníkem zadání nových požadavků. Ty tvoří tzv. **User Stories (Product Backlog)**. Poté se na **Sprint Planning Meetingu** sejde **PO, SM a Scrum Team** a společně odhadnou zadané **User Stories**. Poté podle priorit naplánují budoucí **Sprint**, tedy vyberou **US**, které budou v tomto **Sprintu** dokončeny.
- Tyto **US** jsou poté ve **Scrum Teamu** dále rozepsány do **BLI (Sprint Backlogu)** a ty následně do **Tasků**. Během trvání sprintu (asi 3 týdny) probíhají každodenní meetingy **(Daily Scrum Meetings)**. Na konci Sprintu, resp. **Releasu** je zákazníkovi předvedeno demo vzniklých úprav. Zákazník se k nim může vyjádřit a zhodnotit, zda jsou splněny jeho požadavky. SCRUM díky tomu dokáže rychle reagovat na změny zadání od uživatele.



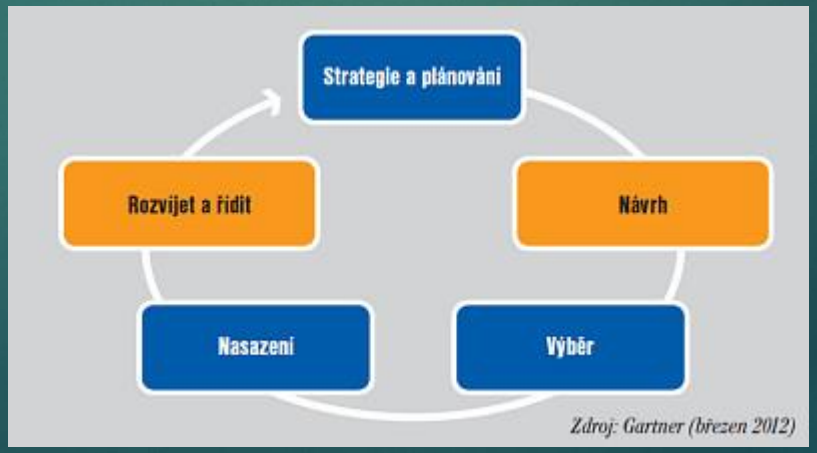
Řízení životního cyklu ERP systému (Gartner)

Aby komplexní podnikové aplikace typu ERP přinášely společnosti největší možnou přidanou hodnotu, je třeba řídit všechny fáze jejich životního cyklu. Analytici společnosti Gartner jich definují pět: strategie a plánování, návrh, výběr, nasazení a řízení a rozvoj.

- **Strategie a plánování** (zakládáme ERP projekt, stanovujeme základní mantinely pro následující etapy, definujeme procesy, jejichž rozsah by měl ERP pokrývat, slaďujeme potřeby s obchodními a IT strategiemi, vytváříme **základní dokumenty projektu**, určujeme důležitost jednotlivých kroků, **tvoříme plány a rozpočet**, vybíráme členy projektového týmu a řídicí a kontrolní systémy)
- **Návrh** (zpřesňuje se předchozí fáze a doplňují se o detaily, strategie a vize spojené s ERP systémem jsou transformovány do **konkrétních plánů a návrhů**. Pro fázi návrhu je typické **definování architektury, technologie a standardů** projektu, určení zdrojů a způsobu jejich získání, **modelování požadavků ze strany byznysu**, detailní specifikace požadovaných parametrů řešení, definování detailů procesů a výkonnostních měřítek, vytvoření podrobnějšího obchodního případu, vytvoření a schválení plánu řízení změn v organizaci včetně její struktury po nasazení řešení).
- **Výběr** (snažíme se získat informace z okolí firmy. Cílem této fáze je **vybrat řešení**, které bude vhodné právě pro náš podnik. Než k tomu ale dojde, je třeba provést celou řadu kroků jako např. finalizovat požadavky na řešení, rozšířit projektový tým o byznys uživatele, vytvořit **zadávací dokument**, zvolit technologii a dodavatele či poskytovatele služeb, vyjednat SLA a smlouvu.

Řízení životního cyklu ERP systému

- **Nasazení** (určujeme, kdo **bude systém implementovat a kdo bude tuto činnost řídit**. Je třeba kontrolovat dodržování projektu, harmonogramu a rozpočtu, monitorovat a minimalizovat rizika. Je také vhodné neustále sledovat rozdíl mezi našimi potřebami a tím, co můžeme reálně dostat například pomocí gap analýzy. A samozřejmě konfigurujeme, integrujeme, testujeme a zaškolujeme klíčové uživatele. Výstupem této fáze životního cyklu ERP by mělo být **rozhodnutí, zda řešení posunout ze stavu projektu do produkčního prostředí**).
- **Rozvíjet a řídit** (ptáme se, jak organizace ERP systém používá a jakým způsobem jej bude třeba přizpůsobit, aby dobře podporoval její neustále se měnící požadavky, řídí se a monitoruje činnost ERP, sleduje se jeho využívání a měří se výkonnost, sledují s rizika a měnící se požadavky ze strany byznysu a v důsledku toho se pak zlepšují vlastní procesy, dovednosti, metody a nástroje).



- Děkuji za pozornost.