Testování softwaru dle ISTQB

Učební pomůcka od **ITnetwork.cz** - Přehled základní syntaxe

Základy testování

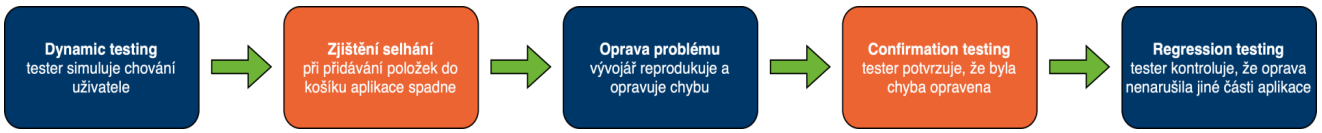
Testování je proces ověřování **správnosti** a **kvality** softwaru.

Dynamické testování

Probíhá **za reálného běhu aplikace** (manuálně či automatizovaně) a slouží k ověření funkčnosti i k odhalení chyb.

Statické testování

Revize, kontroly a analýzy kódu či dokumentace **bez spuštění aplikace**. Cílem je co nejdříve odhalit chyby, například v požadavcích, návrhu nebo kódu.



Cíle testování

1. Odhalit defekty (chybné funkce, bezpečnostní slabiny).
2. Předejít chybám (**QA** - preventivní přístup).
3. Posoudit kvalitu produktu (**QC** - nápravný přístup).
4. Ověřit, že systém plní potřeby uživatele.

Quality Assurance (QA) vs. Quality Control (QC)

Quality Assurance (QA) se zaměřuje na procesy a metodiky, tedy na ověřování, že software vzniká správným způsobem (termín **verifikace** – "Are we building it right?").

Quality Control (QC) se zaměřuje přímo na hotový produkt a jeho kvalitu, tedy na ověřování, že software plní očekávání uživatelů (termín **validace** – "Are we building the right thing?").

Principy testování

- **Testování ukazuje přítomnost defektů, ne jejich nepřítomnost.**
- **Kompletní testování není možné** – prioritizace a vhodné techniky.
- **Včasné testování** = úspora času i peněz.
- **Shlukování defektů** – většina chyb se nachází v několika klíčových oblastech.
- **Testy se opotřebovávají** – je potřeba je udržovat a aktualizovat.
- **Závislost na kontextu** – přístup k testování se liší podle projektu.
- **Nepřítomnost defektů neznamená, že je produkt použitelný.**

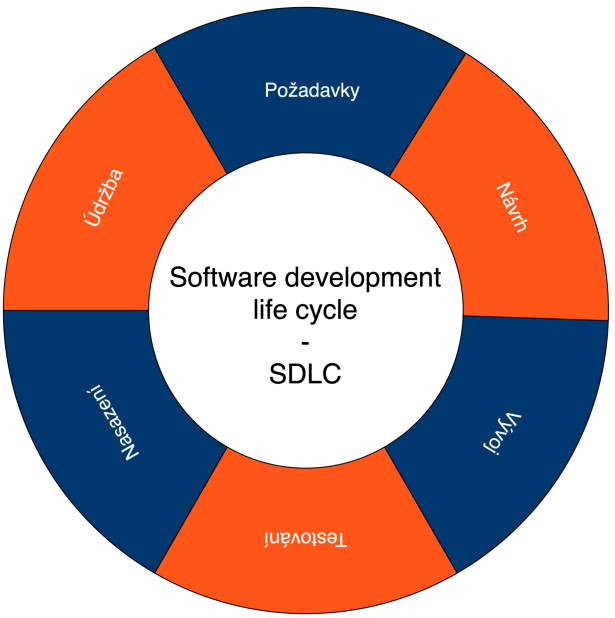
SDLC (Software Development Life Cycle)

Modely vývoje

- **Sekvenční (např. waterfall)** – jasné fáze (analýza, návrh, implementace, testování, nasazení).
- **Iterativní** – cykly vývoje, průběžné zlepšování.
- **Inkrementální** – přidávání nových funkčností v malých krocích.
- **Agilní přístupy (Scrum, Kanban, XP, TDD, BDD, ATDD)** – kladou důraz na rychlou zpětnou vazbu a neustálé zlepšování.

Testeři v SDLC

- **V sekvenčních modelech** testeři kontrolují dokumentaci v raných fázích, testování probíhá až ve vyhrazené fázi.
- **V iterativních a agilních modelech** testují průběžně (části produktu, inkrementy) a udržují **regression testing** (opakované testování, zda nové změny nepoškodily dříve funkční části).



Testovací činnosti

- **Test planning**
  - Definice cílů testování, přístupu a harmonogramu.
  - Stanovení priorit, rizik, prostředků.
- **Test monitoring & control**
  - Průběžné sledování stavu testů (metriky, reporty).
  - Úpravy plánu testování dle potřeby.
- **Test analysis**
  - Analýza požadavků a specifikací.
  - Identifikace testovacích podmínek a potenciálních defektů.
- **Test design**
  - Tvorba testovacích scénářů, testovacího prostředí a dat.
- **Test implementation**
  - Příprava testovacích sad, skriptů a procedur.
- **Test execution**
  - Spuštění testů, vyhodnocení výsledků, reportování chyb.
- **Test closure**
  - Archivace testware, závěrečný report, zjištění ponaučení.

Testware a report o defektu

Testware je **souhrn výstupů z testovacích činností** (např. test plány, skripty, defekt reporty, protokoly).

Report o defektu zaznamenaný během dynamic testingu obvykle obsahuje **jedinečný identifikátor**, **název a shrnutí anomálie**, **datum**, **organizaci a autora reportu**. Dále zahrnuje **informace o testovaném objektu a prostředí**, **kontext defektu**, **popis selhání** umožňující jeho reprodukci a vyřešení, **očekávané vs. skutečné výsledky**, **závažnost a prioritu opravy**, **stav defektu** a případné **odkazy**.

Úrovně testování

- **Unit testing -** testování jednotlivých komponent izolovaně (obvykle vývojáři).
- **Component integration testing -** Test interakcí mezi moduly/komponentami.
- **System testing -** test celého systému podle definovaných požadavků (funkční i nefunkční).
- **System integration testing -** Ověření integrací s externími systémy/službami.
- **End-to-End (E2E) testing** – Ověření celého uživatelského procesu napříč systémem a jeho integracemi, simulující reálné scénáře použití.
- **Acceptance testing -** Poslední fáze, často zapojení koncoví uživatelé (UAT, beta test)

Typy testů

- **Functional testing** - Kontrola funkčních požadavků.
- **Non-functional testing -** Výkon, zátěž, bezpečnost, kompatibilita, udržovatelnost, použitelnost (ISO/IEC 25010)
- **Black-box testing** - Zaměřeno na vstupy a výstupy, **bez znalosti vnitřní logiky kódu**, např.: Equivalence Partitioning (EP), Boundary Value Analysis (BVA), Decision Table, State Transition.
- **White-box testing** - Testování **s detailní znalostí kódu**, měří pokrytí příkazů a větví (Statement & Branch Coverage), zaměřuje se na vnitřní strukturu a logiku programu.
- **Experience-based testing** - Exploratory testing, Checklist-based testing, Bug Estimation.