MASARYKOVA UNIVERZITA

Základy projektování informačních systémů

Jaroslav Šmarda

Základy projektování IS

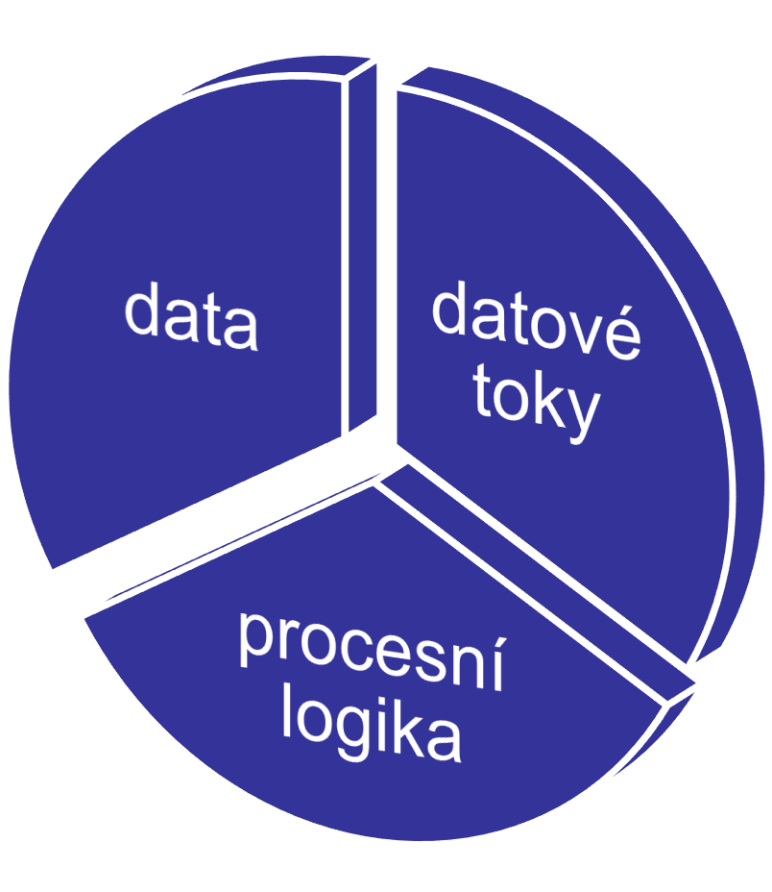
- Proces vývoje IS
- Generické modely softwarového procesu
  - Vodopádový model
  - Evoluční vývoj
  - Komponentový vývoj
- Iterace jako základ vývoje IS
  - Přírůstkový vývoj
  - Spirálový vývoj
- Další modely odvozené z iteračního vývoje
  - RAD - rychlý vývoj aplikací, prototypování, agilní vývoj aplikací
- CASE - počítačem podporovaný softwarový vývoj

Proces vývoje IS

- Základním cílem je:
  - zvýšení efektivity klíčových podnikových procesů pro pracovníky, kteří využívají počítačovou podporu softwarových řešení

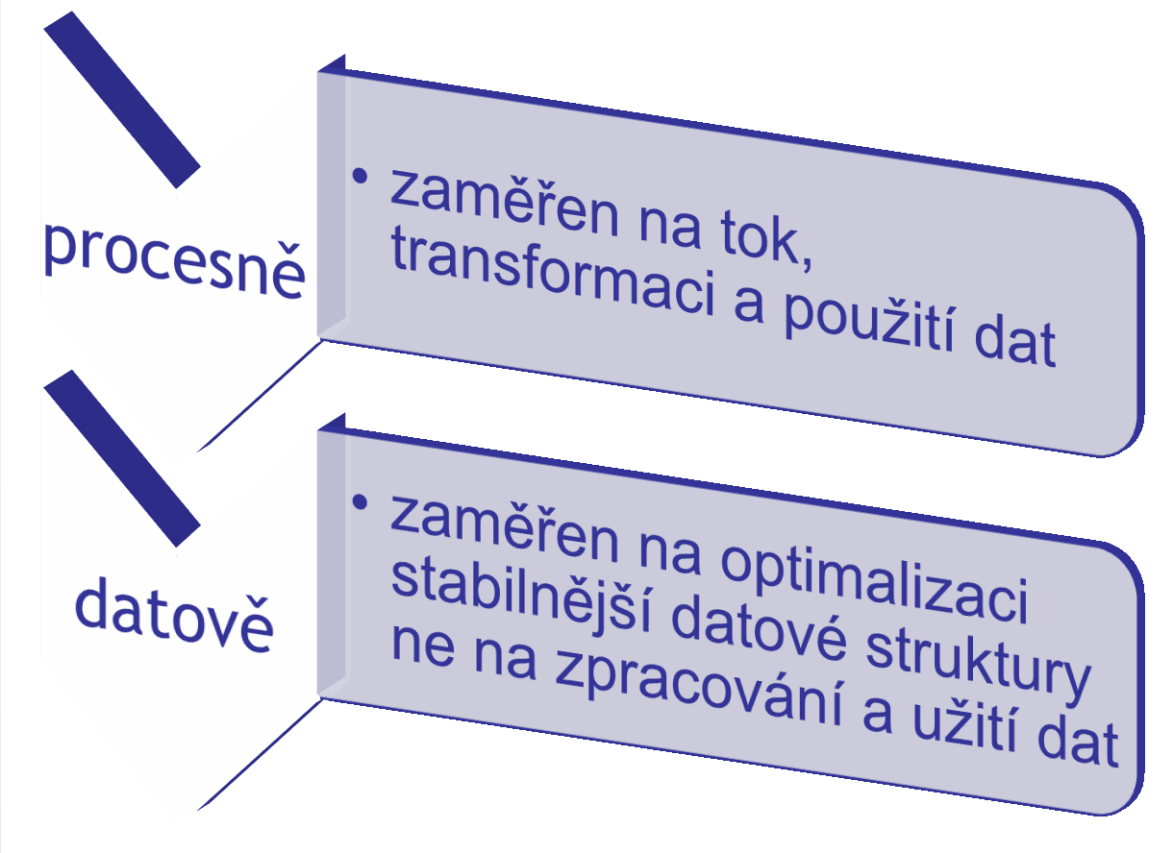
Rozvoj IS může probíhat těmito způsoby

3 složky analýzy a návrhu IS



Analýza a návrh IS

- Přístup může být orientován:



Analýza a návrh IS

- Široké spektrum technik pro různé fáze životního cyklu vývoje:
  - čistě formální (např. Petriho sítě)
    - cílem je jednoznačnost a úplnost
  - poloformální techniky (např. jazyk UML - standardní jazyk pro zobrazení, specifikaci, konstrukci a dokumentaci prvků systémů nejčastěji se softwarovou charakteristikou)
    - nyní nejpopulárnější
  - neformální techniky

Proces vývoje IS (softwarový proces)

- Strukturovaná množina aktivit, která zahrnuje:
  - specifikace
  - návrh
  - ověření
  - další vývoj
- Model softwarového procesu – abstraktní reprezentace procesu

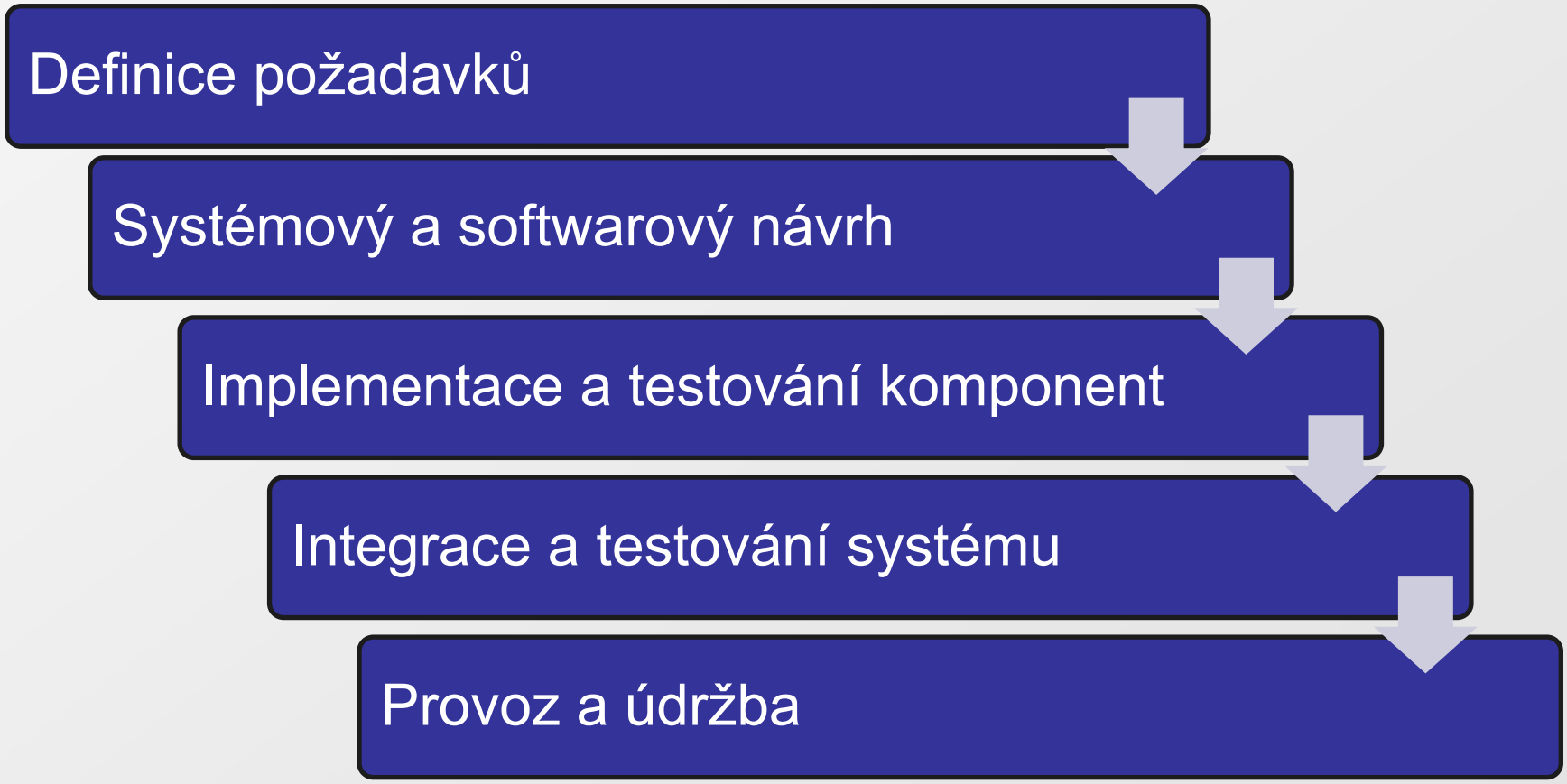
Generické modely softwarového procesu

- **Model softwarového procesu** představuje abstraktní reprezentaci tohoto procesu.
- **Generický model** pak představuje **šablonu** pro celou množinu (třídu) modelů, které mají společné rysy obsažené v generickém modelu.
- Na základě modelu softwarového procesu si můžeme představit řadu konkrétních softwarových procesů v organizacích, které vyvíjejí IS, včetně dodavatelů IS.

Generické modely softwarového procesu

- Vodopádový model
  - striktně odděleny fáze specifikace, vývoje a ověření
- Evoluční vývoj
  - specifikace, vývoj a ověření jsou vzájemně prokládány
- Komponentový vývoj
  - založený na použití existujících komponent

Vodopádový model



Problémy vodopádového modelu

- Nepružné rozdělení do fází
  - po ukončení fáze se k ní nelze vrátit, nelze reagovat na změny požadavků během vývoje
- Model je vhodný jen pro systémy s jasně danými a stabilními požadavky
  - jen málo podnikatelských systémů má stabilní požadavky
- Vodopádový model se používá pro velké projekty
  - systém vyvíjen na několika místech

Evoluční vývoj

- Základní myšlenkou **evolučního vývoje** je vcelku **libovolné prokládání jednotlivých fází** vývoje IS.
- Evoluční vývoj je spíše vnímán jako **řada malých vodopádových modelů** za sebou.
- Jedna z variant evolučního vývoje je označována jako **průzkumný vývoj**.

Evoluční vývoj

- Průzkumný vývoj
  - Cílem je práce se zákazníkem a postupný vývoje finálního systému na základě nástinu specifikace
  - Mělo by se začít s několika dobře pochopenými požadavky a pak přidávat nové funkce, které navrhuje zákazník
- Prototypování
  - Cílem je pochopit požadavky zákazníka
  - Mělo by se začít se špatně specifikovanými požadavky zákazníka a postupně vyjasnit, co je vlastně požadováno

Evoluční vývoj

- Problémy
  - Špatně sledovatelné fáze procesu
  - Systémy jsou obvykle špatně strukturovány
  - Požadavky na speciální nástroje
    - např. programovací jazyk pro zápis prototypů
- Použitelnost
  - Pro malé a střední interaktivní systémy
  - Pro části velkých systémů
    - např. uživatelské rozhraní
  - Pro systémy s krátkou životností

Komponentový vývoj

- Založený na systematickém znovupoužití existujících komponent
- Systém vzniká integrací existujících komponent
- Stádia procesu
  - Analýza komponent
  - Modifikace požadavků
  - Návrh systému s použitím existujících komponent
  - Vývoj a integrace
- Stále více populární

Principy komponentového vývoje

- Komponenty jsou nezávislé a vzájemně do sebe nezasahují
- Implementace komponenty je skrytá
- Komunikace s okolím je prostřednictvím dobře definovaných rozhraní (interface)
- Využití komponent snižuje náklady

Komponenta jako poskytovatel služby

- Komponenta je nezávislá spustitelná entita a nemusí být přeložena před tím, než je použita společně s jinými komponentami
- Služby poskytované komponentami jsou dostupné jedině prostřednictvím rozhraní, veškerá komunikace mezi komponentami je prostřednictvím rozhraní

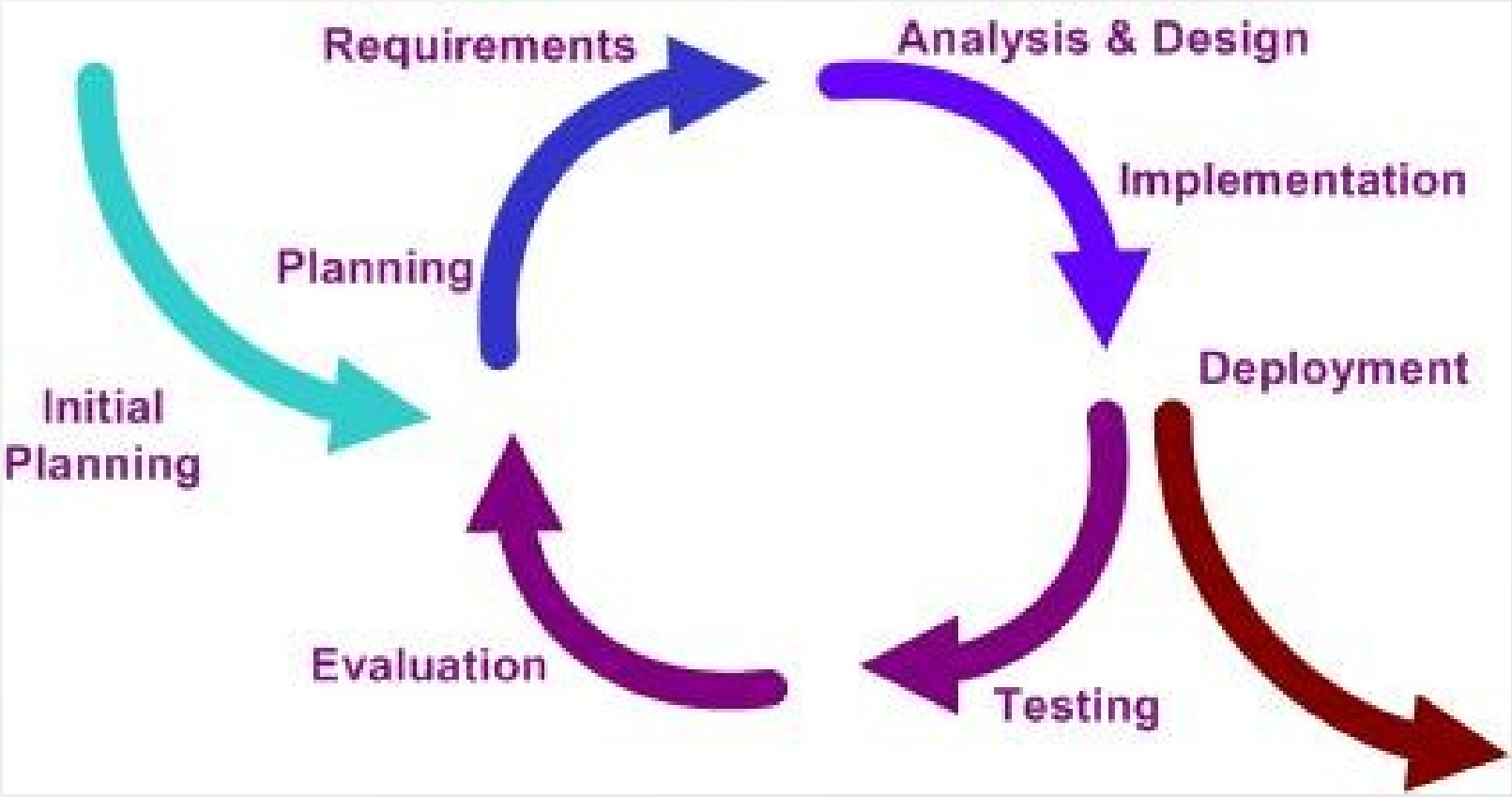
Komponentové rozhraní

- Poskytuje rozhraní
  - Definuje služby, které komponenta poskytuje ostatním komponentám
- Požaduje rozhraní
  - Definuje takové služby, která komponenta potřebuje od ostatních komponent, aby komponenta mohla poskytnout své komponenty

Iterace jako základ vývoje IS

- Požadavky na software se VŽDY vyvíjejí během procesu vývoje
  - iterace, při kterých jsou přepracovávány dřívější fáze jsou součástí procesu vývoje větších systémů
- Iterace může být použita pro libovolný generický model vývoje
- Dva přístupy k iteraci vývoje:
  - Přírůstkový vývoj (extrémní programování)
  - Spirálový vývoj

Iterace jako základ vývoje IS



Iterační vývoj - přírůstkový

- Spíše než dodávka celého systému jako jediného celku, je systém dodáván po částech (komponentách) s definovanou funkcionalitou
- Uživatelským požadavkům jsou přiřazeny priority a nejdříve jsou dodávány komponenty s funkcemi nejvyšší priority
- Požadavky na komponentu, která je právě vyvíjena (aktuální přírůstek), se zastaví, zatímco požadavky na pozdější komponenty přibývají

Výhody iteračního přírůstkového vývoje

- Součástí každé dodávky jsou nové funkce, takže řada funkcí je přístupná dříve než kdyby byl systém dodán jako celek
- První dodávky slouží jako prototypy, které pomáhají zpřesnit požadavky na nové funkce v dalších dodávkách
- Nižší riziko celkového selhání projektu
- Funkce s vyšší prioritou jsou více testovány

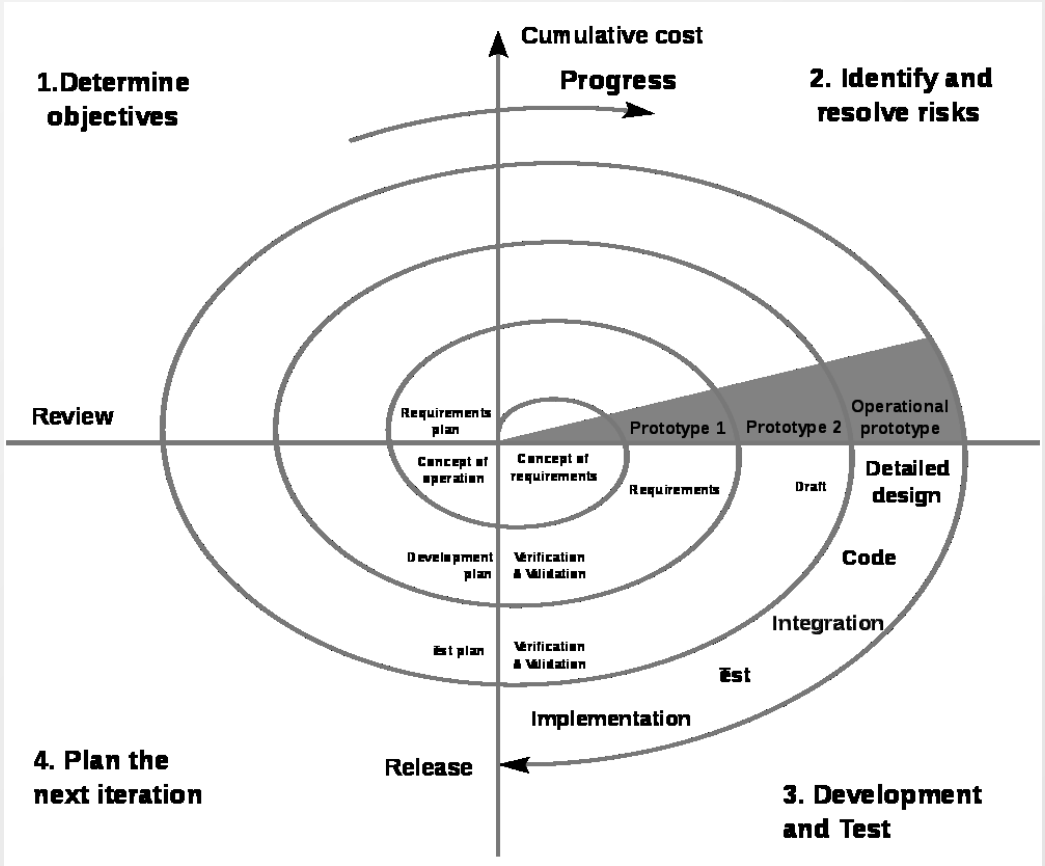
Iterační vývoj - extrémní programování

- Přístup založení na velmi malých přírůstcích – kontinuální vývoj
- Neustálé vylepšování kódu
- Uživatelé součástí vývojového týmu

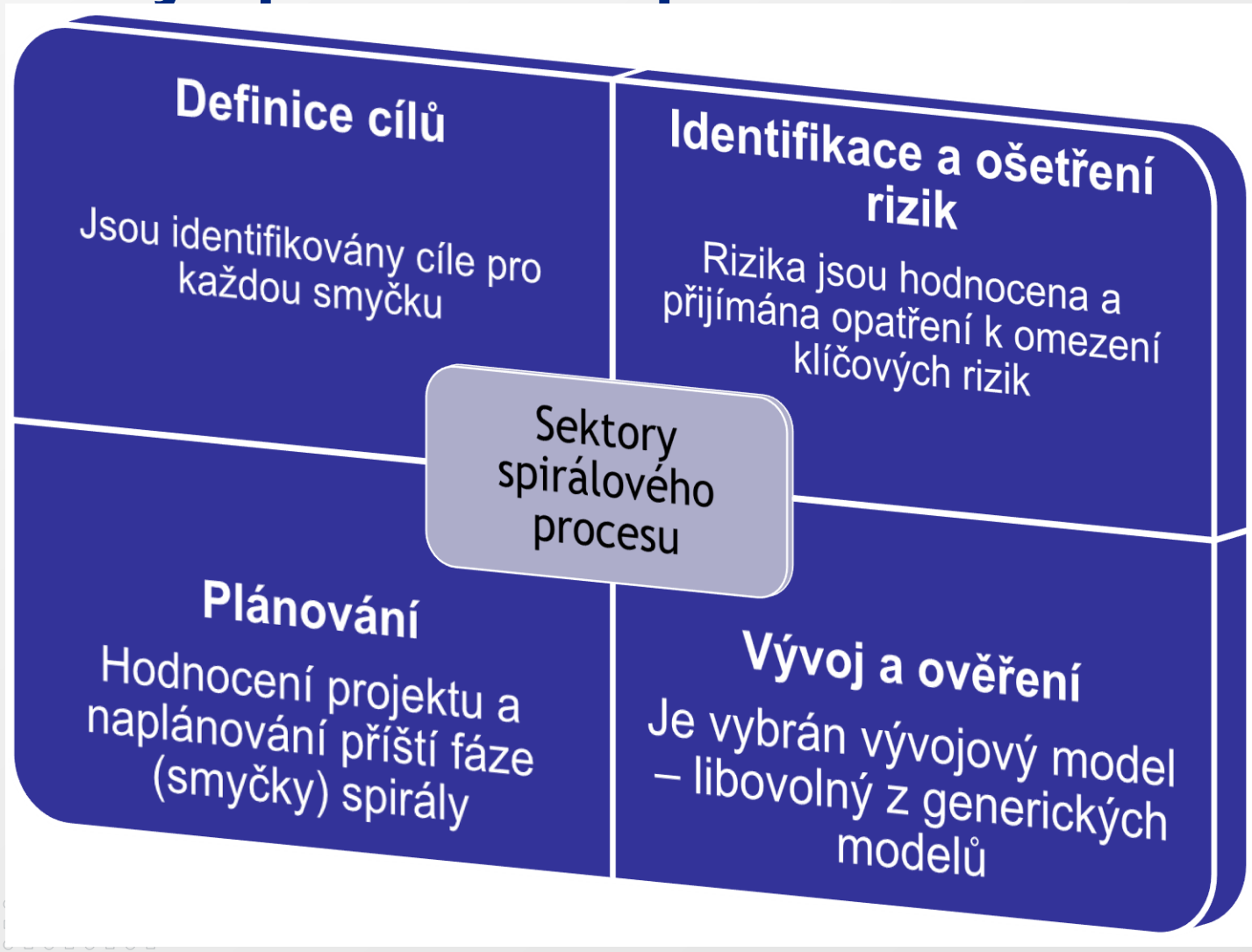
Iterační vývoj - spirálový

- Vývojový proces reprezentován jako spirála než jako posloupnost činností se zpětnou vazbou
- Každá smyčka spirály představuje fázi vývojového procesu
- Žádné přesně definované fáze (specifikace, návrh,..), každá smyčka spirály obsahuje činnosti, které jsou třeba
- Rizika jsou explicitně ohodnocena a v každé fázi přijímána příslušná opatření

Spirálový model vývojového procesu



Sektory spirálového procesu



Rychlý vývoj aplikací (RAD Rapid Application Development)

- Kvůli rychle se měnícímu prostředí musí podnik rychle reagovat na nové příležitosti a hrozby
- To vyžaduje změny v softwaru a požadavek na rychlý vývoj nového softwaru nebývá základním požadavkem implementace
- Podnik může být ochoten akceptovat nižší kvalitu výměnou za rychlý vývoj

Požadavky

- Kvůli rychle se měnícímu prostředí často není stabilní a konzistentní systém požadavků na vývoj
- Proto je vodopádový model nepraktický a vývoj založený na iteracích specifikací a distribucí je jediný způsob, jak dodat software rychle

Vlastnosti procesů RAD

- Procesy specifikace, návrhu a implementace jsou paralelní. Není žádná detailní specifikace a dokumentace návrhu je minimalizována
- Systém je vyvíjen přírůstkově. Uživatel hodnotí každý přírůstek a dává návrhy pro další přírůstky
- Uživatelská rozhraní jsou vyvíjena s využitím interaktivních vývojových nástrojů

Prototypování (Prototyping)

- Pro vývoj některých velkých systémů může být iterativní vývoj a distribuce nepraktická především v případě několika týmů pracujících na různých místech
- Při prototypování je vyvinut experimentální systém, který slouží pro formulaci požadavků. Tento experimentální systém je zahozen v okamžiku, kdy je dokončena specifikace systému

Agilní vývoj aplikací

- Nespokojenost s vysokými náklady vedla k vývoji agilních metod. Tyto metody:
  - se soustřeďují spíše na kódování (programování) než na návrh
  - jsou založeny na iterativním přístupu
  - jsou zaměřeny na rychlý vývoj a distribuci softwaru, který reaguje na měnící se požadavky
- Agilní metody jsou vhodné především pro malé a středně velké aplikace

Principy agilního vývoje aplikací

Princip	Popis
Zapojení zákazníka	Zákazník definuje požadavky a hodnotí výsledky jednotlivých iterace
Přírůstkové distribuce	Software je vyvíjen v přírůstcích a zákazník ovlivňuje každou iteraci
Lidé a ne procesy	Lidé se nepřizpůsobují předdefinovaným procesům
Přijetí změny	Požadavky se mění a vývoj se musí přizpůsobit změnám
Jednoduchost	Eliminace komplexnosti, důraz na jednoduchost

CASE (Computer-Aided Software Engineering) - počítačem podporovaný softwarový vývoj

- Činnosti podporované počítačem
  - Grafické editory pro vývoj systémového modelu
  - Datový slovník pro správu prvků návrhu
  - Grafické nástroje pro tvorbu uživatelského rozhraní
  - Programy pro hledání chyb (Debugger)
  - Překladače pro generování nových verzí programu

Technologie nástrojů CASE

- Použití CASE nástrojů vedlo ke znatelnému zkvalitnění vývoje, ale nepřineslo zásadní změny, které se očekávaly
  - Softwarový vývoj vyžaduje kreativní myšlení a ten nelze snadno automatizovat
  - Softwarový vývoje je týmová činnost, většinu vývoje představuje týmová interakce, které CASE nástroje přímo nepodporují

Nástroje CASE podle stupně integrace

- Nástroje (Tools)
  - Podporují individuální procesní úkoly jako je kontrola konzistence návrhu, editace textů apod.
- Pracovní stoly (Workbenches)
  - Podporují procesní fáze jako je specifikace nebo návrh, obvykle obsahují několik nástrojů
- Prostředí (Environments)
  - Podporují všechny podstatné části sofwarového vývoje, obvykle obsahují několik Pracovních stolů (Workbenches)