# Validácia Verifikácia a Testovanie Softvéru – Modely a techniky

MARTIN MICHÁLEK

*Slovenská technická univerzita*
*Fakulta informatiky a informačných technológií*
*Ilkovičova 3, 842 16 Bratislava*
michalek[zavináč]stonline[.]sk

**Abstrakt.** Tak ako aj v iných oblastiach, aj v oblasti vývoja softvéru zohráva kvalita významnú rolu pri určovaní konkurencie schopnosti výsledného produktu. Jedným zo základných nástrojov na zvýšenie kvality softvéru je jeho systematické testovanie a overovanie. Pri enormnom tempe v oblasti vývoja softvéru sa aj samotné testovanie stalo netriviálnym procesom zasahujúcim do všetkých etáp životného cyklu softvéru. V tejto práci sa budeme podrobnejšie zaoberať procesom testovania. Predstavíme niektoré známe modely testovania ako je klasický V-model alebo model Testami riadeného vývoja (Test Driven Development model). Základnou motiváciou práce je vyzdvihnúť aspekty a techniky testovania aplikovateľné najmä pre softvérové projekty menšieho rozsahu.

## Úvod

V roku 1972 Dijkstra prehlásil, že testovaním programov môžeme existenciu chýb v programe dokázať, nemôžeme ju však vyvrátiť, a preto testovanie ako také nie je pre určenie správnosti programu akceptovateľné[6]. Napriek tomu zaznamenalo testovanie v posledných dekádach neuveriteľný pokrok (možno aj preto, že žiadnu akceptovateľnú metódu, ktorá by zaručila 100% overenie správnosti programu sa zatiaľ nepodarilo vymyslieť).

Ďalší z priekopníkov v oblasti testovania G. J. Mayers pokladal testovanie za proces vykonávania programu za účelom nájdenia chyby a problém testovania rozdelil do troch dimenzií: účel, granularita a stratégia[3] .

Neskôr boli dodatočne pridané ďalšie dimenzie ako použité technológie či problémová oblasť. Spomenuté skutočnosti hovoria o pokroku, ktorý bol v rámci testovania dosiahnutý. Motiváciou pre ďalší rozvoj testovania môže byť skutočnosť, že samotné testovanie môže v rámci vývoja softvérového systému zabrať až 50% úsilia resp. zdrojov.

*Manažment v softvérovom inžinierstve, október 2007, s. 1-1.*

| Účel | Granularita | Stratégia |
|---|---|---|
| Akceptácia | Funkcia | Riadená špecifikáciou (back-box) |
| Defekt | Objekt | Riadená implementáciou (white-box) |
| Spoľahlivosť | Komponent | Riadená používaním (Operational) |
| Výkon | Aplikácia | Náhodná (Štatistická) |
| Použiteľnosť | Systém | Intuitívna (Ad hoc) |

**Tab. 1.** Tri dimenzie testovania softvéru

V mojej práci sa budem snažiť opísať testovanie ako nenahraditeľnú časť vývoja softvéru. Na začiatku si zadefinujem základné pojmi z oblasti testovania. Následne vyzdvihnem niektoré problémové aspekty testovania, ktoré som si všimol. Predstavím V-model vývoja softvéru, ktorý zároveň predstavuje základný pohľad na testovanie softvéru. Opíšem techniku Testami Riadeného Vývoja (Test Driven Development-TDD) softvéru ako jednu z najrýchlejšie sa rozvíjajúcich techník testovania v súčasnosti. Nakoniec zhrniem zozbierané poznatky, pričom sa ich budem snažiť aplikovať na použitie v malých projektoch.

## Validácia, verifikácia, testovanie

Testovanie sa často spája s pojmami validácia a verifikácia. Rozdiel medzi týmito pojmami môže byť pomerne nejasný a v rôznych literatúrach je definovaný inak. Z môjho pohľadu by som zvolil nasledovnú definíciu. Verifikácia je kontrola za účelom zabezpečenia konzistencie s danou špecifikáciou. Validácia je proces zisťovania či systém spĺňa to, čo si "zákazník" naozaj prial. Testovanie je vo všeobecnosti kontrola určitej vlastnosti. Môže byť použité za účelom verifikácie tak ako aj validácie, pričom využíva rôzne nástroje a techniky.

- Verifikácia: Je systém vytvorený správne? Zhoduje sa zo špecifikáciou?
- Validácia: Vytvorili sme správny systém? Spĺňa to, čo si zákazník prial?
- Testovanie: Bola zabezpečená požadovaná vlastnosť?

Vo všeobecnosti sú v rámci testovania rozlišujeme dva typy nekorektného správania. Je to chyba (*fault* angl.) a zlyhanie (*failure* angl.). Chybu môžeme interpretovať ako nesprávnosť v kóde, návrhu, architektúre či špecifikácii. Na ich

odstránenie je určená verifikácia. Zlyhanie je problém zlej funkcionality a na jeho detekovanie slúži validácia.

Pojmi zlyhanie a validácia sa používajú hlavne v kontexte, kde chceme poukázať na ich odlišný význam. Vo všeobecnosti budem pojmom chyba prezentovať ako všeobecný problém.

## Problémy testovania

Pri študovaní materiálov z oblasti testovania som si všimol niektoré problémy a prekážky, ktoré sa pri testovaní softvéru objavujú. V nasledujúcich odstavcoch opíšem tie, ktoré sa mi zdali najzaujímavejšie a bolo by vhodné si ich priblížiť. Sú to:

1. Problém času
2. Problém kompletnosti
3. Psychologické problémy
4. Problém Quis Custodiet Ipsos Custodes? (Kto dozerá na dozerajúceho?)
5. Problém špecifikácie

### Problém času

Na to aby systém fungoval istú dobu bez toho aby sa v ňom objavila chyba, je potrebné aby bol aspoň tak isto dlhú dobu testovaný. Nie vždy je možné podrobiť systém tak dlhému testovaniu(v praxi takmer nikdy). Napriek tomu boli vymyslené niektoré metódy, ktoré to dovoľujú. Príkladom je takzvané Beta-testovanie (Beta testing angl.). Princípom Beta- testovania je sprístupnenie Beta-verzie systému skupine používateľov, väčšinou zadarmo alebo dokonca za dohodnutú odmenu (Beta-testers), ktorí systém používajú a tým objavujú skryté chyby. Tento prístup sa využíva väčšinou pri vytváraní nových verzií už existujúcich systémov(staršia oficiálna verzia sa používa, nová beta verzie sa pripravuje). Tieto systémy patria z pravidla do skupiny nazývanej „of the shelf“. Z môjho pohľadu je zrejmé problém ťažko odstrániteľný. Jeho riešenie vidím v úplnej automatizácii procesu testovania, ktorá je však zatiaľ iba víziou do ďalekej budúcnosti a expertmi je považovaná za nedosiahnuteľnú[2].

### Problém kompletnosti

Problém kompletnosti priamo súvisí s problémom času. Jeho základ tvorí už spomínaná myšlienka, ktorá hovorí, že testovaním môžme dokázať existenciu chýb v programe, nie však ich absenciu. Z toho vyplýva, že bezchybnosť systému sa nedá garantovať. Čím je systém zložitejší, tým viac je problém kompletnosti aktuálny. Argumentom v tomto prípade nemôže byť ani zoznam už nájdených chýb. Práve naopak, podľa Mayersa počet nájdených chýb v systéme zvyšuje pravdepodobnosť ich ďalšieho výskytu[4].

**Psychologické problémy**

Aj keď sa to môže zdať akokoľvek čudné, kvalita testovania závisí v určitých prípadoch aj od psychologickej stránky testera. Táto situácia nastáva v prípadoch, keď programátor testuje svoj vlastný program. Vyplýva to z faktu, že testovanie je vo svojej podstate deštruktívny proces a programátor ho podvedome nechce poškodiť. Do určitej miery to platí aj v prípadoch, keď sú tester s programátorom priatelia. Tento a príbuzné problémy sú pozorované najmä v malých tímoch na malých projektoch. Riešením je vytvorenie samostatného oddelenia v organizačnej štruktúre, zaoberajúceho sa iba manažmentom kvality resp. testovaním.

**Kto dozerá na dozorcu?**

Dalo by sa povedať, že táto množina problémov tvorí nadmnožinu Psychologických problémov. Okrem už spomenutej chyby sem patria všetky chyby, ktoré vzniknú pri testovaní. Môžu byť spôsobené ľudským faktorom, ale aj chybou nástrojov(softvérových, hardvérových, iných...) použitých v procese testovania.

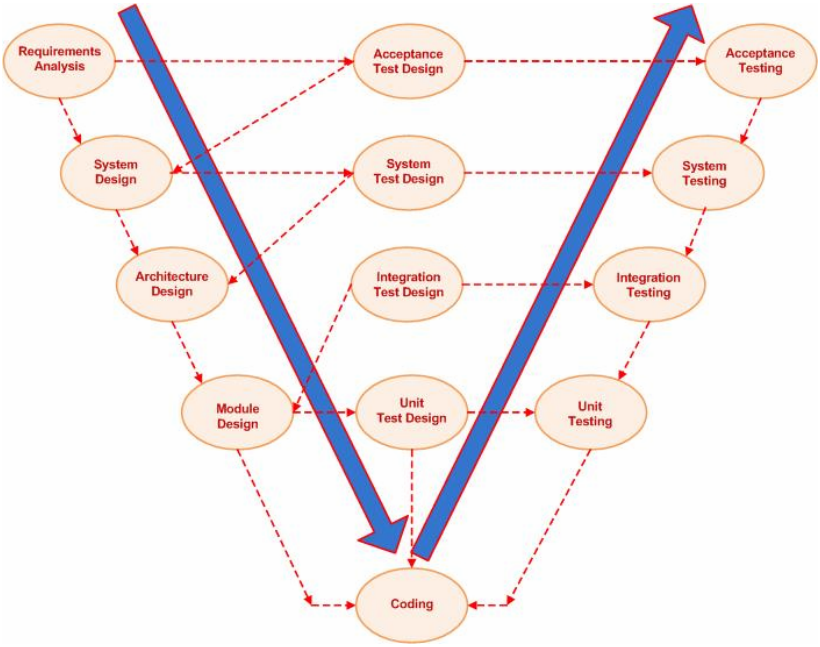
**Problém špecifikácie**

Skúsenosti hovoria, že špecifikácia požiadaviek zákazníka nie je nikdy kompletná. Tento druh je obzvlášť nebezpečný, pretože sa stáva aktuálnym až pri odovzdávaní systému používateľovi(záleží od zvoleného procesu vývoja systému), pri validácii. Preto je odstránenie problémov v špecifikácii obzvlášť nákladné. Podľa môjho názoru spadá problém špecifikácie do skupiny neriešiteľných problémov. Potreba úplnej a konzistentnej špecifikácie je vyzdvihovaná hádam v každom aspekte softvérového inžinierstva a predsa je väčšine prípadov nepostačujúca. Problém priamo vyplýva z rôznych pohľadov programátora/vývojára na jednej strane, a zákazníka na strane druhej, na danú problémovú oblasť. Taktiež vidím problém v tom, že vývoj softvéru trvá určitú dobu počas, ktorej sa mení situácia nielen na strane vývojára ale aj na strane zákazníka.

## V-model vývoja softvéru

V-model je jedným základným modelom vývoja softvéru. Z pohľadu testovania by som ho rozdelil na dve časti:

- Pred “kódovaním“ (Pred programom)
- Po “kódovaní“ (S programom)




Obr. 1. V-model vývoja softvéru

Testovanie pred kódovaní zahŕňa:

- Verifikácia špecifikácie
- Verifikácia návrhu
- Verifikácia architektúry

Testovanie po kódovaní zahŕňa:

- Testovanie súčiastok
- Integračné testovanie
- Systémové testovanie
- Akceptačné testovanie

Z programátorského pohľadu je pre mňa zaujímavejšie testovanie po kódovaní. V nasledujúcich riadkoch rozoberiem jednotlivé jeho fázy.

*Testovanie súčiastok*

Testovanie súčiastok je prvým z krokov testovania kódu. Dochádza ku kontrole samotného kódu(White Box testing). Zaujímavosťou je, že odborníci odhadujú chybu

nájdenú v tejto fáze testovania za sto krát lacnejšiu ako chybu nájdenú pri akceptačnom testovaní.

*Integračné testovanie*

Pri integračnom testovaní sa používa takzvaný Black Box testing. Testujú sa rozhrania, reakcie programu na vstupy a porovnávanie s požadovanými výstupmi.

*Systémové testovanie*

Systém sa testuje ako celok. Sleduje sa funkčnosť vzhľadom na špecifikáciu.

*Akceptačné testovanie*

Testovanie vzhľadom na požiadavky zákazníka. Je vykonávané testermi, ktorí sa vyznajú v problémovej oblasti, v prostredí v ktorom bude systém nasadený- na strane zákazníka.

## Testami riadený vývoj softvéru

Ako hovorí samotný názov ide vlastne model respektíve techniku vývoja softvéru. Za základný kameň pri tvorbe systému považuje tvorbu testov. Jej hlavnou myšlienkou je vytvorenie testov pred vytvorením kódu čím sa stala alternatívou k dovtedy známym technikám testovaniu po kódovaní(Testing After Coding- TAC). Stala sa tak prevratnou aj z pohľadu histórie a vnímania testovania ako takého. Testami riadený vývoj softvéru(ďalej TDD) pozostáva z nasledujúcich fáz[1]:

1. Pridanie nového testu
2. Spustenie testov, sledovanie pridaného testu
3. Dopísanie kódu za účelom odstránenia odhalených chýb
4. Znovu spustenie testov
5. Refaktoring

*Pridanie nového testu*

Z pohľadu V-modelu sa v rámci TDD ako základ používa testovanie súčiastok. V prvej fáze si programátor navrhne súčiastku, definuje jej rozhrania, metódy, funkcie. Následne vytvorí test/testy, ktoré budú zaručovať požadované vlastnosti definovanej súčiastky.

*Spustenie testov*

Spustením testov sa programátor dozvedá ktoré súčiastky boli ovplyvnené novým testom a bude nutné ich modifikovať.

*Dopísanie kódu*

Prispôsobenie a doplnenie kódu, tak, aby spĺňal požiadavky testov.

*Znovu spustenie testov*

Tvorí akúsi spätnú väzbu pre programátora. Opätovným spustení testov sa dozvie, či je kód ktorí napísal v predchádzajúcej etape postačujúci na prejdenie definovanou sadou testov. Ak všetky testy prebehnú správne, nasleduje fáza refaktoringu, ak niektorý z testov zlyhá je nutné znovu doplniť kód tak aby splňoval požadované vlastnosti.

*Refaktoring*

Fáza refaktoringu prichádza na rad, keď sú splnené všetky požadované vlastnosti definované testami. Ide vlastne čistenie kódu od zbytočností, zvyšovanie prehľadnosti a čitateľnosti kódu.

## Záver

V rámci záveru zhrniem nadobudnuté poznatky a budem sa snažiť vyjadriť svoj názor na jednotlivé aspekty testovania opísané v predchádzajúcich kapitolách.

Nepochybne existuje mnoho iných problémov v procese testovania, ktoré som v texte nespomenul. Niektoré z nich sa mi nezdali dosť zaujímavé, boli príliš "otrepané" alebo príliš špecifické, iné som skrátka nemal tú česť spoznať.

Myslím si že V-model vývoja softvéru je skvelá vec ako sa naučiť a porozumieť základom vývoja softvéru a jednotlivým fázam obsiahnutým v tomto procese. Pomocou neho je možné si predstaviť riešenie vývoja akéhokoľvek systému. Jeho praktická aplikácia je však pre jeho(na môj vkus až príliš veľkú) všeobecnosť takmer nemožná.

Testami riadený vývoj softvéru je jednou zo skupiny mladších extrémne rýchlo sa vyvíjajúcich oblastí testovania. Jeho diametrálne odlišný pohľad na vec ma zaujal. Skutočnosť, že programátor je nútený vytvárať testy ešte pred vytvorením kódu, mu dáva možnosť pozrieť sa na problémy z úplne iného pohľadu ako bol zvyknutý. Taktiež vyzdvihnutie refaktoringu na úroveň hlavnej fázy TDD a kladenie dôrazu na tvorbu "pekného" kódu považujem za jednu z hlavných výhod TDD. Ako poslednú by som chcel vyzdvihnúť skutočnosť, že TDD je veľmi dobre aplikovateľný pre malé projekty ako napríklad Tímový projekt.

## Použitá literatúra

1. Beck, K.: Test Driven Development: By Example, Addison-Wesley Longman Publishing Co., Inc., Boston, MA(2002)
2. Bertolino, A.: Software Testing Research: Achievements, Challenges, Dreams. 2007 Future of Software Engineering FOSE '07(2007)

3. Jones, E. L., Chatmon, C.L.: A perspective on teaching software testing. Journal of Computing Sciences in Colleges, Vol. 16 Issue 3(2001)
4. Myers, Glenford J.: Software Reliability, John Wiley & Sons, New York(1976)
5. Pfau, P. R.: Applied quality assurance methodology. ACM SIGSOFT Software Engineering Notes, Vol.3 (1978)
6. Zhu, H.: Software Unit Test Coverage and Adequacy. ACM Computing Surveys (CSUR), Volume 29 Issue 4 (1997)

## Annotation

*Validation Verification and Testing – Medels and techniques*

As well as in other areas, in the area of software development quality plays a major role in determining the competitive attribute of the final product. One of the basic tools for software quality increasment is its systematic testing and validation. With enormous development in area of software development testing become untrivial process spreading into all phases of software lifetime cycle. In this essay I will concern on process of software testing in a detail. I Introduce some of the well known models of testing as V-model of software development or Test Driven Development model. The main motivation is to focus on testing techniques concerning application for small teams.