Úvod do softwarového inženýrství

IUS 2024/2025

8. přednáška

Ing. Radek Kočí, Ph.D.
Ing. Bohuslav Křena, Ph.D.

8. a 11. listopadu 2024

Studijní koutek – Studentská unie FIT

- SU FIT zastupuje zájmy studentů FIT.
  - Studenti mají své zástupce (4+1) v Akademickém senátu FIT.
  - Studenti mají svého zástupce v Akademickém senátu VUT.
  - Studenti mají své zástupce na Kolegiu děkana FIT.
  - Studenti mají svého zástupce v radách studijních programů.
  - Studenti mají svého zástupce v Radě pro využití informačních technologií a vybavení či v Knihovní radě.
  - Polovinu Disciplinární komise tvoří studenti.
- SU FIT pomáhá fakultě s organizací různých akcí: Gaudeamus, zápisy, DOD, ...
- SU FIT organizuje vlastní akce: Studentský klub U Kachničky, workshopy, turnaje, deskovky, ples, DZD, ...
- Každý student FIT VUT v Brně se může přihlásit do SU FIT.
- Oficiální informace o SU FIT najdete na URL

  `https://www.su.fit.vut.cz/`

Zkouška – Variantní termíny

- 1. termín: **čtvrtek 2. 1. 2025, 13:00** (výsledky očekávány 9. 1. 2025)
  - **272 míst** (D105, D0206, D0207, E112)
- 2. termín: **pátek 10. 1. 2025, 9:00** (výsledky očekávány 17. 1. 2025)
  - **384 míst** (D105, D0206, D0207, E112, E104, E105, G202, A112)
- 3. termín: **pondělí 20. 1. 2025, 12:00** (výsledky očekávány 27. 1. 2025)
  - **218 míst** (D105, D0206, D0207)
- 4. termín: **pondělí 27. 1. 2025, 9:00** (výsledky očekávány 1. 2. 2025)
  - **218 míst** (D105, D0206, D0207)
- 5. termín: **pondělí 3. 2. 2025, 9:00** (výsledky očekávány 10. 2. 2025)
  - **384 míst** (D105, D0206, D0207, E112, E104, E105, G202, A112)

- Celkem vypsáno **1 476** míst pro **948** zapsaných studentů.
- Na studenta je **1,56** místa (tedy lehce více než minimum 1,50).

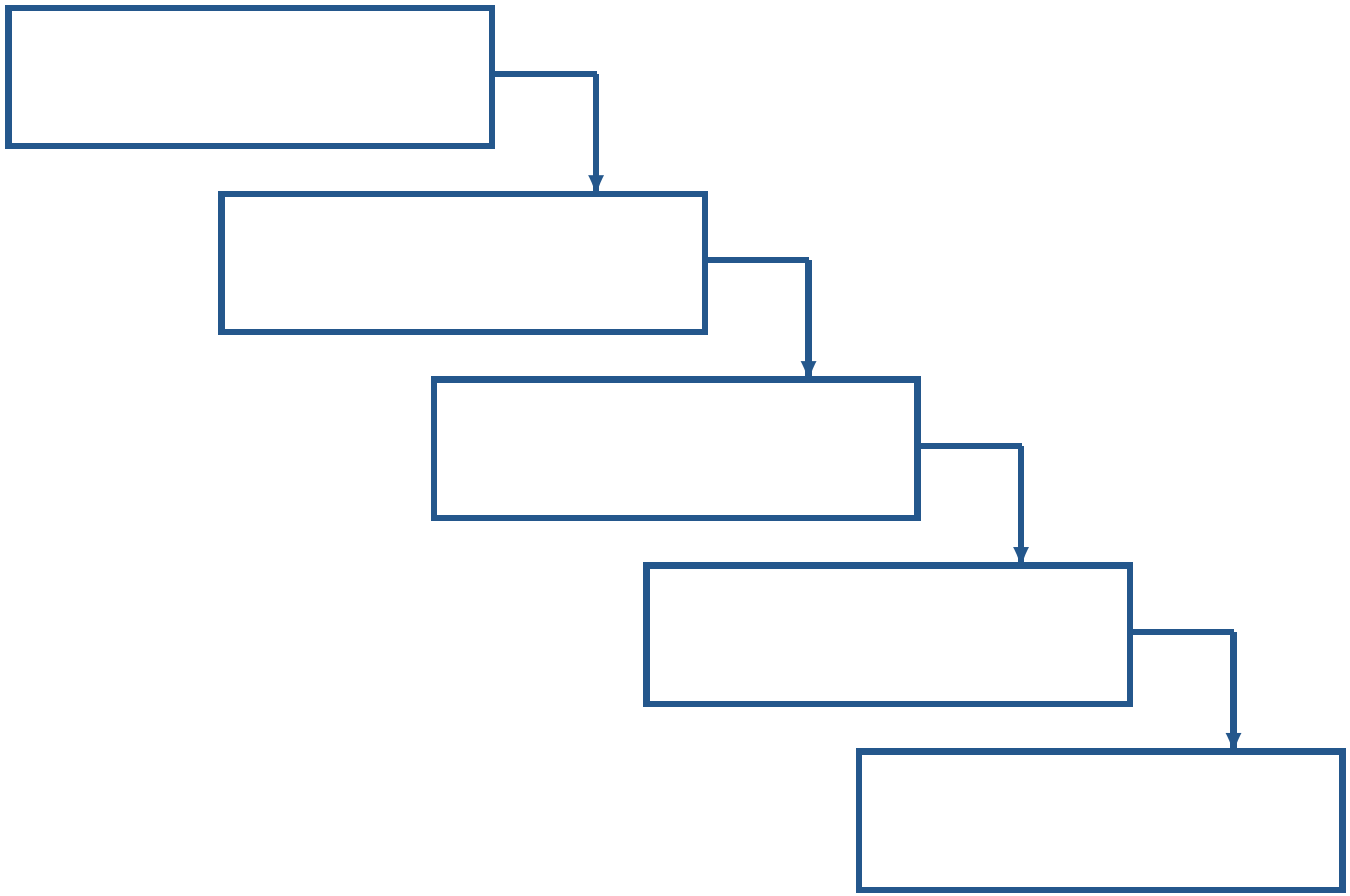
Téma přednášky

- **Modely životního cyklu**
- **Metodiky vývoje softwaru**
  - heavyweight metodiky
  - agilní metodiky

Lineární modely životního cyklu

Lineární (sekvenční) modely

- životní cyklus jde postupně od první etapy až do poslední



- typický představitel je vodopádový model

V-model

Vlastnosti

- vychází z vodopádového modelu
  - má stejné základní vlastnosti
  - zachovává si jednoduchost a srozumitelnost vodopádového modelu
- písmeno V symbolizuje grafické uspořádání etap, zdůrazňuje vazby mezi návrhovou a testovací částí
- písmeno V je také synonymem pro validaci a verifikaci

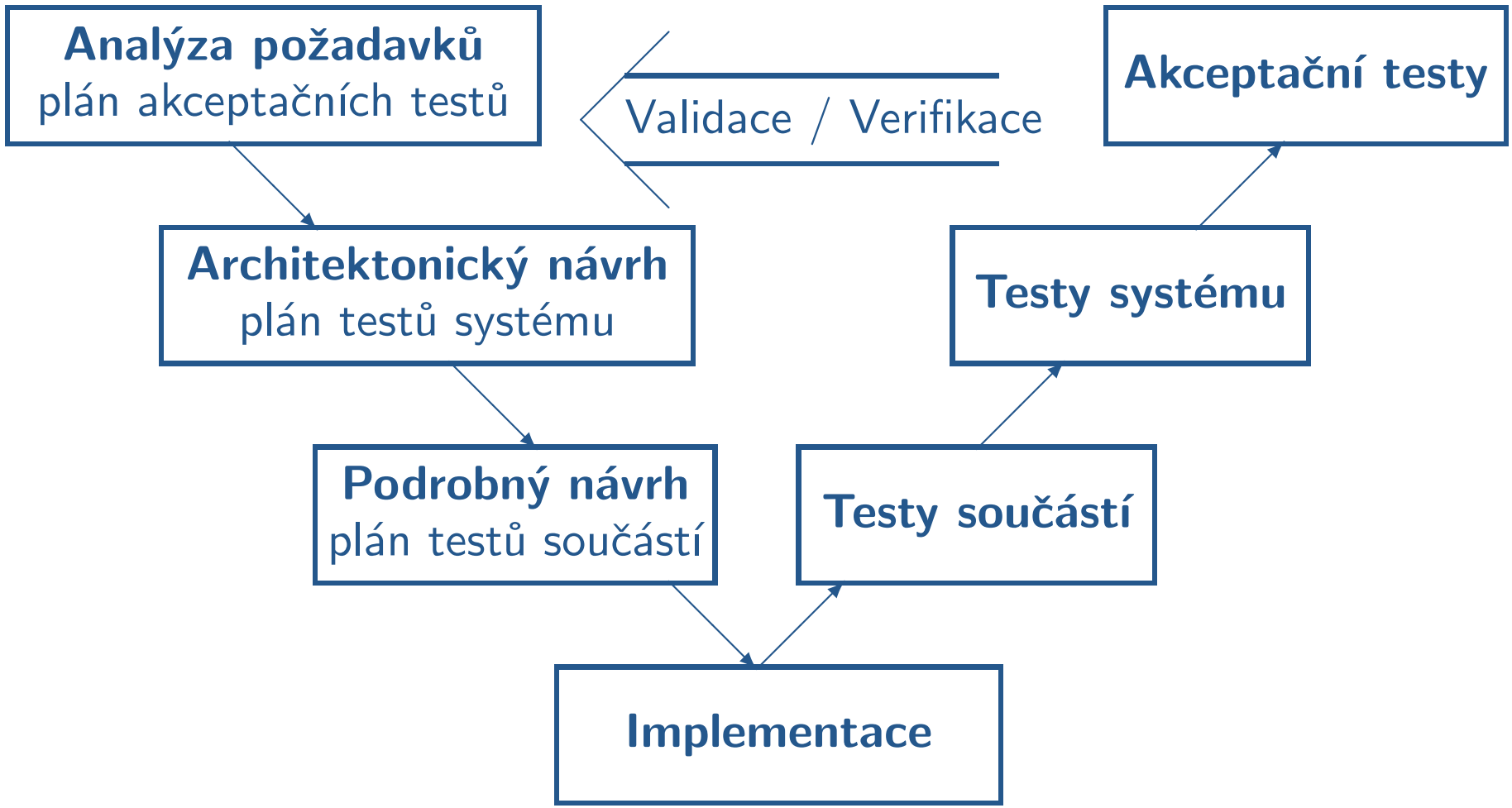
Levá část

- vývojové aktivity
- plánování testů

Pravá část

- testovací aktivity
- provádění testů podle plánů

V-model


Analýza požadavků
plán akceptačních testů
Validace / Verifikace
Akceptační testy
Architektonický návrh
plán testů systému
Testy systému
Podrobný návrh
plán testů součástí
Testy součástí
Implementace

W-model

Vlastnosti

- vychází z V-modelu
- aktivity spojené s ověřováním a testováním jsou na stejné úrovni jako návrhové aktivity $\Rightarrow$ druhé souběžné V

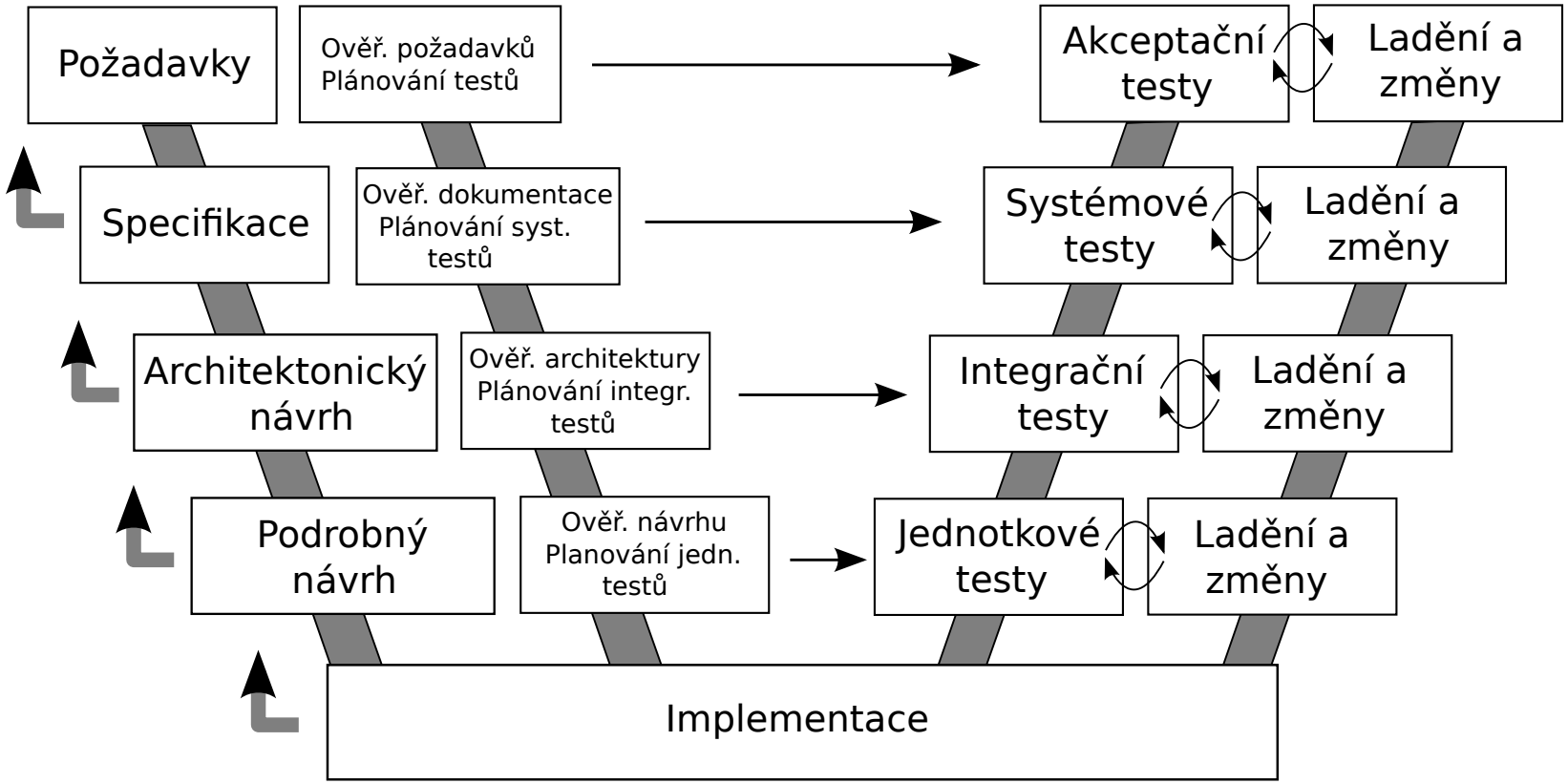
Levá strana

- V1: analýza, specifikace, návrh, ...
- V2: ověřování výstupů etap a plánování a návrh testů

Pravá strana

- V1: provádění testů (dle navržených plánů)
- V2: ladění, změny kódu, regresní testování, ...

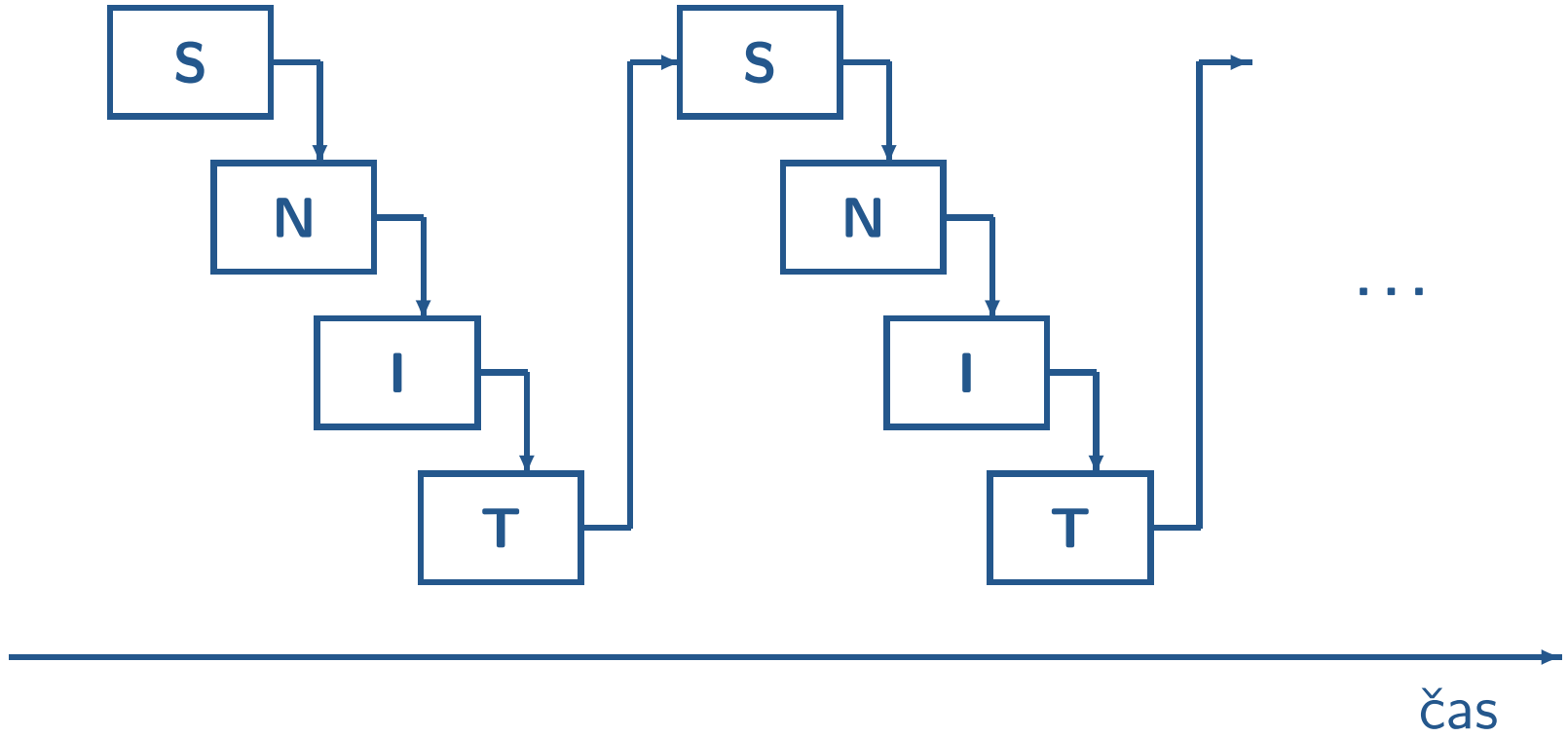
W-model


Požadavky
Ověř. požadavků
Plánování testů
Akceptační testy
Ladění a změny
Specifikace
Ověř. dokumentace
Plánování syst. testů
Systémové testy
Ladění a změny
Architektonický návrh
Ověř. architektury
Plánování integr. testů
Integrační testy
Ladění a změny
Podrobný návrh
Ověř. návrhu
Plánování jedn. testů
Jednotkové testy
Ladění a změny
Implementace

Iterativní modely životního cyklu

Iterativní modely

- sekvence etap se v životním cyklu opakuje



Iterativní modely životního cyklu

Vlastnosti

- systém se vyvíjí v iteracích
- v každé iteraci se vytvoří reálný výsledek
- zákazník se účastní vývoje (předpoklad)

Silné stránky

- v každé iteraci se vytvoří reálný výsledek $\Rightarrow$ zákazník má možnost validovat výsledek se svými požadavky, rychlejší odhalení chyb ve specifikaci

Slabé stránky

- náročnější na řízení
- potenciálně horší výsledná struktura
  $\Rightarrow$ existují techniky, jak tento nedostatek zmírnit (např. refaktorizace)

Inkrementální model

Vlastnosti

- Na základě specifikace celého systému se stanoví ucelené části – moduly
  např. plánování a řízení výroby, sklady, mzdy, . . .
- ty jsou pak vyvíjeny v samostatných vodopádech . . .
  . . . a po dokončení postupně předávány uživateli.

Silné stránky

- Omezuje projektová rizika.
  Jeden modul lze dodat rychleji než celý systém.
- Zjednodušuje zavedení změn během vývoje.
  zejména promítnutí získaných zkušeností do dalších modulů

Slabé stránky

- Vyžaduje dobré plánování a pečlivý návrh rozhraní mezi moduly.
- Vývoj po částech může vést ke ztrátě vnímání logiky celého systému.
- Nemusí být vhodný pro všechny systémy (např. překladač).

Spirálový model

Vlastnosti

- Barry Boehm, *A Spiral Model of Software Development and Enhancement*, 1986
- kombinace prototypování a analýzy rizik
- vyžaduje stálou spolupráci se zákazníky
- přístupy řízené riziky (*risk-driven approach*)

Proces vývoje

- vývoj je rozdělen na cykly, v každém cyklu se řeší ucelená část vývoje
- každý cyklus je rozdělen na kvadranty vymezující zásadní činnosti, které se mohou opakovat v každém následujícím cyklu (stanovení cílů, vyhodnocení, plánování)
- postupně se každým cyklem rozšiřuje množina zvládnutých problémů $\Rightarrow$ vývoj postupuje po spirále

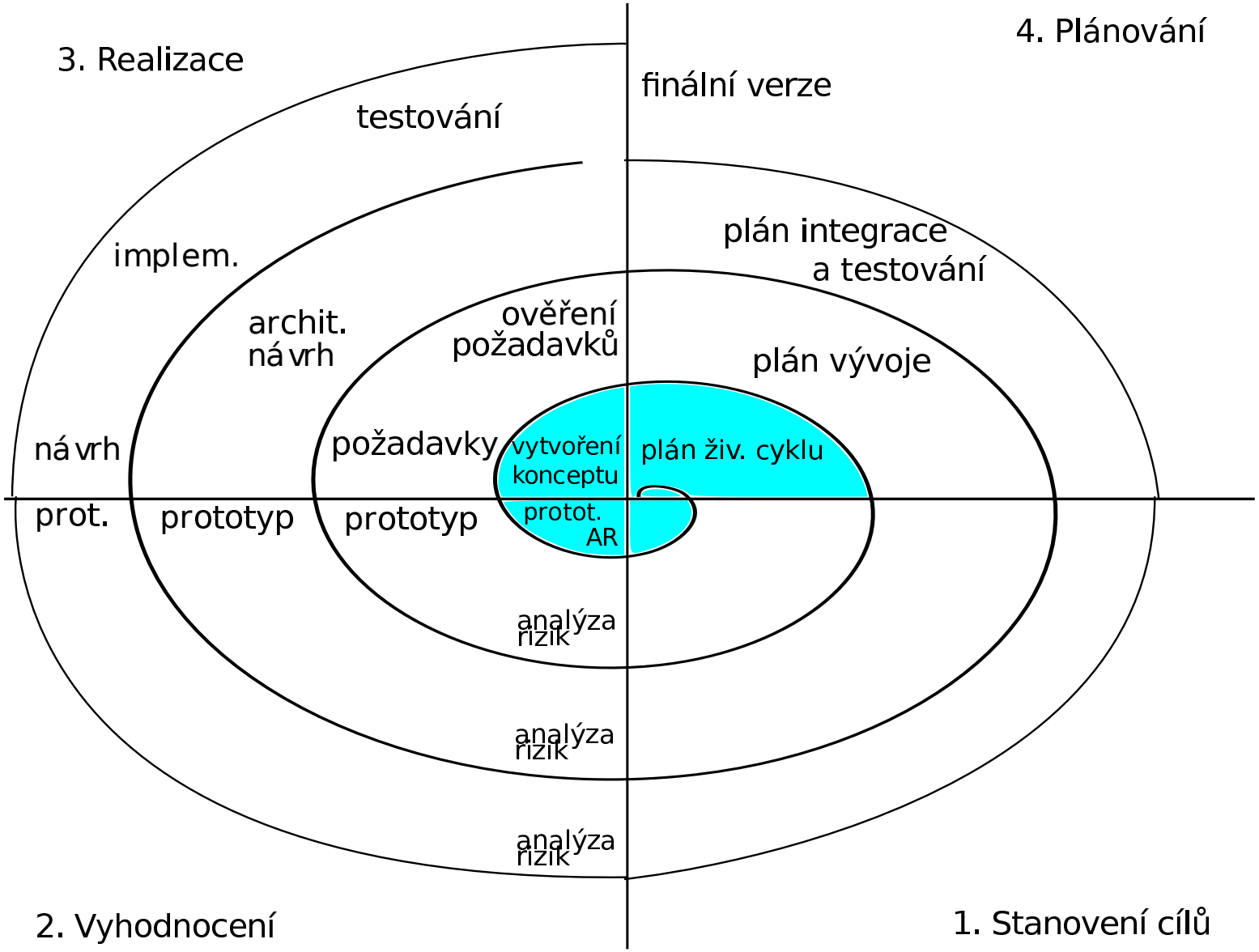
Spirálový model

Význam cyklů (*počet cyklů není pevně stanoven*)

- první: globální rizika, základní koncept vývoje, volba metod a nástrojů
- druhý: vytváření a ověřování specifikace požadavků
- třetí: vytvoření a ověření návrhu
- čtvrtý: implementace, testování a integrace

- pokud jsou výsledky cyklu nedostatečné (chyby ve specifikaci, návrhu apod.), stejný cyklus se zopakuje s upraveným plánem

Spirálový model



Spirálový model

Význam kvadrantů

- Q1 – cíle cyklu
  - definice cílů: např. výkonnostní požadavky, funkcionální požadavky, vytvoření architektury
  - alternativy: různé způsoby řešení cílů
  - omezující podmínky: např. cena, plán projektu
- Q2 – vyhodnocení
  - ověření splnitelnosti stanovených cílů
  - analýza rizik, prototypování, simulace, ...
- Q3 – realizace
  - realizace cílů cyklu
- Q4 – plánování
  - plánování (úprava plánů) následujícího cyklu, stanovení jeho průběhu (kdo kdy co)

Spirálový model

Analýza rizik: Jaké jsou cíle

- zjistit možná ohrožení průběhu projektu
- připravit reakce na tato rizika
- rizika se identifikují a analyzují v každé fázi vývoje
  $\Rightarrow$ včasné vyloučení nevhodných řešení

Analýza rizik: Jaká mohou být rizika

- projektová: odchod lidí, snížení rozpočtu, ...
- technická: neznámé technologie, selhání hardwaru, ...
- obchodní: špatný odhad zájmu, ...

Spirálový model

Mezníky (Milestones)

- Life Cycle Objectives (LCO): po 2. cyklu
  - vyhodnocení záměrů/cílů projektu, rozhodnutí o pokračování projektu
  - *všechny požadavky podchyceny, stejné chápání požadavků*
  - *cena, plán, priority apod. odpovídají záměrům*
  - *jsou identifikována rizika a procesy pro jejich odstranění/zmírnění*
- Life Cycle Architecture (LCA): po 3. cyklu
  - vyhodnocení výběru architektury, řešení závažných rizik, ...
  - *požadavky a architektura jsou stabilní*
  - *osvědčené postupy testování a vyhodnocování*
- Initial Operation Capability (IOC): po 4. cyklu
  - systém je připraven na distribuci pro uživatelské testování
  - *stabilní verze schopná testového nasazení u zákazníka/uživatele*
  - *srovnání plánovaných a skutečných výdajů a použitých zdrojů*

Spirálový model

Silné stránky

- komplexní model vhodný pro složité projekty
- chyby a nevyhovující postupy jsou odhaleny dříve (analýza rizik)
- nezávislost na metodice či strategii návrhu/implementace/testování

Slabé stránky

- závislý na analýze rizik – musí být prováděna na vysoké odborné úrovni
- vyžaduje precizní kontroly výstupů, zkušené členy týmu
- software je k dispozici až po posledním cyklu (*lze vyřešit použitím většího počtu implementačních cyklů*)
- problematické je přesné plánování termínů a cen

Metodika Rational Unified Process – RUP

Co je RUP

- výsledek výzkumu zkušeností řady velkých firem koordinovaný firmou Rational Software, 1997
- první kniha *The Unified Software Development Process*, 1999
- od roku 2003 je Rational Software součástí IBM
- spíše než konkrétní metodika je chápán jako rozšiřitelný framework, který by měl být uzpůsoben organizaci či projektu (*customizable framework*)
- komerční produkt, dodávaný společně s nástroji

Metodika Rational Unified Process – RUP

Základní vlastnosti

- objektově orientovaná metodika
- přístupy řízené případy užití (*use-case-driven approach*)
- návrh softwarového systému je vizualizován
  - UML, ...
- iterativní vývoj
  - verze systému, po každé iteraci spustitelný kód
- průběžná kontrola kvality produktu
  - objektivní měření, metriky, ...
- snaha o využívání existujících komponent
- věnuje se všem otázkám procesu tvorby softwaru (*kdo*, *co*, *kdy* a *jak*)

Metodika RUP – základní elementy

Pracovníci a role (*kdo*)

- chování je popsáno pomocí činností
- důležitá je *role*: analytik, návrhář, ...

Činnosti – *Activities* (*jak*)

- jasně definovaný účel s definovaným výsledkem (meziprodukt)

Meziprodukty – *Artifacts* (*co*)

- výsledky projektu (činností)
- model, dokument, zdrojový kód, ...

Pracovní procesy – *Workflows* (*kdy*)

- definuje posloupnost činností a interakce mezi pracovníky
- RUP definuje 6 klíčových a 3 pomocné procesy

Metodika RUP – pracovní procesy

Klíčové procesy

- *Business Modeling*: obchodní požadavky, popis procesů, ...
- *Requirements*: další požadavky, model rozhraní, scénářů, ...
- *Analysis and Design*
- *Implementation*
- *Testing*
- *Deployment*

Pomocné procesy

- *Project Management*
- *Configuration Management*
- *Environment*: administrace, školení vývojářů, ...

Metodika RUP – vývojový cyklus

Vývojové cykly

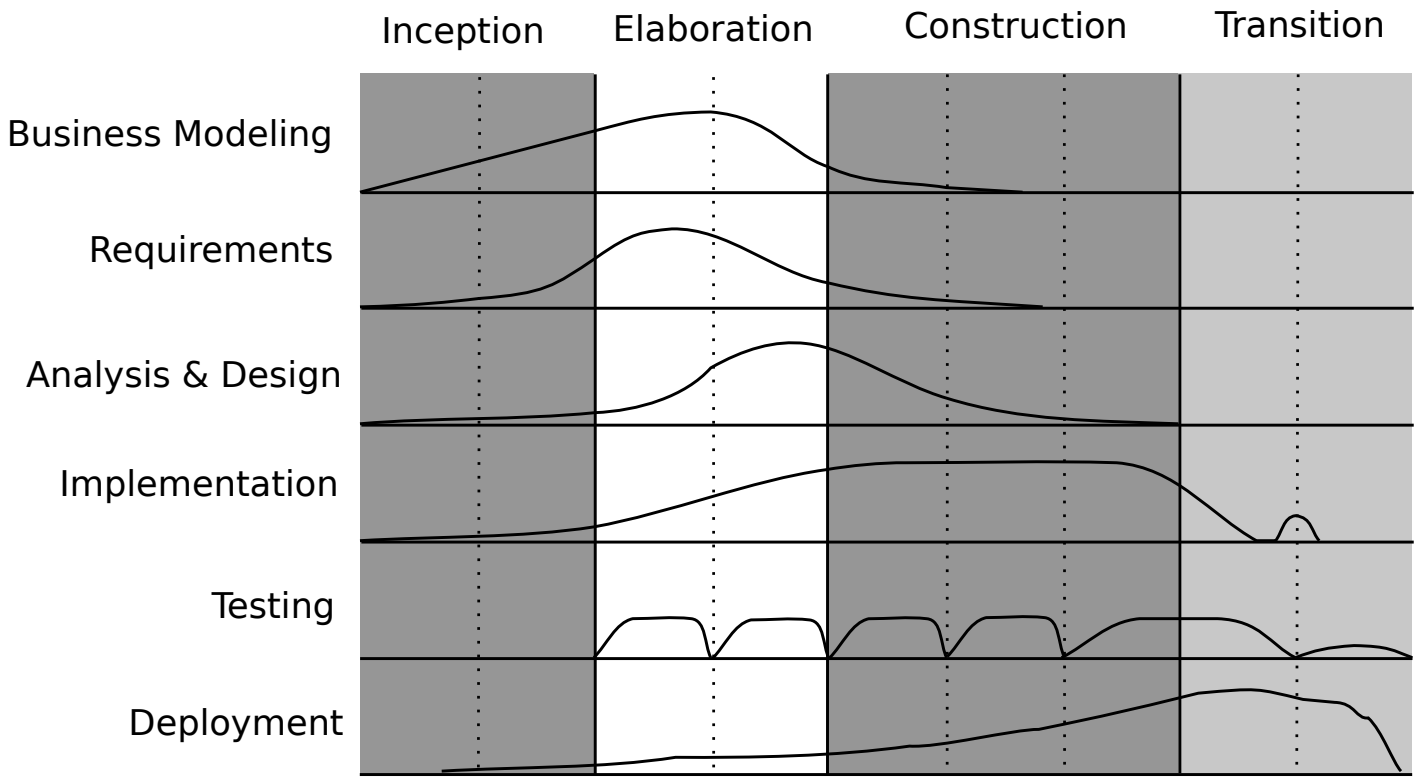
- *Initial Development Cycle* – výsledkem je funkční softwarový produkt
- *Evolution Cycles* – další vývoj, verze, ...

Základní cyklus

- je rozdělen na čtyři fáze
  - zahájení (*inception*) ... 10%
  - projektování (*elaboration*) ... 30%
  - realizace (*construction*) ... 50%
  - předání (*transition*) ... 10%
- každá fáze je rozdělena na iterace
  - délka jedné iterace 2 až 6 týdnů

Metodika RUP – model životního cyklu

- iterativní model jednoho vývojového cyklu
- etapy (pracovní procesy) se překrývají (souběžné provádění)



Metodika RUP – vývojový cyklus

Fáze cyklu

- zahájení (*inception*)
  - rozsah projektu, náklady, základní rizika, základní UC, ...
  - 1-2 iterace
- projektování (*elaboration*)
  - plánování, specifikace požadavků, architektura, analýza rizik, ...
  - zpravidla 2 iterace, může až 4 iterace
- realizace (*construction*)
  - kompletace analýzy a návrhu, implementace, hodnocení výstupů, ...
  - 2-4 iterace
- předání (*transition*)
  - dodání, školení, podpora při zavádění, ...
  - alespoň 2 iterace (betaverze, plná verze)

Metodika RUP – mezníky (Milestones)

Mezníky (Milestones)

- převzaté ze *Spirálového modelu*
  - **Life Cycle Objectives**
    vyhodnocení záměrů/cílů projektu, rozhodnutí o pokračování projektu
  - **Life Cycle Architecture**
    vyhodnocení výběru architektury, řešení závažných rizik, ...
  - **Initial Operation Capability**
    systém je připraven na distribuci pro uživatelské testování

\+ **Product Release**

- rozhodnutí, zda byly záměry projektu splněny a zda pokračovat v dalším vývojovém cyklu
- *je uživatel spokojen*, *odpovídají náklady plánu*, ...

Metodika RUP – zhodnocení

Silné stránky

- robustní, vhodný pro velkou škálu projektů
- iterativní přístup, včasné odhalení rizik, správa změn, ...
- detailní propracovanost
- vazba na UML

Slabé stránky

- detailní propracovanost: u menších projektů značná zátěž na zkoumání metodiky; vývoj může postrádat efektivitu
- komerční produkt (obsahuje hodně podpůrných nástrojů)

Metodika Rapid Application Development

Vlastnosti RAD

- James Martin, *Rapid Application Development*, 1991
- rychlý iterativní vývoj prototypů
- funkční verze jsou k dispozici dříve než u předchozích přístupů
- intenzivní zapojení zákazníka/uživatele do vývojového procesu
- zaměřuje se na splnění *business potřeb* (potřeby a požadavky zákazníka), technologické a inženýrské kvality mají menší důležitost
- určen pro menší až středně velké projekty

Fáze (přehled)

- Plánování: rozsah projektu, omezení, systémové požadavky, ...
- Návrh: modelování, prototypování, využívání CASE nástrojů, ...
- Provedení: pokračování návrhu, kódování, integrace, testování, ...
- Uzavření a nasazení: příprava dat, finální testování, přechod zákazníka na nový systém, zaškolení uživatelů, ...

Rapid Application Development (RAD)

Silné stránky

- flexibilita, schopnost rychlé změny návrhu podle požadavků zákazníka
- více projektů splňuje termíny a ceny (úspora času, peněz a lidských zdrojů)
- vyšší kvalita zpracování *business potřeb* (prototypování)

Slabé stránky

- nižší kvalita návrhu, problém s udržovatelností
- flexibilita vede k menší míře kontroly nad změnami
- projekt může skončit s více požadavky, než je nutné (problém s udržovatelností)

Další přístupy k procesu vývoje softwaru

Unified Software Development Process (zjednodušeně UP)

- stejné principy a myšlenky jako RUP, není komerční, nenabízí nástroje
- není tak detailně rozpracována, např. pouze 5 pracovních procesů

Modifikované verze vodopádu

- možnost prolínání etap
- vodopád s podprojekty
- ...

Agilní přístupy (metodiky)

- skupina metodik s odlišným přístupem k procesu tvorby softwarového produktu

Heavyweight a Agilní metodiky

Heavyweight methods

- častá kritika „byrokratizace“ metodik – příliš mnoho aktivit, které jsou předepisovány, způsobuje snížení efektivity celého procesu vývoje
- člen vývojového týmu sleduje přesně postup, krok po kroku

Lightweight methods

- nová skupina metodik, dnes nazývána **agile methods** (agilní metodiky)
- kompromis mezi chaotickým přístupem bez procesů (žádná metodika) a přístupem s mnoha procesy (heavyweight metodiky)
- definují základní rámec vývoje, termíny (*mile-stones*), předpokládané výstupy, techniky, ...
- *agilní* = *čilý, aktivní* $\Rightarrow$ člen vývojového týmu používá procesy *aktivně*, tj. sám přizpůsobuje procesy a techniky potřebám projektu a týmu

Heavyweight a Agilní metodiky

Srovnání vlastností

	Heavyweight	Agilní
přístup	**prediktivní**	**adaptivní**
velikost projektu	velká	malá
velikost týmu	velká	malá (kreativní)
styl řízení	centralizovaný	decentralizovaný
	příkaz-kontrola	vedení-spolupráce
dokumentace	**velký objem**	**malý objem**
zdůraznění (důraz na)	**process-oriented**	**people-oriented**
fixní kritéria	*funkcionalita*	*čas a zdroje*
proměnná kritéria	*čas a zdroje*	*funkcionalita*

Predikovatelnost procesu vývoje

Prediktivní přístupy

- plánují velké části softwarových procesů velmi detailně pro dlouhý časový úsek
- projekty vyžadující mnoho procedur, času, velké týmy a **stabilní požadavky**

Predikovatelnost procesu vývoje

Prediktivní přístupy

- plánují velké části softwarových procesů velmi detailně pro dlouhý časový úsek
- projekty vyžadující mnoho procedur, času, velké týmy a **stabilní požadavky**

Problém většiny projektů je,
že se požadavky neustále mění.

Predikovatelnost procesu vývoje

Prediktivní přístupy

- plánují velké části softwarových procesů velmi detailně pro dlouhý časový úsek
- projekty vyžadující mnoho procedur, času, velké týmy a **stabilní požadavky**

Predikovatelnost procesu vývoje

- dobrá predikovatelnost procesu vývoje
  - projekty s jasnými a stabilními požadavky
  - např. projekty NASA, ...
- špatná predikovatelnost procesu vývoje
  - projekty s požadavky, které se v čase mění
    - změna okolních podmínek, účelu softwaru...
    - zákazník si požadavky ujasňuje v průběhu vývoje
  - business projekty

Predikovatelnost procesu vývoje

Prediktivní přístupy

- plánují velké části softwarových procesů velmi detailně pro dlouhý časový úsek
- projekty vyžadující mnoho procedur, času, velké týmy a **stabilní požadavky**

Jak na těžko predikovatelné projekty?

Adaptivní přístupy k procesům vývoje

Adaptivní přístupy

- plánují s přiměřenou mírou detailu
- plány se v průběhu procesu vývoje revidují
- jak řídit adaptivní procesy? $\Rightarrow$ iterativní přístup

Iterativní přístup a plánování procesů

- jedna iterace většinou zahrnuje základní etapy, může se měnit podle zvolené metodiky
- v první iteraci se provádí plánování procesů, tento plán se v dalších iteracích upravuje podle reálného stavu
- otázka délky iterace (týdny, měsíce, ...), určení mile-stones

Process-oriented přístupy

Předpoklady

- lidé jsou zdroje, které jsou dostupné v několika rolích: analytik, programátor, tester, manažer, ...
- procesy by měly fungovat za všech okolností (změna týmu, ...)

Process-oriented přístupy

Předpoklady

- lidé jsou zdroje, které jsou dostupné v několika rolích: analytik, programátor, tester, manažer, ...
- procesy by měly fungovat za všech okolností (změna týmu, ...)

Důsledky

- podstatná je role, nikoliv individualita lidí
  - není důležité, *jaké* analytiky máte, ale *kolik* jich máte
  - člověk je predikovatelná (a tedy jednoduše nahraditelná) komponenta vývojového procesu
- procesy by měly fungovat za všech okolností
  $\Rightarrow$ velký objem procesů, detailní specifikace procesů, velká míra režie
- za standardní prostředek komunikace se považuje dokumentace
  $\Rightarrow$ zvýšení režie

Process-oriented přístupy

Předpoklady

- lidé jsou zdroje, které jsou dostupné v několika rolích: analytik, programátor, tester, manažer, ...
- procesy by měly fungovat za všech okolností (změna týmu, ...)

Teze: člověk vykonávající práci není ten, kdo může nejlépe určit, jak tuto práci nejlépe udělat.

People-oriented přístupy

> Design and programming are human activities; forget that and all is lost.
>
> Bjarne Stroustrup, 1991

Předpoklady

- lidé nepracují konzistentně v průběhu času
  - pokud by člověk dostal každý den stejný úkol, vytvoří *podobné* výsledky, ale nikdy ne *stejné*
  - schopnost pracovního nasazení/soustředění se mění
- lidé jsou komunikující bytosti
  - fyzická blízkost – gestikulace, hlasový projev, intonace
  - otázky a odpovědi v reálném čase
- žádný proces nikdy nevytváří dovednosti (znalosti) vývojového týmu

People-oriented přístupy

> Design and programming are human activities; forget that and all is lost.
>
> Bjarne Stroustrup, 1991

Důsledky

- podstatná je individualita lidí
  - důležitá je kvalita a osobní rozvoj členů týmu
  - role člena týmu se může měnit
  - kvalitní člen týmu je hůře nahraditelný
- osobní komunikace

  $\Rightarrow$ dokumentace slouží především k dokumentačním účelům (pro potřeby revizí návrhu, údržby, ...)
- proces nevytváří dovednosti (znalosti) vývojového týmu

  $\Rightarrow$ úlohou procesů je podpora práce vývojového týmu, vymezení základních vodítek (pracovního rámce) a termínů

  $\Rightarrow$ menší objem procesů

People-oriented přístupy

> Design and programming are human activities; forget that and all is lost.
>
> Bjarne Stroustrup, 1991

Teze: člověk je kompetentní profesionál schopný rozhodovat všechny technické otázky své práce.

Agilní metodiky

Základní teze

- Minimum formálních a byrokratických artefaktů.
  - *důležitou součástí dokumentace je i zdrojový kód*
- Člen týmu je schopen rozhodovat technické otázky své práce.
  - *důraz na složení týmu a komunikaci uvnitř týmu*
  - *komunikace jako jedna z forem vývoje*
  - *techniky vyžadující komunikaci, např. párové programování*
- Ověření správnosti navrženého systému zpětnou vazbou
  - *iterativní inkrementální vývoj, časté uvolňování průběžných verzí*
  - *předložit zákazníkovi a na základě zpětné vazby upravovat*
  - *zákazník je členem vývojového týmu*

Agilní metodiky

Základní teze

- Důraz na rigorózní, průběžné a automatizované testování.
  - *zejména kvůli neustálým změnám v kódu i návrhu*
- Princip jednoduchosti
  - *návrh odráží aktuální potřeby uživatele*
  - *do systému vložíme to, co potřebujeme, když to potřebujeme*

Agilní metodiky

Metodiky označované jako agilní

- Extreme programming (XP)
- Scrum
- Crystal
- Feature Driven Development (FDD)
- Test Driven Development (TDD)
- Dynamic System Development Method (DSDM)

- Scrum of Scrums
- Scaled Agile Framework (SAFe)

- ...

Extrémní programování (XP)

Kořeny XP

- Kent Beck, Ward Cunningham
- 80. léta – Smalltalk
- 90. léta – získávání zkušeností v různých projektech, rozšiřování idejí agilního přístupu

Reference

- `http://www.extremeprogramming.org`
- *Beck, K., and Andres, C., Extreme Programming Explained: Embrace Change, 2nd ed. Addison-Wesley, 2004*

XP: Základní techniky

Přírůstkové (malé) změny

- návrh a implementace se mění v čase jen pozvolna
- uvolňování malých verzí systému (nejpodstatnější požadavky, postupně vylepšované a doplňované)

XP: Základní techniky

Přírůstkové (malé) změny

- návrh a implementace se mění v čase jen pozvolna
- uvolňování malých verzí systému (nejpodstatnější požadavky, postupně vylepšované a doplňované)

Testování

- *Co nelze otestovat, to neexistuje.*
- ke každé funkci píšeme testy, někdy i před tím, než začneme programovat
- zautomatizovaný systém testů
- jednotkové i integrační testování

XP: Základní techniky

Párové programování

- jednu věc programují vždy 2 programátoři (ale pouze 1 skutečně píše)
- ten, kdo píše, se soustřeďuje na nejlepší způsob implementace problému
- druhý se soustřeďuje na problém z globálnějšího pohledu
  bude to fungovat, jaké další testy, možnost zjednodušení, ...
- páry jsou dynamické

XP: Základní techniky

Párové programování

- jednu věc programují vždy 2 programátoři (ale pouze 1 skutečně píše)
- ten, kdo píše, se soustřeďuje na nejlepší způsob implementace problému
- druhý se soustřeďuje na problém z globálnějšího pohledu
  bude to fungovat, jaké další testy, možnost zjednodušení, ...
- páry jsou dynamické

Refaktorizace

- úprava stávajícího programu – zjednodušení, zefektivnění návrhu
- odstranění (úprava) nepotřebných částí
- změna architektury (pravidlo přírůstkové změny)
- *při refaktorizaci se nemění funkcionalita!*

XP: Základní techniky

Metriky

- důležitá součást určení kvality softwarových procesů
- např. poměr plánovaného času a skutečného času
- přiměřený počet metrik (3-4)
- pokud přestane metrika plnit svůj účel $\Rightarrow$ nahradit jinou
  např. metrika testů funkcionality se blíží 100% $\Rightarrow$ nahradit jinou s menší úspěšností
- existují pravidla udávající, kdy a jak často by se měly jednotlivé techniky používat

XP: Základní techniky

Metriky

- důležitá součást určení kvality softwarových procesů
- např. poměr plánovaného času a skutečného času
- přiměřený počet metrik (3-4)
- pokud přestane metrika plnit svůj účel $\Rightarrow$ nahradit jinou
  např. metrika testů funkcionality se blíží 100% $\Rightarrow$ nahradit jinou s menší úspěšností
- existují pravidla udávající, kdy a jak často by se měly jednotlivé techniky používat

Motivace vývojářů

- lidé lépe pracují, pokud je práce baví
- jídlo, hračky, vybavení pracoviště, ...

XP: Proces vývoje

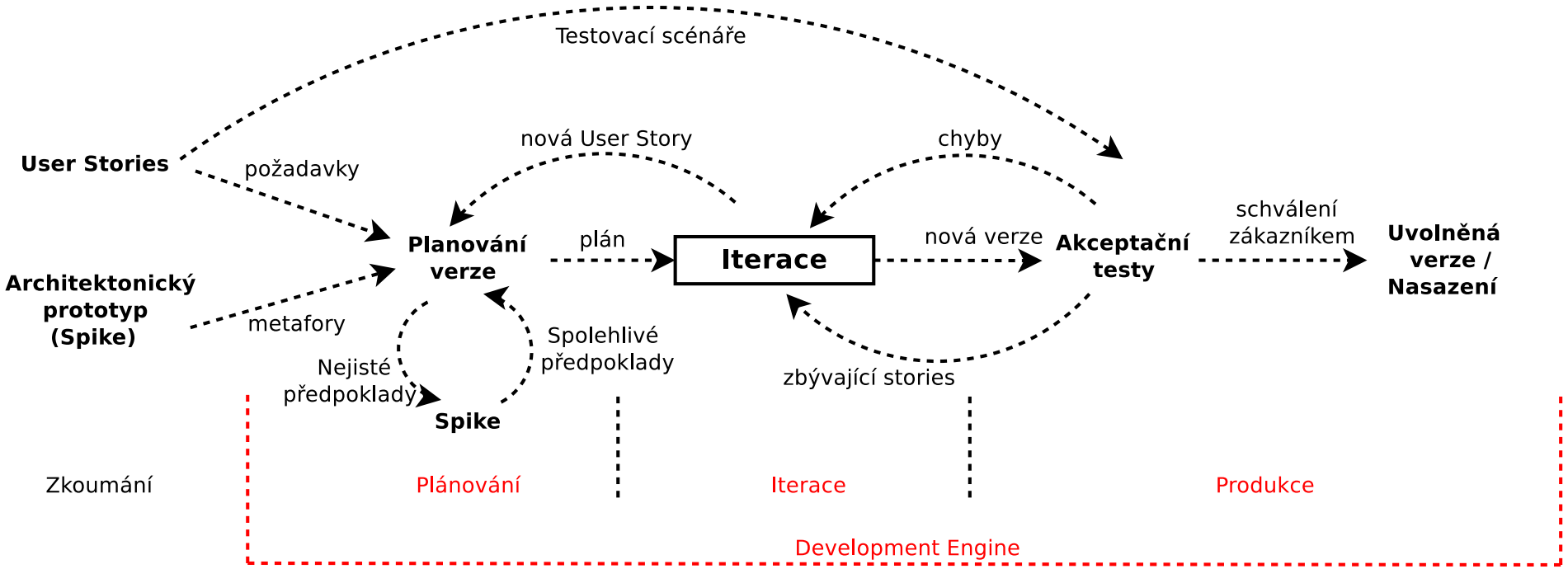
- **Zkoumání** (*Exploration*)
  - tvorba vysokoúrovňových požadavků a základního návrhu
- **Development Engine**
  - reprezentuje iterativní proces vývoje a údržby
  - realizuje vybranou množinu požadavků
    výsledkem je verze produktu (inkrement)
  - v dalších bězích realizuje zbývající požadavky
  - *během vývoje se mohou požadavky měnit*
- **Uzavření** (*Death*)
  - další vývoj projektu je nepotřebný, neúčelný nebo nemožný
  - ukončení všech formálních závazků a vazeb (finance, tým atd.)
  - vytvoření závěrečné dokumentace
  - vyhodnocení průběhu projektu – získané poznatky, *co jsme se naučili při řešení tohoto projektu*

XP: Zkoumání (Exploration)

- Utváření týmu
- Návrh počáteční množiny *User Stories*.
- *User Story*
  - definuje vlastnost systému z pohledu zákazníka/uživatele
  - je psána uživatelem terminologií problémové domény
  - zaměřuje se na cíl, podrobnosti se ujasňují během vývoje
  - ke každé story by měl být vytvořen akceptační test
    Story: Hledání a nahrazování ve velkém dokumentu musí být rychlé.
    Test: Nahrazení 1 000 výskytů řetězce o délce 4 znaky $\leq$ 700 ms.
- Tvorba systémových metafor (*Metaphor*)
  - základní (jednoduchý) návrh, třídy, ...
  - rychlé pochopení pro každého člena týmu
- Tvorba prototypů (*Spikes*)

XP: Development Engine

- výběr množiny požadavků (*Stories*) a jejich realizace
- výsledkem je verze systému (inkrement)
- v dalším běhu *Development Engine* realizujeme dosud nezpracované (příp. nové) požadavky (*Stories*)



XP: Development Engine

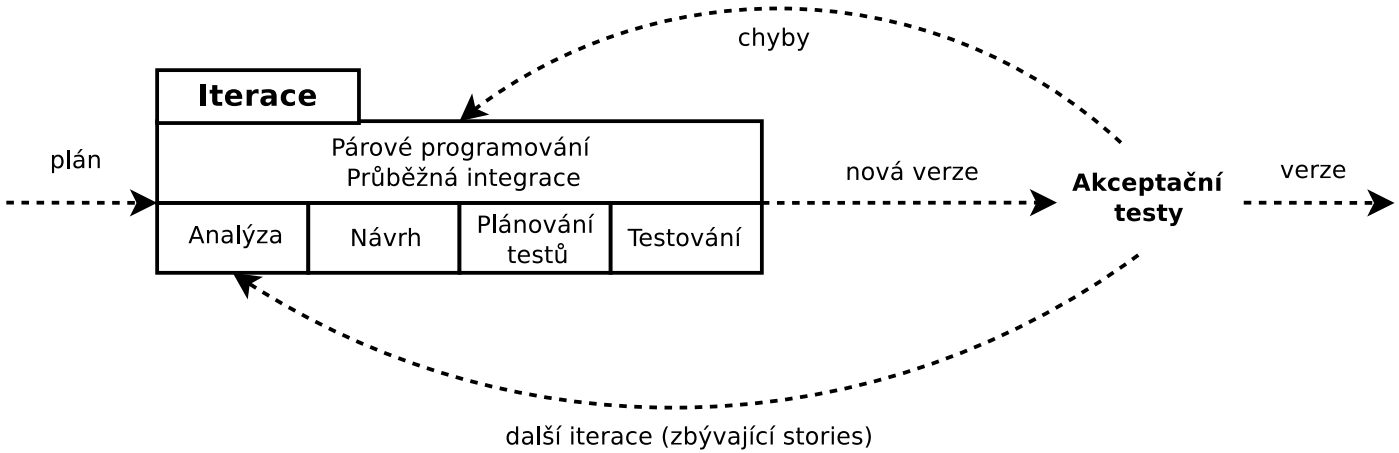
Plánování (Planning)

- Odhad vývojového času
  - odhad času pro vývoj každé *user story*
  - *user stories* vyžadující více než 3 týdny jsou rozděleny na menší
  - *user stories* vyžadující méně než 1 týden jsou sloučeny
  - při odhadu jsou využívány prototypy (*spikes*)
- Nastavení priorit
  - zákazník seřadí *user stories* podle priorit
- Plánování první verze / dalších verzí
  - výběr množiny *user stories* k implementaci
  - shoda na datu uvolnění
  - rozhodnutí o délce iterace (1-3 týdny); je stejná pro všechny iterace

XP: Development Engine

Iterace (Iterations to Release)

- v každé iteraci se vybere část *user stories* k implementaci nebo nápravě při selhání akceptačních testů
- výběr *stories* a plánování iterace bere v úvahu dosavadní rychlost vývoje
- identifikace úloh
  - rozdělení *user stories* na jednotlivé úlohy
  - každá úloha by měla být dokončena během 1 až 3 dnů
- realizace s využítím technik XP



XP: Development Engine

Produkce (Productionizing)

- Verifikace a validace
  - testování uvolněné verze (*release*)
  - regresní testy, akceptační testy
  - nalezené chyby jsou odstraněny v rámci *Iterations*
- Nasazení
  - nasazení verze do produkčního (uživatelského) prostředí
  - obsahuje standardní integraci, ladění, zaučování, dokumentace, . . .
  - ladění a stabilizace je chápáno jako vývojová aktivita a probíhá v rámci krátkých iterací (týden) ve fázi *Iterations*

XP: Údržba (Maintenance)

- *user stories* jsou implementovány a systém je používán jako celek
- změny a úpravy se prováději v rámci *development enginu*
- malé změny se integrují do systému běžícího v provozu
- nové požadavky jsou zpracovány stejným způsobem jako běžné požadavky, tj. jsou vyjádřeny pomocí *user stories* a implementovány v *development enginu*
- fáze údržby běží, dokud existují *user stories* nebo se očekávají v budoucnu

XP: Vyhodnocení

Silné stránky

- iterativní inkrementální proces
- proces se *ladí* na základě zpětné vazby
- požadavky se *ladí* během celého vývoje
- průběžná integrace
- zapojení uživatelů
- vývoj založený na testování

Slabé stránky

- nepředepisuje modely pro návrh, často se od *User Stories* a *Metaphor* přechází na implementaci
- hůře akceptovatelný pro vývojáře – vyžaduje striktní dodržování základních principů a procesů

XP: Collective-Code-Ownership

Podstata

- každý člen týmu má možnost (i povinnost) ovlivňovat kód (nová funkcionalita, odstranění chyb, refaktorizace)
- snižuje riziko, že nepřítomnost jednoho vývojáře zpomalí práci
- podporuje pocit odpovědnosti každého vývojáře za kvalitu celku

Základní techniky

- jednotný styl programování – zlepšení komunikace
- účastnit se postupně všech prací – znalosti o všech částech systému
- párové programování
- *test-driven development*
  - ke každému kódu musí existovat jednotkové testy (*unit tests*), které se sdružují do sad (*test suites*)
  - při každé změně (úprava, integrace nového kódu) musí být provedena (automatizovaně) sada testů
- *průběžná integrace* (*continuous integration*)

XP: Průběžná integrace

Co je průběžná integrace

- automatizované a reprodukovatelné sestavování (*build*)
- obsahuje automatizované testování, které probíhá mnohokrát za den
- umožňuje průběžně integrovat změny a tím redukovat problémy s integrací

Základní procesy průběžné integrace

- integrace zdrojového kódu
  - sdílené repozitáře, ...
- automatizovaná správa sestavování (*build management*)
  - sestavování se provádí často, několikrát za den
  - sestavení se provádí při změně kódu, v naplánovaném čase, ...
  - vývojář musí být informován o výsledku
- automatizované ověřování (testování)
  - po sestavení je nutno ověřit, že nová verze splňuje všechny testy