25. SW inženýrství (přístupy k vývoji SW)

Softwarové Inženýrství

Standart IEEE 1993 (softwarové inženýrství) je systematický, disciplinovaný a kvalifikovaný přístup k vývoji, tvorbě a údržbě software.

"Kód, kterému rozumí počítač, umí napsat každý trouba. Dobří programátoři píší kód, kterému rozumí lidé." – Martin Fowler

"Vždy pište kód tak, jako by ten chlapík, co ho po vás bude udržovat, měl být násilnický psychopat, který bude vědět, kde bydlíte." – Martin Golding

Důvody vzniku SW inženýrství

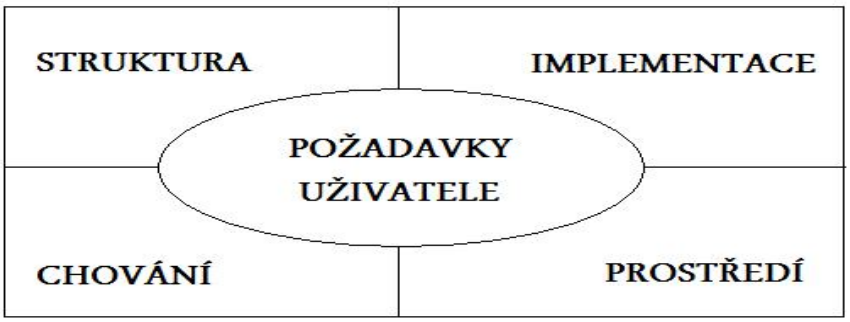
Softwarová krize měla dva hlavní problémy:

- **Rozsáhlost a komplexnost softwaru**
  - Počítače se rozmohly a měli k nim přístup i normální lidé
  - Software se začal zvětšovat a přestalo být možné, aby jeden člověk naprogramoval rozsáhlý software sám → Týmová spolupráce.
- **Opakování chyb**
  - Problémy se pořád opakují
  - Mění se pouze doba, jazyky a hardware
  - Pokud tedy někdo v historii vymyslel řešení daného problému a byl ochoten se o svoji myšlenku podělit, není důvod proč ji nevyužít → Návrhové, Architektonické vzory

UML

- 1994
- Unifikovaný modelovací jazyk
- Množina diagramů, které slouží pro vývojáře, návrháře i údržbu, k domluvě na projektu
- Nezávislé na programovacím jazyce
- Existuje software, ve kterém lze vytvořit UML diagram a zároveň vygenerovat kód v daném jazyce
- UML definuje své diagramy na určité pohledy

4 + 1 Pohled / 4 + 1 View



Požadavky uživatele / zákazníka

Nezajímá ho jak / v čem zadaný program bude vytvořen

Diagram případu užití (**Use case view**) → Z pohledu běžného koncového uživatele → Počítá, že daný software bude něco umět (přihlásit, odhlásit apod.)

Struktura

Statický pohled na systém (Structural View)

Diagram tříd (**Class diagram**) → Z jakých komponent (tříd) se systém bude skládat

Zachycuje z jakých částí, se daný systém bude skládat → Popis jednotlivých komponent

Chování

Dynamický pohled (Behavioral view) – jak se má systém chovat, než úspěšně dokončí určitou akci

Diagram aktivit (**Activity diagram**) → Pohled z dynamického pohledu → co všechno se musí udělat, než bude akce dokončena

Implementace

Pohled na implementaci / ze strany programátora (**Implementation view**) → Pohled programátora. Popis v konkrétním programovacím jazyce. Komponenty, které se musí programovat.

Prostředí / nasazení

Zajišťuje uživatelům, že ten systém bude použitelný pro uživatele

Diagram nasazení (**Deployment diagram**) → Pohledu správce systému → Na jaké technologii software poběží (požadavky)

PRO KAŽDÝ Z POHLEDŮ MÁ UML DEFINOVANÉ SVOJE DIAGRAMY.

Návrhové vzory

Popsané, ověřené a funkční řešení, nějakého problému.

- **Architektonické**
  - Zobrazují z jakých podsystémů, se daný systém bude skládat, a jak budou mezi sebou komunikovat (ISO/OSI, TABULE – komponenty řeší společný problém na tabuli)
  - **MVC**
  - **Pipeline**
  - **P2P**
  - **Server-Client**
  - **SOA – Service oriented architecture**
- **Návrhové**
  - Popis podsystému
  - **Vytvářecí**
    - **Factory**
      - Vyrábí instance podle seznamu
    - **Singleton / jedináček**
      - Od dané třídy existuje jediná instance

- **Strukturální**
  - **Fasade**
    - Vrstva mezi samostatným systémem a třídami
  - **Proxy**
    - Zástupný objekt za nějaký jiný → kontrola přístupu k danému objektu
- **Chování**
  - **Iterátor**
    - Slouží k vytvoření rozhraní, které sekvenčně prochází objekty uložené v kontejneru (složité architektuře)
  - **Observer**
    - Pozorovatel
    - Dohlíží nad systémem

- **Idiomy**
  - Platformě závislé
  - Konkrétní řešení (v daném jazyce) popsáno pomocí idiomů