7. Základy softwarového inženýrství

Počítačový systém v softwarovém inženýrství

Počítačový systém v softwarovém inženýrství zahrnuje vzájemnou souhru hardwaru a softwaru, která umožňuje řešit konkrétní úlohy nebo poskytovat služby. Hardwarová architektura představuje fyzické komponenty, jako jsou procesory, operační paměť, datová úložiště a periferie, jejichž vlastnosti ovlivňují výkon a možnosti systému. Softwarová architektura pak definuje, jak jsou softwarové komponenty navrženy, organizovány a jak mezi sebou komunikují. Klíčovou roli zde hraje **operační systém (OS)**, jenž je základní vrstvou softwaru, která spravuje systémové zdroje, řídí běh programů, správu paměti i komunikaci s hardwarem. OS umožňuje efektivní běh více aplikací, zajišťuje správu vstupně-výstupních operací i bezpečnost systému.

Nástroje pro vývoj software a jeho životní cyklus

Vývoj moderního softwaru je podporován širokou škálou nástrojů, které pokrývají celý životní cyklus – od plánování a návrhu přes implementaci a testování až po nasazení a údržbu.

Pro správu projektu a spolupráci se často používá například **Jira** (sledování úkolů a backlogu).

Správa verzí je řešena nástroji jako **Git** (distribuovaný systém umožňující lokální práci a snadné větvení) nebo SVN (centrální model).

Vývoj samotný probíhá v integrovaných vývojových prostředích (IDE) jako Visual Studio Code nebo IntelliJ IDEA.

Testování zajišťují nástroje jako Selenium (automatizace webových testů), **JUnit** či pytest (jednotkové testy). Automatizaci buildů, testování a nasazení podporují CI/CD nástroje jako GitLab CI/CD nebo Jenkins.

Kontejnerizace (např. Docker) usnadňuje přenositelnost aplikací mezi různými prostředími.

Dokumentace je často generována automaticky nástroji jako Swagger (API dokumentace) nebo Sphinx (dokumentace ke kódu).

Git vs. SVN

Git je distribuovaný systém – každý vývojář má plnou kopii repozitáře a může pracovat offline, změny pak slučuje do centrálního repozitáře. To umožňuje rychlé větvení, experimentování a efektivní spolupráci. SVN (Subversion) je centrální systém – veškeré změny probíhají přímo vůči centrálnímu serveru, což usnadňuje správu, ale omezuje práci offline a dělá větvení méně pružným.

Síťové webové technologie, webové technologie a databázové systémy

Moderní webové a síťové technologie umožňují vývoj aplikací a služeb, které běží na internetu a komunikují mezi sebou i s uživateli. **Protokoly jako HTTP/HTTPS** (pro přenos webových stránek) a **WebSocket** (pro real-time komunikaci) jsou základem webového prostředí.

Front-end technologie:

- **HTML** (struktura obsahu webu),
- **CSS** (vzhled a styl),
- **JavaScript** (interaktivita a dynamika na straně klienta, např. pomocí frameworku React).

  Back-end technologie:

- **PHP** (tradiční serverový jazyk, dnes spíše na ústupu),
- **Node.js** (moderní JS runtime na serveru, často ve spojení s Express),
- **Python (Django)**, **Java**, **.NET** (další běžné backend platformy).

Databáze ukládají data webových aplikací. **Relační databáze** (MySQL, PostgreSQL) využívají tabulky a SQL jazyk, **NoSQL databáze** (MongoDB) umožňují flexibilnější ukládání dat, například ve formě dokumentů (JSON/BSON).

API (Application Programming Interface) umožňuje komunikaci mezi aplikacemi. REST je architektonický styl využívající standardní HTTP metody (GET, POST, PUT, DELETE), GraphQL je moderní dotazovací jazyk s jedním endpointem.

Cloudové služby (AWS, Azure, GCP) umožňují provoz aplikací na vzdálené infrastruktuře a zajišťují škálovatelnost i vysokou dostupnost. Webové API jsou rozhraní, která umožňují komunikaci a integraci mezi různými systémy přes síť.

Softwarový proces

Softwarový proces je systematický přístup k vývoji softwaru – od plánování, návrhu, implementace až po testování a údržbu. Existuje několik modelů:

- **Vodopádový model**: Klasický sekvenční přístup – každá fáze začíná až po dokončení předchozí (požadavky → návrh → implementace → test → nasazení → údržba). Je vhodný pro projekty se stabilními požadavky, nevýhodou je malá flexibilita pro změny v průběhu vývoje.
- **Agilní model**: Flexibilnější iterativní přístup (např. Scrum, Kanban), který umožňuje rychlé reakce na změny požadavků, pravidelné dodávky funkčních verzí a zapojení zákazníka do procesu. Pracuje se v krátkých cyklech (sprintech), důraz je na spolupráci a častou zpětnou vazbu.
- **Inkrementální model**: Software se vyvíjí postupně, po částech (inkrementech) – každá nová verze přináší další funkcionalitu.

Volba procesu závisí na velikosti a povaze projektu, požadavcích zákazníka i zkušenostech týmu.

Role ve vývoji software

Vývoj softwaru je týmová práce a každý člen má svou specifickou roli:

- **Projektový manažer** – plánuje, organizuje, hlídá rozpočet, termíny a kvalitu.
- **Analytik** – zpracovává a specifikuje požadavky, zajišťuje komunikaci se zadavatelem.
- **Architekt** – navrhuje technické řešení, strukturu systému a jeho komponent.
- **Vývojář** – implementuje požadavky, píše a udržuje kód.
- **Tester** – ověřuje správnost a spolehlivost softwaru (jednotkové, integrační, systémové testy).
- **UI/UX designer** – navrhuje vzhled a použitelnost aplikace.
- **DevOps engineer** – zajišťuje plynulý provoz, nasazení, monitorování a automatizaci.
- **Scrum master** (v agilních týmech) – dohlíží na správné dodržování agilní metodiky a pomáhá týmu odstraňovat překážky.

Vodopádový proces

Vodopádový (sekvenční) model je historicky jeden z prvních softwarových procesů, kde vývoj probíhá ve striktně definovaných fázích:

1. **Požadavky** – analýza a specifikace všech funkcí a vlastností systému.
2. **Návrh** – návrh systémové architektury, datových struktur a rozhraní.
3. **Implementace** – samotné programování podle návrhu.
4. **Testování** – hledání a odstraňování chyb, ověřování funkčnosti.
5. **Nasazení** – předání systému uživatelům, nasazení do provozu.
6. **Údržba** – opravy chyb, aktualizace, rozšiřování.

Charakteristické je jednosměrné plynutí procesu, kdy se k předchozím fázím již zpravidla nevrací (minimální zpětná vazba).

RUP (Rational Unified Process)

RUP je iterativní a inkrementální rámec pro vývoj softwaru, který dělí projekt do čtyř hlavních fází:

1. **Inception (počátek)** – stanovení cílů, rozsahu a základní ekonomické analýzy projektu.
2. **Elaboration (rozpracování)** – detailní analýza požadavků, architektury, odhalení rizik, plánování.
3. **Construction (realizace)** – samotný vývoj a testování systému v opakovaných iteracích.
4. **Transition (přechod)** – nasazení, migrace, školení uživatelů a podpora provozu.

RUP je silně **use-case driven** (tj. funkce a scénáře užití jsou středobodem návrhu) a podporuje důraz na kvalitu a rizikové řízení. Definuje jasné role, aktivity a artefakty v průběhu celého vývoje.

UML (Unified Modeling Language)

UML je standardní vizuální jazyk pro modelování a dokumentaci softwarových systémů, který umožňuje vývojářům a analytikům efektivně komunikovat strukturu, chování i architekturu aplikace. UML zahrnuje několik typů diagramů:

- **Use Case Diagramy** – popisují scénáře použití systému (co má systém dělat, jak jej uživatelé využívají).
- **Class Diagramy** – znázorňují strukturu tříd, atributy, metody a vztahy.
- **Sequence Diagramy** – ukazují průběh komunikace mezi objekty v čase.
- **Activity Diagramy** – modelují tok činností a procesů.
- **State Machine Diagramy** – popisují stavy objektů a přechody mezi nimi.
- **Component a Deployment Diagramy** – zobrazují rozložení systémových komponent a jejich nasazení v infrastruktuře.

UML podporuje i pokročilé notace, například stereotypy a tagged values, které umožňují přizpůsobení modelů specifickým potřebám projektu.

Agilní metodiky vývoje softwaru

Agilní vývoj je filozofie i sada metodik, které kladou důraz na flexibilitu, iterativní rozvoj, časté dodávky a spolupráci v týmu i se zákazníkem.

Scrum - pracuje v krátkých iteracích (sprintech), má jasně definované role (Product Owner, Scrum Master, tým), pravidelné schůzky (Daily Scrum, Sprint Planning, Review, Retrospektiva) a transparentní správu úkolů v backlogu.

Kanban - využívá vizuální správu pracovního toku na Kanban tabuli, omezuje množství rozpracované práce (WIP) a zaměřuje se na plynulost a rychlost dodávky. Agilní přístupy umožňují rychle reagovat na změny požadavků, zvyšují motivaci týmu a zlepšují komunikaci s klientem, což je v dnešním dynamickém světě softwaru velmi cenné.