50. MODERNÍ MODELY ŽIVOTNÍHO CYKLU VÝVOJE SW

- SW se v organizacích využívá pro řízení a také rozhodování, ať už se jedná o každodenní a rychlé rozhodování, nebo dlouhodobé strategické rozhodování a plánování
- životní cyklus SW udává fáze, kroky a milníky, kterými SW prochází v průběhu svého "života"
- definuje fáze, kroky, milníky, nástroje, výstupy projektu atd..
- životní cyklus SW se modeluje model životního cyklu SW (životní cyklus vývoje)
- má dvě základní období
  - zavádění produktu (od začátku plánování a požadavku na SW po jeho kompletní zavedení)
  - vyřazování projektu (od zavedení až do vyřazení) - zajímavé je, že jakmile se produkt (SW) dostane do provozu a nastane jeho udržování a provoz tak je ve fázi vyřazování

Fáze životního cyklu vývoje SW

1. **Analýza požadavků** - analytik zjišťuje co klient či potenciální uživatel potřebuje a a vlastně chce (náročné - mnohdy sám neví) a při použití UML mohou být výstupem třeba Use Case diagramy

2. **Návrh** - z uvedených požadavků se vytvoří návrh systému (SW) ať už návrh modulů a komunikace mezi nimi, datový návrh (DB a tříd), nebo návrh složek, návrh testů, rozhraní atd... → výstupem mohou být různé diagramy (UML)

3. **Implementace** - návrh se implementuje (naprogramuje se funkční SW nebo jeho část či verze)
   - součástí je i testování (odhalování chyb a automatické testování atd) a ladění (oprava chyb odhalených testy)
   - testování
     - z kódu (white box) - jestli funguje kód (my)
     - ze specifikace (black box) - jestli to splňuje specifikaci a je jedno jaký to má kód
   - zdola nahoru - od nejmenších částí a modulů - musí se simulovat ty řídící nad nimi (Dáša)
   - shora dolů - od nejvyššího a nejhlavnějšího modulu - ty nižší jemu podřízené se musí simulovat
   - ověření správnosti SW se dělá i validací (jestli se to dělá co se od toho chce) ↳ zda vyhovuje požadavkům zákazníka a verifikací (jestli se to dělá správně) ↳ zda vyhovuje modelu

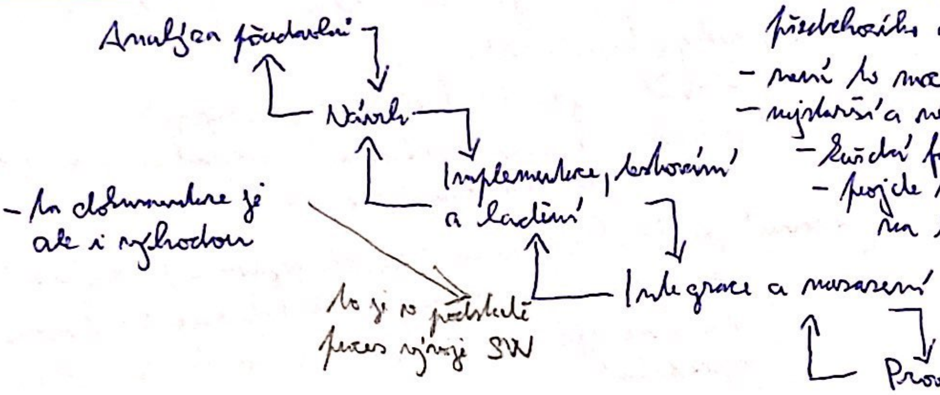
4. **Integrace a nasazení**
   - SW se integruje - sestaví - do jednoho funkčního celku z menších vyvíjených modulů
   - nasadí se do provozu a školení atd...
   - integrační testování

5. **Provoz a údržba**
   - provozují se a udržují - opravují se nalezené a reportované chyby, reagují se na různé změny a přizpůsobuje se SW a dovyvíjí se vylepšení

Modely životního cyklu vývoje SW

- vodopádový se zpětnou vazbou
- iterativní – v podstatě vodopádový, ale iteruje a na konci každé iterace je nějaký do jisté míry funkční celek

Vodopádový model se zpětnou vazbou



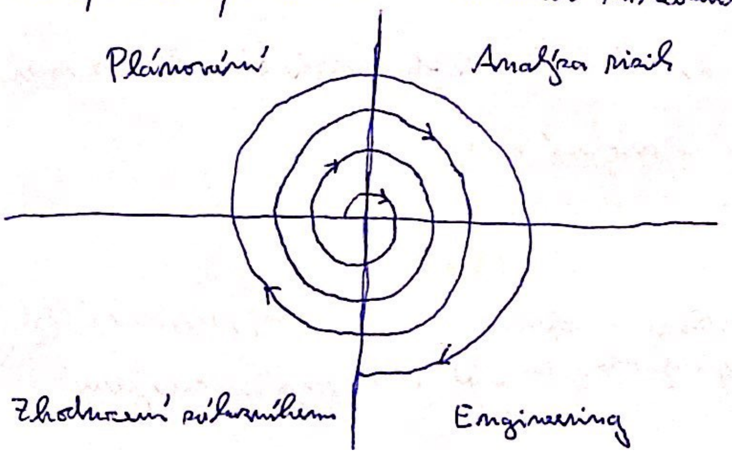
- z každého kroku se dá vrátit zpět do předchozího a tam se dá přepracovat
- není to moc přizpůsobivé změnám
- nejstarší a nejpoužívanější
  - každá fáze musí být dokonale zdokumentována
  - projede to celé 1x a na konci je hotový produkt a blbě se reflektují změny ⇒ byrokracie
  - když se třeba špatně udělá analýza tak je na konci problém

Iterativní model životního cyklu vývoje SW

- krátké a velmi rychlé iterace
- v podstatě vodopádový model který ale velmi rychle iteruje a na konci každé iterace je dostupný z části funkční produkt a vidíme co je blbě a co dobře a může se to reflektovat v další iteraci ⇒ přizpůsobuje se změnám a když se třeba analýza udělá špatně tak se to dá opravit v další iteraci
- ve vodopádovém spustitelná verze až úplně nakonec × tu v každé iteraci
- každá iterace je 1 miniprojekt – vodopád jeden

Spirálový model – je to jen takový referenční model

- je to spíš meta-model nebo rámec, který ve fázi engineering obsahuje jiné modely jako vodopádový nebo iterativní
- spirála – fáze se dějí pořád dokola
- fáze:
  - plánování – vše se naplánuje
  - analýza rizik – vyplatí se to?, jaká jsou rizika
  - engineering – obsahuje nějaký model životního cyklu vývoje SW (analýza, návrh, implementace a testování, integrace a nasazení, ...)
  - zhodnocení zákazníkem
- důraz na opakování plánování a zhodnocení zákazníkem



Modelem řízené architektury (Model-driven architecture MDA)

- cílem je standardizace OO přístupu
- využívá jazyk UML a v něm jsou vytvořené různé modely, které jsou pak spustitelné
- ze specifikace v UML je tedy možné přímo automatizovaně a standardizovaně generovat SW, který je tím UML reprezentován
- vývoj SW je posloupnost transformací z požadavků po spustitelný SW
- jak název napovídá, jedná se o architektury, které jsou přímo řízené modelem, tedy vytvoří se model a z něj se generuje SW/architektura
- musí být automatizováno
  - ⇒ využívají se CASE nástroje – modelování v UML, generování kódu a dokumentace, automatizovaný refaktoring, ... IDE, ...
- vývoj SW za použití SW – nástroje pro tvorbu prototypů, pro backend, pro frontend, ...

Agilní vývoj a modelování

> MDA – nejprve se vytvoří model a z něho se transformacemi a generováním a automatizací tvoří SW
>
> x
>
> Agilní modelování – modely se vytváří jen když jsou potřeba a jen tak podrobně, jak je potřeba

- cílem je rychlá reakce na měnící se požadavky a podmínky
- přizpůsobivost
- důraz na komunikaci se zákazníkem
- projektový manažer a sprinty a weekplany a stand-upy
- reagování na změny požadavků je důležitější než dodržet termíny a plán projektu
- velká spolupráce se zákazníky
- spíše jednotlivci než automatizované nástroje a byrokratické procesy

Postupy a nástroje:
- test driven development – nejdříve testy – fungují i jako specifikace
- akceptační testování zákazníkem
- extrémní programování – programování v párech a skupinkách, ...
- kolektivní vlastnictví kódu – není že jeden člověk dělá jeden modul ale všichni zodpovědní za všechno
- správa verzí – více lidí nad stejným kódem
- kontrolní code – reviews
- refaktoring kódu ...

- využívají se nákresy na tabuli/papír při modelování, modely se vytváří až když jsou potřeba a jen tak přesné jak jsou potřeba
- CRC štítky – přiřazují zodpovědnosti třídám

Unified process (nebo i komerční verze RUP – Rational UP)

- používá se jako model životního cyklu vývoje SW ale je to více než model životního cyklu
- pro vývoj softwaru obecně
- je to i prostředí pro vývojáře – dokumenty, postupy, nástroje, návody, dokumentace, průvodce
- generický proces vývoje SW – adaptuje se pro danou organizaci pro kterou se SW vyvíjí

! – zkoumají se a modelují cílové organizace (standardy, databáze, procesy)
– adaptují se pro každou organizaci a projekt zvlášť

⇒ celý projekt se dělí na několik fází zakončených milníky, kde se poté zhodnocuje – ve fázích probíhají **iterace**

⇒ milník je výstup fáze

- iterativní vývoj
- používá se UML
- architektura založená na komponentách

Fáze:

1. Zahájení – řeší se proveditelnost a rizika a přínos a rozsah
   – řekne se jestli má cenu projekt vůbec realizovat

ANALÝZA + POŽADAVKY – vize projektu a plán vývoje + MODELOVÁNÍ ORGANIZACE

Milník: cíle projektu

2. Rozpracování – zpřesnění odhadů rizik
   – chceme vytvořit spustitelnou verzi architektury
   – určit zdroje a formulovat nabídku

POŽADAVKY + ANALÝZA + NÁVRH + IMPLEMENTACE + TESTOVÁNÍ
↳ zpřesnění ↳ vyřešení co přesně se bude dělat → návrh architektury → první spustitelná verze → testování první verze

Milník: spustitelná verze architektury

3. Konstrukce – ukončení požadavků, analýzy a návrhu
   – připravit finální verzi pro spuštění a zavedení

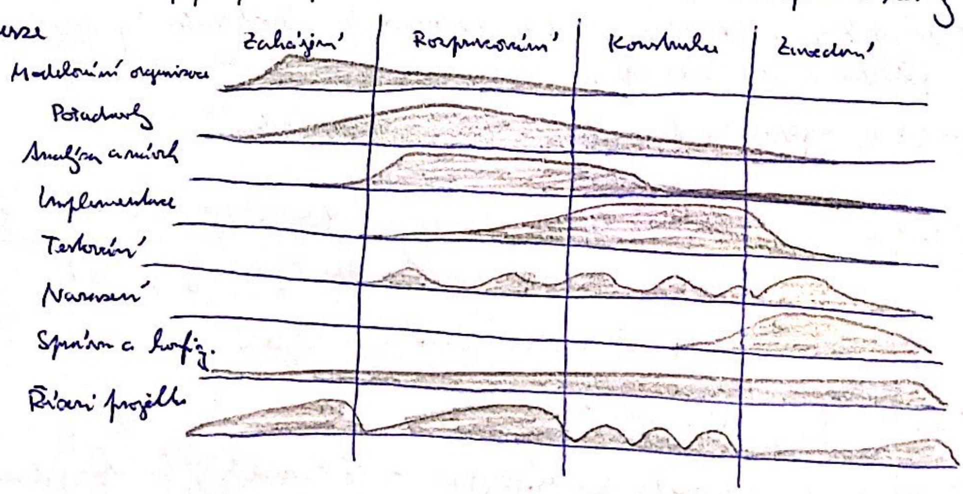
POŽADAVKY + ANALÝZA + NÁVRH + IMPLEMENTACE + TESTOVÁNÍ → testování tak aby to šlo nasadit
dokončení ↳ zajistit aby to fungovalo a šlo nasadit

Milník: počáteční provozní způsobilost

4. Zavedení – opravy chyb
   – konfigurace
   – testování
   – zaškolení uživatelů, příprava pracoviště

NÁVRH + IMPLEMENTACE + TESTOVÁNÍ
→ odstranění chyb z testování
↳ ještě případné úpravy
↳ beta testy a akceptační testy

Milník: produkční verze



Agilní modelování (+MDA +RUP)

- důraz na to, že obrázek či diagram řekne více než tisíc slov textu či zdrojového kódu !
- základní hodnoty jsou otevřená komunikace napříč všemi aktéry (analytiky, konzultanty, projektovými manažery, vývojáři, testery, zákazníky, klienty, ...)
- důraz na komunikaci s klienty / zákazníky
- důraz na přizpůsobivost změnám ⟹ "AGILNOST"
- komunikace, komunikace, komunikace !
- jednoduchost – nejjednodušší řešení je nejlepší řešení
- vytváří se modely – více modelů naráz
  - vytváří se paralelně
  - vytváří se i testovací modely

  ALE – vytvářet jen tolik modelů a návrhů kolik je potřeba a jen tak detailní jak je potřeba
  – model musí být přesný, pochopitelný, dostatečně detailní ale hlavně jednoduchý !

  X MDA (Model Driven Architecture)
  – vytváří se co nejpodrobnější a nejdetailnější model
  – velké spousty modelů a z nich se postupně vytváří SW – klidně i částečně automatizovaně (CASE, ...)

⟹ je třeba vytvářet jen tolik modelů a dokumentace, kolik je potřeba (aby s nimi člověk mohl vléct horu – cestování na lehko =D)

- modelování v malých přírůstcích
- iterativní
- kolektivní vlastnictví – kdokoliv může pracovat na libovolném modelu

⟹ celé agilní modelování vychází z extrémního programování

- TDD (test driven development)
- akceptační testy
- řídit se instinkty lidí a ne hloupými standardy
- rychlá zpětná vazba
- využívání návrhových vzorů, ...
- aktivní účast investorů a klientů

(a podmínkách a organizace) co nejvíce se přizpůsobuje specifickým potřebám projektu
některé by klasické fáze jako analýza, impl, návrh, ... se prolínají těmito fázemi
každá z fází má iterace – výstupy jsou milníky

- používá se i ve spojení s RUP
- pro menší projekty lepší Agile, pro velké RUP ⟵ zahájení, rozpracování / příprava, konstrukce, zavedení, nasazení, předání

UP (RUP)

se vyznačuje tím že se velmi přizpůsobuje a adaptuje na projekt / SW který se bude vyvíjet příp. na organizaci která je zadavatelem (obvykle na začátku ta organizaci si modeluje)

⇒ když to není úplně jasně daný a zavedený postup ale přizpůsobuje se konkrétnímu projektu

⇒ GENERICKÝ PROCES

- má vlastní fáze ← zahájení CÍLE PROJEKTU
  - rozpracování PRVOTNÍ SPUSTITELNÁ VERZE
  - konstrukce PROVOZNÍ ZPŮSOBILOST
  - zavedení PRODUKČNÍ VERZE

  které jsou se klasické fáze životního cyklu vývoje SW prolínají
- každá fáze má jako výstup nějaký milník
- iterace
- UML
- atd...