Procesné modely testovania softvéru

Roman Nagy

V priebehu uplynulých 20 rokov bolo vyvinutých niekoľko modelov pre proces vývoja softvéru. Medzi najznámejšie patria napr. „vodopádový model“ [5], „špirálový model“ [6] alebo „unified process“ [11], ktorý je vhodný najmä na návrh a vývoj objektovo orientovaného softvéru. Cieľom týchto modelov bolo definovať aktivity, ktoré je potrebné počas vývoja softvéru vykonať a zároveň definovať postupnosť ich vykonávania. Tým sa výrazne zefektívnil celý proces vývoja softvéru, a to najmä z hľadiska jeho riadenia a plánovania zdrojov. Spôsob, akým treba jednotlivé aktivity vykonávať, zostáva však otvorený, vďaka čomu sú tieto procesné modely použiteľné pri rôznych prístupoch k vývoju softvéru, ako je napr. procedurálne orientovaný vývoj, objektovo orientovaný vývoj a pod.

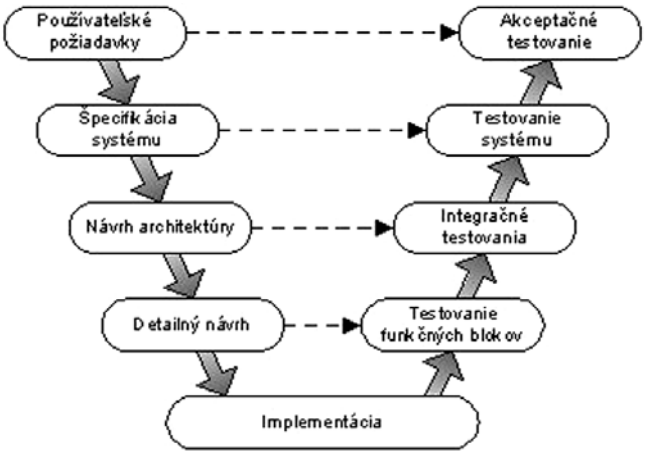
Popri procesných modeloch, ktoré sa zaoberajú opisom procesu vývoja, existujú aj modely, ktoré sa venujú procesu testovania softvéru. Tieto modely sú však menej známe. Dôvodom je, že na testovanie sa často hľadí ako na akúsi pomocnú či vedľajšiu činnosť, ktorá „musí byť vykonaná“ po ukončení implementácie. Je to aj napriek tomu, že aktivity spojené s testovaním tvoria približne 30 až 40 % ([8] uvádza dokonca až 50 %) celkových nákladov softvérového projektu.

Preto je cieľom tohto článku poskytnúť stručný prehľad najviac používaných procesných modelov na testovanie softvéru spolu s vyhodnotením ich výhod, nevýhod a možností použitia v praxi.

V-model

V-model [7], [8] je jedným z najznámejších a v praxi najpoužívanejších modelov testovacieho procesu. Širšie sa etabloval začiatkom 90. rokov a svoje uplatnenie nachádza najmä v Európe, kde je považovaný za alternatívu k „vodopádovému modelu“.

Tento model veľmi jednoduchým a zrozumiteľným spôsobom ukazuje súvislosti medzi vývojovými aktivitami na strane jednej a testovacími aktivitami na strane druhej. Ako vidieť z obr. 1, V-model sa skladá z dvoch vetiev, ktoré spoločne vytvárajú symbol v tvare písmena „V“. Ľavá vetva modelu zobrazuje vývojové aktivity a ich chronologickú postupnosť v procese vývoja softvéru (zhora nadol). Pravá vetva zobrazuje podobným spôsobom testovacie aktivity, pričom chronologická postupnosť aktivít testovacieho procesu



Obr.1 V-model

je naznačená zdola nahor. Vodorovné šípky smerujúce z ľavej vetvy modelu do pravej ukazujú, ktorá vývojová aktivita je podkladom pre ktorú testovaciu aktivitu.

Proces vývoja softvéru v ľavej vetve modelu sa začína definovaním používateľských požiadaviek. Tieto sa následne premietnu do špecifikácie systému, na základe ktorej sa vykoná návrh architektúry. Ďalej nasleduje detailný návrh a ako posledná vývojová aktivita je tu uvedená implementácia.

Aktivity testovacieho procesu sú v tomto modele rozdelené do štyroch základných úrovní, ktoré sú široko akceptované aj v odbornej literatúre [3], [4], [10], [15]. Patrí sem:

- Testovanie funkčných blokov (angl. unit testing). Ide o testovanie funkčne ohraničených elementárnych jednotiek programu. Touto jednotkou môže byť napr. funkcia, procedúra, alebo trieda. Je to testovanie softvéru na najnižšej úrovni.
- Integračné testovanie (angl. integration testing). Integračné testovanie testuje schopnosť vzájomnej integrácie už otestovaných elementárnych jednotiek, príp. ich skupín.
- Testovanie systému (angl. system testing). Ako už z názvu vyplýva, ide o testovanie na úrovni celého systému. Na tvorbu testovacích scenárov sa tu využívajú skutočné scenáre, ktorých vykonávanie sa od vyvíjaného systému očakáva.
- Akceptačné testovanie (angl. acceptance testing). Pri tomto testovaní sa pracuje s už vyvinutým systémom, a to spôsobom, pre ktorý bol daný systém vyvinutý. Testovanie prebieha jednak u dodávateľa, ako aj u odberateľa softvéru.

Z V-modelu vyplýva, ktorá vývojová aktivita produkuje podkladové materiály pre ktorú testovaciu aktivitu. Mnoho ľudí sa domnieva, že testovanie je aktivita, ktorá sa začína až po naprogramovaní celého systému. Celý testovací proces tým redukujú iba na vykonávanie, príp. automatizované spúšťanie testovacích scenárov. Z tohto pohľadu sa potom V-model javí ako rozšírený „vodopádový model“, ktorý po ukončení všetkých vývojových aktivít začína s testovacími aktivitami.

V skutočnosti je však testovací proces omnoho komplexnejší. Tak, ako má svoj životný cyklus vyvíjaný softvér, rovnako ho má aj každý testovací scenár, ktorým sa vyvíjaný softvér testuje. Testovanie má niekoľko etáp, ktoré začínajú fázou plánovania, pokračujú návrhom jednotlivých testovacích scenárov až po ich implementáciu a spustenie. Tie fázy testovacieho procesu, ktoré sa netýkajú priamo implementácie a spúšťania testovacích scenárov, je možné vykonávať už v úvodných etapách samotného vývoja softvéru. A práve toto sa snaží presadiť V-model. Návrh testovacej stratégie pre testovanie celého vyvíjaného systému je totiž možné stanoviť už vo fáze špecifikácie systému. Podobne je možné návrh jednotlivých testovacích scenárov (ich ciele, predpokladané vstupy a očakávané výstupy) vykonať už vo vývojovej fáze detailného návrhu bez toho, aby boli jednotlivé testované časti už implementované, a dokonca aj bez toho, aby bol známy programovací jazyk, ktorým sa tieto časti budú vyvíjať.

Takýto prístup so sebou prináša aj jednu veľkú výhodu ovplyvňujúcu samotný proces vývoja softvéru. Keďže sa návrh testovacích scenárov vykonáva zároveň s detailným návrhom systém, často sa stáva, že práve pri tvorbe testovacieho scenára sa odhalí chyba v podkladovom návrhu vyvíjaného systému. Tým vlastne testovací

proces prispieva k zvýšeniu kvality samotného vývoja. Preto je potrebné, aby sa k testovacím aktivitám pristúpilo tak skoro, ako je to len možné, pričom na tento fakt často upozorňuje odborná literatúra [4], [9], [15].

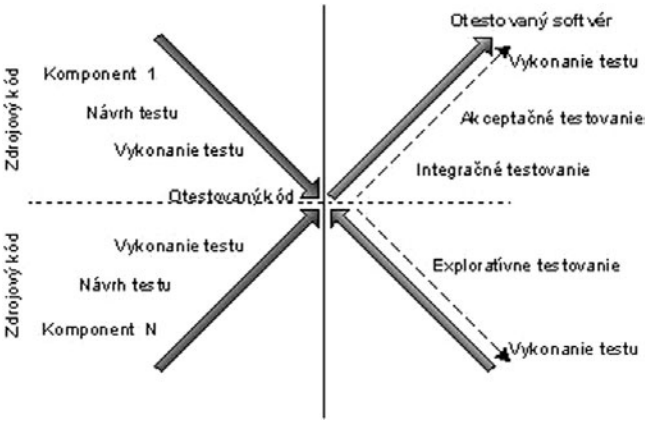
X-model

Tento typ procesného modelu sa nezaoberá aktivitami životného cyklu softvéru. Definuje základné testovacie aktivity, ktoré pri testovaní softvérového produktu treba vykonať, ale nestanovuje presnú časovú súvislosť medzi týmito aktivitami a aktivitami životného cyklu softvéru. Dôvodom je, že podľa zástancov X-modelu [12], [14] nemusí každý softvérový projekt pozostávať z rovnakých etáp. Existujú aj projekty, pri ktorých napr. nie sú vopred zadefinované presné používateľské požiadavky, ale pracuje sa na princípe vytvárania prototypov. Takýmto prístupom sa X-model snaží byť menej reštriktívnym a rozhodnutie o rozložení testovacích aktivít v rámci procesu vývoja prenecháva na používateľa.

X-model rozdeľuje testovacie aktivity do dvoch základných kategórií. Prvou kategóriou je testovanie komponentov, pri ktorom sa testuje funkcionalita základných programových jednotiek a ich spolupráca na úrovni komunikačných rozhraní. Druhou kategóriou je podľa X-modelu testovanie systému, čo predstavuje testovanie integrácie jednotlivých komponentov a ich skupín na základe očakávaného fungovania celého vyvíjaného systému. Z hľadiska už spomínaného všeobecne akceptovaného členenia testovacích úrovní patrí do prvej kategórie testovania podľa X-modelu testovanie funkčných blokov a do druhej kategórie patria integračné a akceptačné testovanie.

Ako vidieť na obr. 2, X-model je vertikálne rozdelený na dve časti, pričom v ľavej časti sa vykonáva testovanie jednotlivých komponentov a v pravej časti modelu sú znázornené aktivity na testovanie celého systému. V prípade, že testy komponentov vykonané v ľavej časti modelu sú úspešné, prechádza proces testovania do pravej časti modelu na testovanie na úrovni celého systému. X-model je zaujímavý aj z toho hľadiska, že naznačuje miesto v organizačnej štruktúre projektu, na ktorom sa daný typ testovania vykonáva. Aktivity v ľavej časti modelu sa vykonávajú vždy vo firme, ktorá daný softvérový produkt vyvíja, pretože nízkoúrovňové testovanie komponentov sa vykonáva zväčša na báze znalosti zdrojového kódu. Aktivity v pravej časti modelu sa môžu vykonávať jednak na výstupnej kontrole dodávateľskej firmy, ako aj priamo u odberateľa daného softvéru.

Ďalšou dôležitou vlastnosťou tohto modelu je aj to, že tvorbu testovacích scenárov na základe vopred definovaných požiadaviek na funkcionalitu nepovažuje X-model za dostatočnú. Testovaný softvérový systém by sa mal podľa tejto domnienky testovať aj exploratívnym spôsobom na báze náhodného výberu testovacích scenárov. To znamená, že po otestovaní systému na základe vopred definovaných používateľských požiadaviek by malo nasledovať testovanie pomocou akcií, ktoré presahujú rámec týchto požiadaviek.

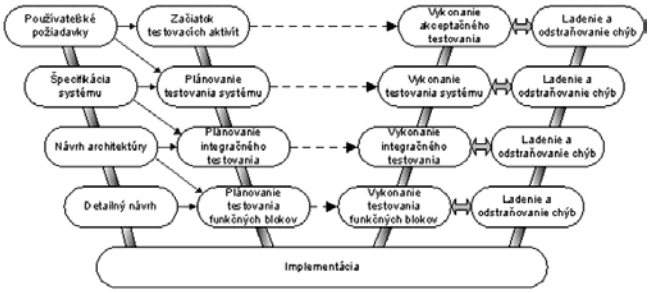


Obr.2 X-model

Osoba vykonávajúca testovanie si v tomto prípade kladie otázku: „Čo sa stane, ak spravím „toto“?“. Cieľom je, aby takto testovaný systém zostal stabilný. Exploratívne testovanie je v X-modeli znázornené v pravej časti modelu v dolnom kvadrante.

W-model

Podkladom pre vytvorenie tohto procesného modelu je V-model, ktorý podľa autorov W-modelu [16] nie je postačujúci z dvoch dôvodov. Prvým je, že testovacia aktivita na určitej úrovni začína až po začatí vývojovej aktivity na tejto úrovni, príp. súčasne s ňou. Podľa W-modelu je totiž možné už na vývojovej úrovni N jednoznačne stanoviť základné črty smeru vývoja na úrovni N+1, takže aj s plánovaním príslušných testovacích aktivít možno začať už na úrovni N+1. Druhým dôvodom je skutočnosť, že V-model sa zaoberá testovacími aktivitami iba na úrovni ich existencie a nedefinuje jednotlivé fázy testovacieho procesu na konkrétnych úrovniach modelu. Preto sa autori W-modelu separátne zaoberajú plánovaním testovania, vykonaním testovania a následne procesom ladenia a odstraňovania chýb, ktorý so samotným testovaním úzko súvisí.



Obr.3 W-model

Ako je znázornené na obr. 3, W-model sa skladá z dvoch vetiev na ľavej strane modelu a dvoch na strane pravej. V prvej vetve na ľavej strane sú znázornené aktivity procesu vývoja softvéru, podobne ako pri V-modeli. K tejto vetve je priradená vetva s tzv. konštruktívnymi aktivitami testovania. Tieto aktivity sa zaoberajú plánovaním testovania a návrhom a tvorbou testovacích scenárov. Šípky smerujúce z vývojových do testovacích aktivít naznačujú, že podkladom pre vykonanie konštruktívnej testovacej aktivity na N-tej úrovni sú jednak výsledky vývojovej aktivity na úrovni N, ako aj na úrovni N-1. V pravej časti W-modelu sa nachádzajú takisto dve vetvy. Prvá z nich zahŕňa činnosti na vykonávanie testov, tzv. deštruktívne testovacie aktivity. Druhá vetva na pravej strane zobrazuje aktivity zamerané na ladenie testovaného softvéru a odstraňovanie chýb. Ich funkcionalita je úzko spätá s vykonávaním testov, pretože tieto činnosti sú iba reakciou na prípadné chyby zistené pri deštruktívnych testovacích aktivitách.

Prínosom W-modelu je bezpochyby to, že oddeľuje plánovacie (konštruktívne) a vykonávacie (deštruktívne) aktivity testovacieho procesu. Pri jednotlivých vývojových aktivitách je síce možné začať s testovaním, ale ide len o začatie konštruktívnych testovacích aktivít. Hotové implementované testy sa dajú vykonať skutočne až po implementácii systému, a práve toto W-model zobrazuje. Slabšou stránkou tohto modelu je však pravá vetva v pravej časti modelu zobrazujúca aktivity vykonávajúce ladenie a odstraňovanie chýb. Spôsob, akým tieto aktivity spolupracujú s ostatnými aktivitami, nie je v modeli dostatočne zobrazený. Po nájdení chyby, napr. pri integračnom testovaní, je totiž mnohokrát potrebné zmeniť návrh architektúry, príp. detailný návrh. Vo W-modeli však takýto návrat k vývojovým aktivitám nie je naznačený.

Ďalšie modely

Existujú modely procesu vývoja softvéru, ktoré síce neboli vytvorené pre potreby plánovania testovacieho procesu, ale napriek tomu testovaniu vymedzujú (implicitne alebo explicitne) jeho presné

miesto v softvérovom projekte. Ako príklady tu budú uvedené „vodopádový model“, „špirálový model“ a koncepcia tzv. „extreme programming“ [1].

„Vodopádový model“ je jedným z prvých a zároveň najznámejších procesných modelov vývoja softvéru. Jeho aktivity nasledujú tak, ako je to znázornené v ľavej časti V-modelu – od používateľských požiadaviek až po implementáciu. Tento model vychádza z toho, že každá aktivita sa môže začať až po ukončení všetkých predchádzajúcich aktivít. Napriek tomu, že tento model explicitne nedefinuje aktivity testovacieho procesu, implicitne s nimi počíta až po ukončení implementácie. Tým sa však výrazne zvyšujú náklady na odstránenie prípadných chýb zistených počas testovania. Tieto náklady sa pritom zvyšujú každou aktivitou vývojového procesu, ku ktorej je potrebné vracať sa v záujme odstránenia nájdenej chyby. Ďalšou nevýhodou je aj to, že sa v praxi zvykne testovanie často zanedbávať v prípade, ak je poslednou aktivitou vykonávanou tesne pred ukončením celého projektu.

Ako už zo samotného názvu vyplýva, „špirálový model“ predstavuje cyklický tvar vývojového procesu. Testovacie akcie sú tu definované explicitne a sú rozdelené do jednotlivých kvadrantov modelu. V rámci „špirálového modelu“ je definované testovanie funkčných blokov, integračné a aj akceptačné testovanie. Nevýhodou je však to, že podobne ako pri „vodopádovom modeli“, je začiatok vykonávania testovacích aktivít plánovaný až po ukončení implementácie.

Napriek tomu, že sa koncepcia „extreme programming“ nezaoberá testovacím procesom, prináša nový spôsob spolupráce medzi vývojom softvéru a jeho testovaním. Tento prístup k vývoju softvéru nepoužíva špecifikácie, ale presadzuje myšlienku, že pred implementáciou každej programovej jednotky je potrebné najskôr vytvoriť test, ktorý bude túto jednotku testovať. Takto vopred vytvorené testy slúžia čiastočne ako špecifikácia vytváranej funkcionality. Po ukončení určitej skupiny funkcií, príp. celého programového modulu sa potom pomocou týchto testov sleduje samotná funkcionalita vytvoreného celku.

Zhrnutie

Ako už bolo spomenuté, najstarším a zároveň najznámejším procesným modelom v oblasti testovania softvéru je V-model. Tento model ako prvý stanovil začiatok testovacích aktivít nie až po ukončení implementácie, ale paralelne so začiatkom aktivít samotného vývojového procesu. Jeho kritici mu vyčítajú prílišnú reštriktívnosť, pretože očakáva rovnaké vývojové aktivity pre každý softvérový projekt. V skutočnosti však V-model iba zvýrazňuje to, na ktorých testovacích aktivitách je možné pracovať počas jednotlivých vývojových aktivít. Nepredpisuje množstvo ani typ práce, ktorú je potrebné v rámci konkrétnej aktivity vykonať, vďaka čomu sa jeho flexibilita zvyšuje. V-model je vhodnejší skôr pre organizáciu a riadenie väčších a komplexnejších softvérových projektov.

X-model by mal predstavovať flexibilnejšiu alternatívu k V-modelu. Otázkou však je, nakoľko má jeho flexibilita dopad na precíznosť riadenia projektu. Problémom je totiž vykonávanie náhodného testovania naznačeného v pravom dolnom kvadrante tohto modelu. Pri testovaní softvérového systému vzhľadom na presne definované používateľské požiadavky je nutný rozsah testovania jasný. Pri náhodnom testovaní zaoberajúcim sa otázkou „Čo sa stane, ak spravím „toto“?“ sa však automaticky vynára otázka „Koľko náhodných testov je “dosť“?“. Napriek tomu však X-model predstavuje alternatívu modelu testovacieho procesu, a to najmä pre menšie softvérové projekty. Pri väčších projektoch je totiž princíp náhodného testovania nezvládnuteľný.

Zaujímavé rozšírenie V-modelu predstavuje tzv. W-model. Jeho výhodou je presnejšie rozdelenie testovacích aktivít na aktivity konštruktívne a deštruktívne. Tým naznačuje, že testovacie aktivity (v tomto prípade konštruktívne) je možné skutočne vykonávať pred samotnou implementáciou testovaného systému. Kedže V-model takéto rozdelenie explicitne neposkytuje, mnoho používateľov si ho nesprávne zamieňa s „vodopádovým modelom“.

Koncepcia „extreme programming“ predstavuje síce veľmi flexibilný a jednoduchý prístup k tvorbe testov, vhodná je však iba pre veľmi malé projekty alebo časti projektov.

Záver

V tomto článku boli predstavené najznámejšie a v praxi najviac akceptované modely pre plánovanie a riadenie procesu testovania softvéru. Napriek tomu, že každý z nich má svoje nevýhody, zároveň do tejto oblasti prináša aj nové myšlienky. Žiadny z týchto modelov však nie je univerzálny pre každý projekt. To, ktorým z týchto modelov sa bude softvérová firma riadiť, záleží vždy od veľkosti a komplexnosti konkrétneho projektu, ako aj od stupňa spoľahlivosti, ktorý sa od vyvinutého softvérového systému bude požadovať.

Literatúra

[1] BECK, K., FOWLER, M: Planning Extreme Programming. Addison-Wesley 2000.

[2] BERTOLINO, A.: Software Testing Research and Practice. 10th. International Workshop on Abstract State Machines 2001.

[3] BERTOLINO, A.: Knowledge Area Decription of Software Testing – version 0.9. Instituto di Elaborazione della Informazione 2002.

[4] BINDER, V.: Testing Object-Oriented Systems: A Status Report. American Programmer 1994.

[5] BOEHM, B. W.: Software Engineering Economics. Englewood Cliff: Prentice Hall 1981.

[6] BOEHM, B. W.: A Spiral Model of Software Development and Enhancement. IEEE Computer 1988.

[7] BRÖHL, A. P., DRÖSCHEL, W.: Das V-Model – Der Standard für die Softwareentwicklung mit Praxisleitfaden. Oldenbourg Verlag, Wien 1993.

[8] GOLDSMITH, R. F., GRAHAM, D.: The Forgotten Phase. Software development, July 2002.

[9] GOLDSMITH, R. F., GRAHAM, D.: This or That – V or X?, Software Development, August 2002.

[10] JACOBSON, I., CHRISTERSON, M., P. Jonsson, G. Övergaard: Object-Oriented Software Engineering. Addison-Wesley 1994.

[11] JACOBSON, I., BOOCH, G., RUMBAUGH, J.: The Unified Software Development Process. Addison-Wesley 1998.

[12] LUCAS, I.: Testing Times in Computer Validation. In: Journal of Validation Technology, February 2003.

[13] MARICK, B.: The Craft of Software Testing. Prentice Hall PTR, New Jersey 1991.

[14] MARICK, B.: New Models for Test Development. www.testing.com

[15] MCGREGOR, J., D., SYKES, D. A.: A Practical Guide to Testing Object-Orienred Software. Addison-Wesley 2001.

[16] SPILLNER, A.: The W-Model - Strengthening the Bond Between Development and Test. University of Applied Sciences, Germany 2001.

Roman Nagy

microTOOL GmbH
Berlin, Germany
e-mail: Roman.Nagy@microTOOL.de