Moderní modely životního cyklu vývoje software

Z FITwiki

Moderní modely životního cyklu vývoje software (iterativní model životního cyklu; MDA, agilní vývoj; Unified Process (UP) - fáze, jejich cíle a činnosti v nich).

Obsah

Základní pojmy

SW se nevyrábí, ale implementuje

Úrovně použití SW v organizacích k rozhodování

- **operační** (každodenní rozhodování)
- **taktické** (obvyklé rozhodování)
- **strategické** (dlouhodobé rozhodování)

Životní cyklus SW

Životní cyklus software je reprezentace softwarového procesu. Definuje fáze, kroky, aktivity, metody, nástroje a očekávané výstupy softwarového projektu. Životní cyklus modelujeme pomocí modelu životního cyklu.
Má dvě základní období:

- **Období postupného zavádění** (phasing in)
- **Postupného vyřazení z provozu** (phasing out)

Fáze životního cyklu SW

1 - Analýza požadavků

- získání požadavků na SW od zákazníka
- více viz Získávání_a_modelování_požadavků

2 - Návrh

- na základě požadavků je proveden návrh SW
- více viz Logická_architektura_software a Objektově-orientovaný_návrh

3 - Implementace

- implementace návrhu
- součástí je i **testování a ladění**

Testování
: veškeré aktivity, jejichž cílem je odhalení chyb

- posouzení (např. techniky procházení či inspekce)
- spuštění proveditelného kódu
  - testování ze specifikace (black-box) (také označované jako funkční) - při návrhu testu vychází ze specifikace testovaného programu (co má umět)
  - testování z kódu (white-box) - vychází ze znalosti zdrojového kódu

Ladění
: aktivity zaměřené na odstranění chyb

4 - Integrace a nasazení

Integrace
: spojení jednotlivých částí SW do jednoho celku

Integrační testování

- **shora dolů**

  postupně přidáváme jednotlivé komponenty od kořene hierarchie. Vzniká ale problém, že testovaná část může používat komponenty, které v ní nejsou obsaženy. Použijí se náhražky (stubs)

- **zdola nahoru**

  postupuje naopak v hierarchii komponent od listů ke kořenům. I v tomto případě je třeba pro účely testování nahradit chybějící část, tentokrát komponenty, které testovanou část používají (řídí). Pro náhradu se v tomto případě používá označení driver.

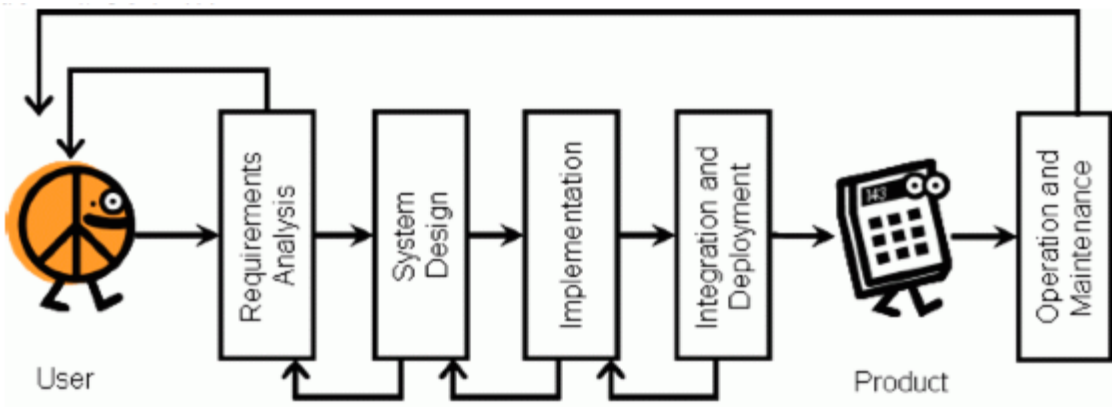
5 - Provoz a údržba

Modely životního cyklu SW

Životní cykly a jejich modely lze rozdělit do dvou základních skupin:

- vodopád se zpětnou vazbou (označovaný také jako klasický)
- iterativní s inkrementy

Vodopád



- Základní nejstarší model, v praxi není příliš vhodný (nedokáže se přizpůsobit)
- Jednotlivé fáze následují po sobě a jsou jednoznačně oddělené
- Každá fáze plně dokumentovaná
- Fáze může poskytnout zpětnou vazbu předchozí fázi (tj. návrat k předchozí fázi a její přepracování)

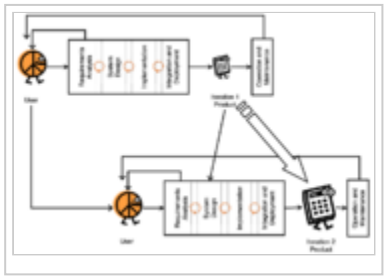
Nevýhody:

- nedokáže se přizpůsobit změnám
- vše se odhaduje na začátku
- slabá zpětná vazba
- nelze paralelizovat

Vodopád s překrytím
: modifikace klasického vodopádu - jednotlivé fáze se částečně překrývají

Vodopád s prototypováním
: nejprve je vytvořen prototyp, na základě prototypu se upřesní požadavky a teprve potom je spuštěn ostrý vývoj (prototyp se buď zcela zahodí nebo se použije jako základ)

Iterativní inkrementální modely



Iterativní
: opakování procesu s cílem zlepšit produkt (iterace by měly být krátké - dny až týdny). Typicky iterace pevné nebo časově omezené délky, miniprojekt.

Inkrementální/evoluční
: každé opakování přináší vylepšenou/rozšířenou verzi

Sestavení (angl. build)
: výsledek iterace: otestovaný, integrovaný, spustitelný kód.

Výhody:

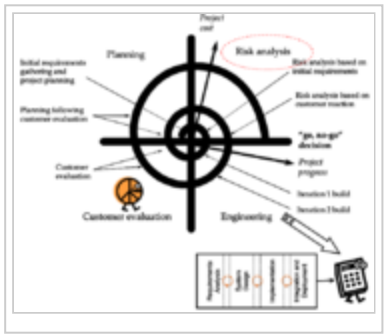
- průběžné plánování
- spolehlivější řízení projektu
- odhady se postupně zpřesňují
- požadavky se mohou mezi iteracemi zpřesňovat

VŠECHNY NÁSLEDUJÍCÍ MODELY JSOU TAKÉ ITERATIVNÍ...

Spirálový model

- iterativny inkrementralny model

Je ve skutečnosti rámcem nebo metamodelem, který může obsahovat jiné modely životního cyklu. Model se prezentuje v podobě spirály, která prochází čtyřmi kvadranty:



1. Planovanie
2. Analyza rizik - pokracovat alebo nie?
3. Engenieering (analyza poziadavkov, navrh, implementacia, integracia a nasadenie)
4. zhodnotenie zakaznikom

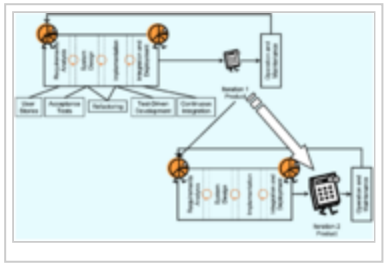
Modelem řízené architektury

Model-driven architecture (MDA)

- rámec životního cyklu
- vývoj software jako posloupnost transformací z formální specifikace
- je vytvořen model který se pomocí CASE nástrojů převede na implementaci
- každá transformace by mela zajišťovat nebo umožnit verifikovat, že výstup odpovídá vstupu
- výrazně automatizován

Computer-aided software engineering (CASE)
použití SW při vývoji SW (generování kódu, datové modelování, refaktorování, UML)

Agilní vývoj



- rychlá reakce na měnící se podmínky
- klade důraz na přizpůsobivost a spolupráci se zákazníkem po celou dobu vývoje

Hlavní zásady (definované v tzv. Agilním manifestu):

- Jednotlivci a interakce před procesy a nástroji
- Fungující software před úplnou dokumentací
- Spolupráce se zákazníkem před vyjednáváním smlouvy
- Reagování na změny před dodržováním plánu projektu

Některé postupy:

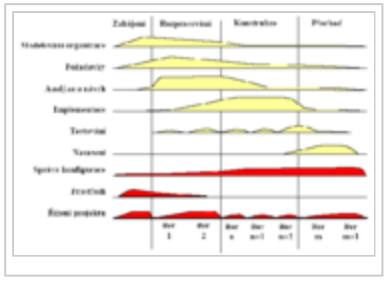
- Přejímací testy (acceptance test) - testy ověřující, že zákazník dostává to co chtěl
- Vývoj řízený testy (Test-driven development - TDD) - nejprve testy pak implementace
- Programování ve dvojicích (kód vždy tvoří dva lidé)
- Kolektivní vlastnictví kódu (všichni jsou zodpovědní za celý kód)
- user stories - místo klasické analýzy požadavků uživatel vylíčí co by rád aby systém uměl (pár vět, běžný jazyk, nestrukturované)
- Refaktorizace (refactoring) - vylepšování kódu bez změny funkčnosti

Agilní modelování

- náčrtky na tabuli/papír, souběžné vytváření několika modelů (např. interakce a tříd)
- modely se vytvářejí až když jsou potřeba, jen tak přesné, jak je pro daný účel potřeba
- CRC štítky - přiřazení zodpovědností třídám

[1] (http://nb.vse.cz/~buchalc/clanky/objekty2005.pdf)

Unified Process (UP)



Doporučuju přečíst [[2] (http://objekty.vse.cz/Objekty/RUP)]

UP je generický proces vývoje SW - vždy musí být nejdříve adaptován pro organizaci a každý projekt (vychází z Rational Unified Process - RUP)

3 základní axiomy

-
  - proces řízený požadavky a riziky (význam analýzy rizik)
  - staví na robustní architektuře systému (architecture centric) - používá pohled 4+1 z UML
  - iterativní a inkrementální

- požadavky se považují za měnící se
- **celý projekt se dělí na několik fází zakončených milníky v rámci jednotlivých fází pak probíhají iterace** (jedna nebo několik iterací na fázi dle potřeby)

Best practices v UP

- Iterativní vývoj SW
- Správa požadavků
- Architektura založená na komponentách
- Vizuální modelování (UML)
- Ověřování kvality SW (funkcionalita, spolehlivost, výkonnost)
- Řízení změn SW

Činnosti v iteracích (v podstatě klasický vodopád)

- Získávání požadavků (Requirements)
- Analýza (Analysis)
- Návrh (Design)
- Implementace (Implementation)
- Testování (Testing)

Disciplíny ve fázích UP (statický rozměr)

- Modelování organizace (Business modelling)
- Požadavky
- Analýza a návrh
- Implementace
- Testování
- Nasazení
- Podpůrné:
  - Správa konfigurace a změn
  - Vývojové prostředí (podpůrné nástroje)
  - Řízení projektu

Fáze UP (dynamický rozměr)

Zahájení (Inception)

Cíle:

- stanovení proveditelnosti
- určení přínosu projektu (vytvoření business případu)
- určení rozsahu (dle nejpodstatnějších požadavků)

- identifikace kritických rizik
- **Na konci fáze zahájení se rozhoduje o dalším pokračování projektu**

Zaměření:

- Požadavky
- Analýza

Výstupy:

- Vize projektu
- Předběžný ekonomický plán projektu
- Plán vývoje

Milník:

- Stanovené cílů projektu

Rozpracování (Elaboration)

Cíle:

- vytvoření spustitelné verze celkové architektury
- zpřesnění odhadů rizik
- definice atributů kvality
- zachytit případy užití 80% funkčních požadavků
- vytvořit detailní plán fáze konstrukce
- určit potřebné zdroje (náklady, čas, vybavení, lidí, ...) a formulovat nabídku

Zaměření:

- Požadavky - zpřesnění požadavků a rozsahu
- Analýza - určení, co se bude konstruovat
- Návrh - vytvoření stabilní architektury
- Implementace - vytvoření verze architektury
- Testování - testování verze architektury

Milník:

- Spustitelná verze architektury

Konstrukce (Construction)

Cíle:

- ukončení sběru požadavků, analýzy a návrhu
- údržba integrity architektury
- připravit verzi připravenou k nasazení

Zaměření:

- Požadavky - odhalení posledních požadavků
- Analýza - dokončení modelů analýzy
- Návrh - dokončení modelů návrhu
- **Implementace** - zajistit provozní způsobilost pro nasazení
- Testování - testování způsobilosti pro nasazení

Milník:

- Počáteční provozní způsobilost

Zavedení (Transition)

Cíle:

- opravy chyb
- příprava pracoviště uživatele k nasazení nového SW
- přizpůsobení SW prostředí (nastavování)
- úprava SW v případě problémů
- tvorba dokumentace a manuálů
- zaškolení uživatelů
- provedení závěrečné revize

Zaměření:

- Požadavky - nic
- Analýza - nic
- Návrh - případná úprava návrhu při problémech v průběhu beta testování
- Implementace - přizpůsobení SW pracovišti uživatele a odstranění problémů vzniklých při beta testování
- **Testování** - beta-testování a přejímací testy na pracovišti uživatele

Milník:

- Přodukční verze

Test Driven Developement

- vyvoj riadeny testom
- najskor sa napise test, potom sa implementuje,
- iteracie, pokial testy niesu ok

Vyhody:

- unit testy sa naozaj napisu
- ujasnenie detailneho rozhrania a chovania
- prokazatelna, opakovana a automatizovana verifikace
- podpora v sw (IDE,..)

Kategorie: Státnice MIS | Analýza a návrh informačních systémů | Státnice AIS | Státnice 2011

- Stránka byla naposledy editována 4. 6. 2016 v 23:30.