Zabezpečenie kvality a testovanie Web Aplikácií

MILAN BARAN

Slovenská technická univerzita
Fakulta informatiky a informačných technológií
Ilkovičova 3, 842 16 Bratislava
milan[.]baran[zavináč]gmail[.]com

Abstrakt. Rýchly vývoj vo svete softvéru zmenil tvár Internetu z malého množstva web stránok na veľký komplex web aplikácií, ktoré dnes poskytujú služby rôznych druhov a sú rovnocennou alternatívou tradičných softvérových produktov. Zvyšujúcou sa zložitosťou Web aplikácií vznikala potreba vývoja nových techník testovania tohto druhu softvéru, ktorá však nebola doposiaľ uspokojená. Web aplikácie sú stále testované tradičnými metódami, ktoré sú v mnohých prípadoch neefektívne a nákladné. Cieľom tejto eseje je poskytnúť pohľad autora na problematiku zabezpečenia kvality Web aplikácií. Pojednáva o zraniteľnosti nekvalitného softvéru a príčinách jeho vzniku. Chce poukázať na neefektívne testovacie techniky súčasnej doby. Rozoberá možnosti testovania Web aplikácie ako interaktívneho, viac-platformového, dynamického systému. Nakoniec sa snaží poskytnúť riešenie na zlepšenie efektívnosti testovania.

Úvod

Súčasná moderná doba sa nesie v duchu rýchleho napredovania technológií. Zo značnej časti sem patrí aj skupina Web aplikácií, ktorá si čoraz viac buduje stabilnejšie miesto na trhu softvérových produktov. Tento trh je nasýtený množstvom komerčných ako aj voľne dostupných projektov. Častokrát sa svojou kvalitou a službami rovnajú. Užívateľ z takejto ponuky preferuje výber ekonomicky výhodnejších riešení, čo núti komerčné projekty zlepšiť kvalitu ich služieb alebo priniesť na trh niečo celkom nové, čo nemá obdobu v podobe voľne dostupného produktu. Pre softvérové inžinierstvo tu vzniká požiadavka na zrýchlenie procesu vývoja softvéru, za účelom predbehnutia konkurencie. Z druhej stránky je tu tiež požiadavka zabezpečiť dostatočnú kvalitu.

V tejto eseji otváram problematiku kvality softvéru, ďalej identifikujem tradičný prístup k vývoju softvéru ako nedostačujúci a pojednávam o alternatívnych riešeniach súčasnej doby. Neskôr sa zameriavam na testovanie ako celok, ktorý sa dá vylepšiť cestou automatizácie niektorých testovacích procesov.

Zabezpečenie kvality a testovanie Web Aplikácií, s. 1-7.

Kvalita

Kvalita sa dá chápať rôzne. Všeobecne sa chápe, že produkt je kvalitný, ak slúži svojmu účelu dlho a bezproblémovo. Dnes je tento trend značne poznačený rýchlosťou vývoja čo so sebou prinieslo viac nekvalitných produktov. Produkt, ktorý si kúpim dnes môže už byť zajtra zastaralý. Tento fenomén dnešnej doby si určite všimol každý z nás. Taktiež tento fakt núti každého výrobcu prehodnotiť svoje priority. V minulosti bolo trendom, že meno firmy sa budovalo na kvalite produktov. No dnes je tu trend noviniek a imidžu, ktorý so sebou prináša novú politiku pri výrobe a zabezpečení kvality. Produkty sú vytvárané s cieľom krátkodobého používania, pretože sa v krátkom časovom horizonte počíta s nástupom novej generácie. Výrobcovia si uvedomujú, že inováciu vo forme novej generácie produktov môže priniesť aj konkurencia a preto sa snažia uprednostňovať rýchlosť vývoja pred kvalitou. Týmto nechcem podporovať znižovanie kvality, chcem len upozorniť na tento fakt. Avšak na trhu je stále prítomnosť firiem, ktoré majú možnosti prispôsobiť sa rýchlosti vývoja a pri tom zabezpečiť dostatočnú kvalitu svojim produktom. Ide však o náročný proces, za ktorý si treba priplatiť.

Softvérový produkt ako taký je nehmotný a neexistujú presne definované kritériá na popis jeho kvality. Norma ISO/IEC 9126, bola vytvorená za účelom bližšej definície kvality softvéru. Tvorí ju 6 charakteristík, ktoré sú ešte bližšie špecifikované v niekoľkých znakoch (viď. **Obr. 1**). Tieto charakteristiky slúžia na zlepšenie predstavy o tom čo má kvalitu softvéru určovať. Mal som možnosť stretnúť v praxi zákazníkov, ktorí nemali tušenie, aké znaky by mal kvalitný softvér spĺňať. Zaujímali ich len časti, ktoré zaujímajú väčšinu bežných užívateľov. Teda kvalita web aplikácií by sa dala rozlíšiť na dve skupiny a to na kvalitu z pohľadu užívateľov a na kvalitu z pohľadu vývojárov, ktorí vedia, čo všetko sa skrýva za bezproblémovou prevádzkou kvalitných web aplikácií.

Obr. 1. Charakteristiky kvality podľa ISO/IEC 9126.

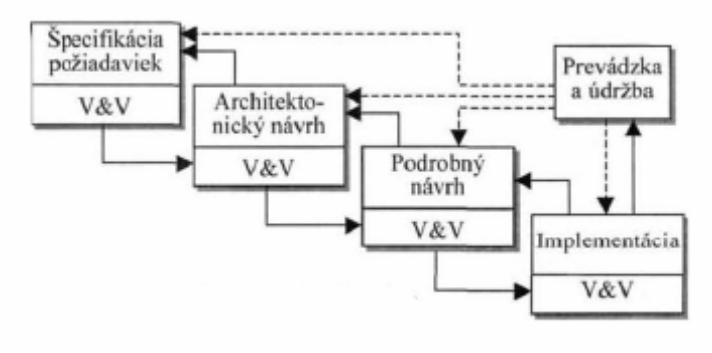
Web aplikácie ako softvérový produkt sa nevyhol problému rýchlych zmien vo vývoji, ktoré som opisoval na začiatku v tejto časti. Tu však narážame na problém v tom, že Web aplikácia nie je reálny produkt, ktorý sa distribuuje do rúk užívateľa. Je to skôr vzdialená služba, ktorá slúži potrebám väčšej skupiny užívateľov. Je zrejmé, že takýto softvér s rýchlosťou vývoju musí zabezpečiť aj svoju kvalitu. Na manažéroch kvality teda zostala neľahká úloha zabezpečiť dostatočnú výkonnosť, odozvu a dostupnosť systému[8] so zameraním sa na stranu užívateľov a nové metodológie vývoja a testovania softvéru so zameraním sa na stranu vývojárov.

Dostupnosť je vyjadrená percentuálnou hodnotou, ktorá je závislá od času, v ktorom je Web aplikácia plne dostupná pre zákazníkov. Rôzne druhy aplikácií sa líšia svojou politikou v tomto smere. Aplikácie závislé na počte návštevníkov kladú vyššie nároky na dostupnosť oproti aplikáciám, kde dostupnosť nieje taký kritický problém. Ďalším aspektom je dostupnosť v určitom čase, kde sa kladie dôraz na maximálny zvládnutý počet užívateľov v tom istom čase. V takomto prípade môže nastať situácia, že užívateľ bude jednoducho odmietnutý. Osobne môžem povedať, že takáto situácia dokáže frustrovať natoľko, že užívateľ po niekoľkých neúspešných pokusoch odchádza zo stránky a prehlbuje nedôveru k takémuto produktu.

Ďalším kritickým bodom v zabezpečení kvality je čas odozvy. Definuje sa časom medzi požiadavkou na systém zo strany užívateľa a doručením odpovede. Z ďalšieho pohľadu ide o doručenie vygenerovanej HTML stránky k užívateľovi. Treba si uvedomiť, že tento čas je závislý nielen na rýchlosti vygenerovania HTML stránky ale aj na rýchlosť internetového spojenia medzi poskytovateľom a adresátom služby. Odozvu tiež môže zhoršovať použitie služieb tretej strany, ako geografické mapy, interaktívne média, elementy sociálnych sieti, reklamy, informačné transparenty a iné. V prípade dlhej odozvy užívateľ stráca interakciu so systémom, čo vedie k nespokojnosti a strate dôvery.

Proces vývoju softvéru

Tradičný vývoj softvéru je založený na vzore vodopádového modelu, ktorý pozostáva zo 4 časti (vid. **Obr. 2**). Základným predpokladom pre efektívne využitie takéhoto modelu je nemennosť základnej špecifikácií a následného návrh, pretože vo fáze implementácie sa so zmenami už neráta. Čoraz viac a viac sa však stáva, že takýto softvér po dokončení už nespĺňa aktuálne požiadavky zákazníka a aj napriek vysokej kvalite je systém nepoužiteľný pre súčasné potreby zákazníka. V iných prípadoch si zákazník môže rozmyslieť prvotnú špecifikáciu alebo jednoducho na niečo zabudol. Tento model nie je natoľko flexibilný, aby dokázal zohľadňovať nové požiadavky zákazníka, pretože časová náročnosť tohto modelu vývoja softvéru je príliš vysoká na uspokojenie jeho momentálnych potrieb. Tento nastupujúci trend som už opísal v predchádzajúcej časti a tu sa zobrazili jeho dopady na tradičnú metódu vývoja softvéru, ktorá tempom nestačí na vývoj moderných softvérov ako napríklad Web aplikácie.



Obr. 2 Vodopádový model [3].

Uvažujme, že softvér úspešne prešiel fázou testovania a dostal sa do fázy nasadenia, v ktorom bude udržiavaný do konca svojho životného cyklu. Tu nachádzam ďalšie problémy z hľadiska použitého modelu. Web aplikácie sú význačne svojou funkcionalitou, ktorá sa mení veľmi často v závislosti od úspešnosti projektu. Tu je kladená zložitá úloha na tím vývojárov, ktorí dotvárajú novú funkcionalitu s absenciou priebežnej kontroly výsledkov. Úloha je o to ťažšia pretože ide o nasadený systém s reálnymi informáciami, ktoré nesmú byť zmenené po aktualizácií projektu. Testovanie vo fáze implementácie je ponechané len na schopnostiach a skúsenostiach programátora, ktorý je častokrát nútený meniť staré funkcie a jeho abstraktné myslenie nie je natoľko dokonalé, aby dokázal zvážiť všetky aspekty dopadu zmeny na súčasný systém. Z vlastnej skúsenosti môžem povedať, že neprítomnosť priebežného testovania je tu viac než kritická. Zatiaľ však ide len o vnútro-projektové problémy, novú funkcionalitu špecifikuje zákazník a ten sa k reálnym výsledkom dostane až v priebehu testovania alebo po ňom. Tu môže nastať ďalší problém s rozdielom medzi reálnou funkcionalitou a požadovanou funkcionalitou, čo vedie k ďalšiemu predlžovaniu, strate dôvery a zisku. Taktiež z vlastnej skúsenosti môžem povedať, že tento trend nieje nejako neobvyklý. V niektorých prípadoch sa dokonca stalo, že programátor si bol, po vykonaní elementárnej zmeny, natoľko istý funkcionalitou, že preskočil testovanie a chybu odhalil až zákazník, čo je absolútne najhorší prípad zlyhania manažmentu kvality a povedal by som, absencie dobrých a efektívnych návykov testovať.

Našťastie existuje mnoho moderných vzorov na podporu vývoja softvéru, ktoré sú viac či menej vhodné na vývoj Web aplikácií v závislosti od prostredia vývoja a určený rozsah projektu. Medzi bežne používane metódy patria agilné vývojové vzory [1] navrhnuté pre prácu väčších vývojových tímov. Ďalším veľmi využívaným vzorom je tiež rapídny vývoj aplikácie, od ktorého sa derivujú dva ďalšie vzory a to rapídny vývoj programu a rapídny vývoj systému [7]. V prvom z uvedených vzorov musí skupina programátorov v čo najkratšom čase pochopiť požiadavky a v prísne určenom čase vyprodukovať vysoko kvalitný kód. Tento druh vývoja je príznačný výrazom „programovanie nad dokumentáciu“, preto by mal kód odzrkadľovať patričnú kvalitu.

Komunikácia medzi užívateľom a programátorom v tomto prípade nieje nutná, keďže ide o presne definovanú úlohu.

Rapídny vývoj systémov je práve naopak, založený na interakcií medzi užívateľom a tímom vývojárom. Projekt začína len so základnou ideológiou systému a po každej ukončenej etape sa konzultuje vytvorený prototyp s užívateľom. Projekt nemá pevne stanovené ciele, ale postupne sa vyvíja a požiadavky sa upravujú v závislosti od potrieb užívateľa. Tím teda musí preukázať svoje technické aj komunikačné schopnosti na dosiahnutie úspešného ukončenia projektu.

Všetky menované vzory sú vhodné na tvorbu Web aplikácií a môžem spomenúť niektoré ďalšie ako extrémne programovanie alebo riadenie vývoja testovaním. Nieje mojim cieľom ich všetky rozobrať, chcem len vzbudiť myšlienku, že rozdelenie systému na menšie časové úseky s možnosťou spätnej väzby alebo možnosťou zmeny špecifikácie je viac než nevyhnutné pre vývoj Web aplikácií. Nakoniec, je na skúsenostiach a úsudku každého manažéra kvality alebo vedúceho tímu, pre ktorý vývojový vzor sa rozhodne. Je potrebné však venovať dostatočnú pozornosť pre podporu testovania už pri vývoji. Myslím si, že jej absencia napomáha k vzniku zlých až žiadnych testovacích návykov čo spôsobuje problémy nielen na konci celého projektu ale k prieťahom jednotlivých etáp vývoja.

Testovanie naše každodenné

Z môjho pohľadu je testovanie dôležité, pretože potrebujem validovať svoju prácu. Na to, aby som mohol, ako vývojár, prehlásiť o vykonanej zmene alebo novovytvorenej funkcionalite to, že je funkčná, nestačí len môj osobný názor. Je potrebné aplikáciu podrobiť sade testov, ktoré sa tykajú vykonanej zmeny kódu, prípadne vytvoriť nové. Už samotná testovacia požiadavka môže byť v niektorých prípadoch zložitá, keďže v mnohých prípadoch sú aplikácie svojim rozsahom gigantické a predstava o testovaní zmenenej časti alebo dôležitej časti aplikácie môže naháňať hrôzu. Naviac v prípade vývoja projektu v tíme je potrebné komplexné testovanie všetkých častí. Na prvý pohľad vidieť, že je potrebné vytvoriť komplexnú infraštruktúru čiastkových testov na otestovanie celého projektu. Tu sa naskytá otázka, je možné vytvoriť testovacie scenáre, ktoré by sa dali automatizovane spúšťať počas vývoja programu a tak sledovať či aplikácia zodpovedá alebo nezodpovedá požiadavkám z návrhu. Metodológii k tejto problematike je viac než dosť, boli uvedené v predchádzajúcej časti. V tom nevidím problém. Problém však vidím na druhej strane, kde bola v minulosti absencia podporných prostriedkov testovania softvéru, typu jUnit, ako softvér na podporu testovania metódou bielej skrinky. No v súčasnosti sa objavuje čoraz viac a viac voľne dostupných kvalitných podporných prostriedkov na testovanie, ako Canoo WebTest [6,4] alebo Google WebDriver [5]. Tieto programy ponúkajú programátorovi širokú škálu možností pri vytváraní testovacích scenárov a manažérom kvality na vybudovanie kvalitnej testovacej infraštruktúry. Obsahujú podporu pre rôzne techniky testovania ako „Model based“ testovanie, „Data driven“ testovanie, Capture/replay testovanie, automatické testovanie skriptom alebo regresívne testovanie a možnosti pre ďalší vývoj testovacích techník do budúcnosti[2,9].

Predsa, testovanie je stále opakujúca sa činnosť, z pohľadu programátora ide o rutinnú pracú vykonávajúcu stále dokola a v niektorých prípadoch už podvedome podceňuje niektoré časti testovania, čo vedie k chybám. Táto nudná práca by mala byť prenechaná automatom, ktoré na to majú lepšie predpoklady. Týmto nechcem povedať, že testovanie programátora by malo byť nahradené automatizovanou činnosťou. Pravé naopak, programátor je ten čo vynakladá inteligentnú činnosť na tvorbu testovacieho scenára a výsledok mu má poskytnúť obraz o tom akým smerom sa jeho projekt uberá. Tento systém tiež môže využiť pri testovaní, ktoré prebieha v niekoľkých krokoch a je potrebné odtestovať len koncovú časť. Programátor si môže vytvoriť scenár, ktorý ho dovedie až k určitej časti a vyhne sa tak zdĺhavému „preklikávaniu“ cez predchádzajúce časti.

Takéto poloautomatické testovanie by malo poskytovať všetkým zúčastneným stranám na projekte objektívny nástroj na stav projektu a rýchlu odpoveď na otázku, či sa vyvíjaný projekt uberá správnou cestou. Jeho výstupom by mal byt všetkým zrozumiteľný dokument, ktorý jasne ukazuje čo funguje, čo nefunguje a pri problémoch umožní vystopovať ich príčiny.

Záver

V dnešnej dobe, kde sa mnoho Web aplikácií jednoducho nedokončí. Kde konkurencia nespí ani v noci. Kde čas hrá rozhodujúcu úlohu. Kde zákazník je váš pán. Je potrebné presedlať zo starých osvedčených techník naučených sa ešte za študentských čias a vydať sa smerom moderných agilných techník riadenia softvéru, ktoré prinášajú manažérom kvality novú nádej udržať sa na trhu a nové metodiky testovania. Ktoré poskytujú vývojárom plnohodnotnejšie a pohodlnejšie testovanie a dohľad nad vyvíjaným softvérom. Z nemalej časti šetria čas a náklady a tvorivý potenciál vývojárov. Tento problém som sa snažil pojať pohľadu vlastných skúseností, ktoré boli poznačené neodborným prístupom zo strany manažmentu kvality.

Použitá literatúra

1. Agile Software Development: (navštívené: 10/17/2008) http://en.wikipedia.org/wiki/Agile_software_development
2. Bertolino, A.: *Software Testing Research: Achievements, Challenges, Dreams.* FOSE’07, IEEE Internet Computing, 2007.
3. Bieliková M.: *Softvérové inžinierstvo – princípy a manažment*, Vydavateľstvo STU, Bratislava, 2000, p. 95.
4. Canoo WebTest: http://webtest.canoo.com (navštívené: 10/17/2008)
5. Google WebDriver: http://code.google.com/p/webdriver/ (navštívené: 10/17/2008)
6. Guillemot, M.: *Web Testing Made Easy*. OOPSLA'06, Portland, Oregon, USA, 2006,p. 22-26.

7. Howard, A.: *Rapid Application Development: Rough and Dirty or Value-for-Money Engineering?* Communication of the ACM, Vol. 45, No. 10, 2002,p. 27-29.
8. Menascé, D.A.: *Load Testing of Web Sites*. IEEE Internet Computing, 2002,p.70-74.
9. Sanchez, A., Vega, B., Gonzalez, A., Jackson, G.: *Automatic Support for Testing Web-Based Enterprise Applications*. ACM SE'06, Melbourne, Florida, USA, 2006, p. 10-12.

Annotation

Quality assurance and testing of Web application

Fast evolution in software's world changes character of internet from small amount of Web pages into big complex of Web applications, they provide services of various kinds and they are equivalent alternative to traditional software products, these days. Incremental complexity of Web applications results to need of new testing methods of development this kind of software, it's not satisfied yet. Web applications are still testing with traditional methods, they are ineffective and expensive in many cases. Aim of this essay gives an author's view to problems of software quality assurance. It deals with vulnerableness of low-quality software and reasons of their origin. It wants to point out ineffective testing techniques of nowadays. It thinks about possibilities of Web applications testing as interactive, multi-platform, dynamic system. At the end it tries to give a solution to improve efficiency of testing.