Testování software

Testování SW má podstatný vliv na kvalitu dodaného produktu.

Náklady na odstranění chyby stoupají, v čím pozdější fázi životního cyklu aplikace je chyba nalezena.

Na druhé straně, vytvořit zcela bezchybný SW není možné. Je tedy třeba zvolit vhodný kompromis mezi množstvím testů a uvedením produktu do provozu.

Důvody neopravení chyby:

a) Není dost času
b) Není to chyba (nesprávně pochopená funkce)
c) Oprava chyby je riskantní (oprava může způsobit poškození již ověřené části)
d) Oprava nestojí za to (pravděpodobnost výskytu je nízká)

Typy testů

BLACK BOX x WHITE BOX.

STATICKÉ TESTY x DYNAMICKÉ TESTY

Test typu „black box“ znamená, že testujeme aplikaci, od které nemáme k dispozici zdrojový kód. U „white boxu“ je zdrojový kód k dispozici a můžeme tedy k testování využít znalost vnitřní struktury nebo simulovat určité stavy pomocí debugeru.

Statický test znamená např. revizi kódu, revizi analytických podkladů, test funkční specifikace.

Dynamické testování znamená, že testujeme aplikaci, která je spuštěna.

Grey Box - omezená znalost interních datových a programových struktur za účelem navrhnutí vhodných testovacích scénářů, které se realizují na úrovni black box (např. testy interface).

Testy specifikace

Je specifikace:

- Úplná
- Správná (nejsou v popisu chyby?)
- Přesná a jednoznačná
- Konzistentní (nejsou jednotlivé části v konfliktu?)
- Relevantní (jsou navržené funkce potřebné?)
- Proveditelná (je to realizovatelné v rámci rozpočtu a času?)
- Testovatelná

Obvyklé chyby specifikace:

- Obsahuje kvantifikátory vždy, nikdy, každý, žádný, všichni…
- Snaha někam dotlačit – je nabíledni, určitě, evidentně…
- Seznam není úplný – a tak dále, například…
- Vágní specifikace – někdy, něco, obvykle, zpravidla, většinou…
- Nejednoznačné kvantifikátory - dobrý, laciný, malý, stabilní
- If rozhodování bez specifikace else větve

Dynamické testování černé skříňky

a) Test-to-pass (test splněním) – kontrolujeme, zda aplikace vyhovuje minimální požadované funkčnosti
b) Test-to-fail (test selháním) – snaha najít chybu nebo aplikaci „shodit“

Testování dat (vstupů)

a) Test hraničních podmínek – délky řetězců, zadání čísla větší než je maximum/minimum atd.
b) Test subhraničních podmínek – testování vnitřních omezení (např. položky integer)
c) Test výchozích, prázdných, nevyplněných a nulových hodnot
d) Test na zadání neplatných nebo nesmyslných údajů

Testování stavů (logiky aplikace)

Součástí těchto testů by mělo být také

a) testování race condition (souběhu). Příkladem je např. uložení stejného dokumentu z dvou aplikací, současný přístup do databáze, sdílení portů, sdílení tiskárny, stisk klávesy v průběhu zavádění programu…
b) test opakováním – co se stane, když se nějaká akce neustále opakuje (např. klikání na tlačítko pro zobrazení formuláře…), hledají se např. memory leaks (zahlcení paměti)
c) stresové testování – test běhu aplikace při nejhorších možných podmínkách (málo paměti, málo místa na disku…)
d) load test (zátěžové testy) – simulace velkého počtu uživatelů, velkého počtu přístupů, velkého množství transakcí…

Testování konfigurace

Běží aplikace na všech možných kombinacích hardwarových komponent?

Tento typ testu je obtížný na realizace obvykle nemáme většinu HW k dispozici.
Lze využít publikovaných HW specifikací…

Testování kompatibility

Dokáže vytvořený software pracovat souběžně s jiným SW?

Backward compatibility – zpětná kompatibilita – pracuje s předchozími verzemi tohoto software? Je datově kompatibilní?

Forward compatibility (dopředná kompatibilita) – bude kompatibilní s budoucími verzemi?

Testování multijazyčné podpory

Lokalizace není synonymem pro slovo překlad, ale je to soubor celého komplexu opatření. Zahrnuje kromě vlastní aplikace také lokalizaci dat (např. použití UNICODE), lokalizaci horkých kláves, třídění textů, směr psaní, lokalizace textů v grafice, formátování dat.

Problémy s formátováním dat u multilanguage aplikací:

a) měrné jednotky
b) číselné údaje
c) měna (symbol a jeho umístění)
d) zobrazení datumu
e) zobrazení času (12/24)
f) kalendář (juliánský, gregoriánský, arabský…)
g) adresy (např. tvar PSČ)
h) telefonní čísla
i) velikost papíru a obálek pro tisk

Počítalo se s lokalizací od počátku návrhu?

Zásadní pro lokalizaci je nemít texty uprostřed kódu!

Usability test (Test použitelnosti)

Test použitelnosti - jak snadné je produkt zprovoznit a používat ho (snadnost, efektivita, přesnost, zapamatovatelnost…), formou black boxu.

Je aplikace:

- intuitivní na ovládání?
- konzistentní? (např. co se týče použité terminologie, umístění tlačítek…)
- flexibilní?
- dodržuje standardy? (např. pozice ano – ne tlačítek v dialozích)
- dostupná pro handicapované?
- V souladu s dokumentací?

Unit testy

Unit (jednotka) – samostatně testovatelná část aplikace. V OOP odpovídá většinou třídě. Používají se také při regresním testování.

- Provádí vývojář
- Byly použity vhodné algoritmy, návrhové vzory ?
- Ověření správnosti fungování dílčí části kódu
- Stubs, mock object, fakes, …
- Scrum, TDD – unit testy se vytvářejí před kódem

Downgrade test

Je možné se vrátit k předchozí verzi?

Test instalace

Lze aplikaci nainstalovat a poté bez problémů odinstalovat?

Long Term Test

Jak se aplikace chová, je-li trvale spuštěna.

Akceptační test

Testy prováděné při předání aplikace; většinou součástí smlouvy, někdy dohodnuté na počátku projektu. Idea je, že pokud aplikace projde akceptačními testy, měla by být objednatelem převzata a zaplacena.

Performance test

Test rychlostí odpovědi aplikace na činnost uživatele nebo na události.

Security test

Test zabezpečení aplikace. Součástí mohou být také testy implementace integritních omezení.

Recovery test

Vzpamatuje se aplikace po selhání HW? Co se stane, když se vypne počítač v situaci, kdy aplikace je spuštěna?

Regresní testy

Opakované provádění testů na již otestovaných částech. Snaha zjistit, zda oprava chyby nezpůsobila nefunkčnost jiné části aplikace.

Automatizace testování

- Monitor
- Ovladač (skript nahrazující klávesnicové vstupy)
- Maketa (stub) – přebírá výsledek sw (např. místo tiskárny se tisk ukládá do souboru)
- Záznam maker

- Generátory „šumu“
- Nástroje pro stresové a zátěžové testy (např. Microsoft Stress)
- Analytické nástroje
- Nástroje typu „hloupá opice“ (náhodné klikání nebo vstupy)
- Nástroje typu „polointeligentní opice“ – rozpozná nalezení chyby
- Nástroje typu „inteligentní opice“ – nástroj ví kde je a co dělá

Beta testy

Zapojení externích osob do testování, např. potenciální uživatelé. Není zde jistota, že beta testeři otestují vše. Výhodou je ale např. ověření kompatibility s HW kombinacemi.

Strategie testování

Kolik osob, časový plán testů, stanovení metrik…