KAPITOLA 4

Co je testování a kdo jej provádí

V této kapitole:
- Co je a není testování softwaru
- Proč je testování důležité
- Základní axiomy testování softwaru
- Proč programátor nemá testovat svůj kód
- Mýty a pravda o testování
- Testeři a programování
- Role v testovacím týmu

V této kapitole se budeme zabývat již samotným testováním z pohledu jeho vymezení jako pojmu, významu a základních principů, z nichž některé již byly nastíněny v kapitolách předchozích. Podíváme se na to, jaký je současný stav v oblasti testování a řízení kvality u nás, jaké znalosti z oblasti vývoje by měl tester mít, jakou skladbu má typický testovací tým a také na časté omyly a polopravdy, se kterými se v chápání testování a jeho role dnes stále ještě můžeme setkat.

Co je a není testování softwaru

Přestože intuitivně tušíme, co je podstatou testování, pro bližší porozumění potřebujeme tento pojem blíže vymezit. Definovat testování softwaru není úplně snadné a lze říci, že co publikace a autor, to více či méně odlišný názor na to, co testování je (a co není).

V této knize definujeme testování softwaru v užším smyslu jako *proces řízeného spouštění softwarového produktu s cílem zjistit, zda splňuje specifikované či implicitní potřeby uživatelů*. Slovo „spouštění" je záměrně zvýrazněno, protože čistě z tohoto pohledu nejsou metody statické analýzy testováním, a – jak jsme již dříve uvedli – nepoužíváme proto v této knize označení „statické testování". Z širšího pohledu však testování zahrnuje i statickou analýzu čili zkoumání dílčích produktů – požadavků, zdrojového kódu, modelů a podobně.

Testování je řízený proces, neboť musí být prováděno vždy s určitým záměrem, předchází mu plánování ve formě návrhu testů a po něm následuje jejich vyhodnocení.

Z uvedené definice tak vyplývá, že *testování zkoumá softwarový produkt, čímž získává informace o jeho kvalitě*, kterou chápeme jako stupeň naplnění požadavků a přání uživatelů. *Obsahem testování je tedy především sběr a analýza informací*, přičemž nalezené defekty jsou z tohoto pohledu vedlejším produktem této činnosti.

Některé – zejména starší - definice přímo uvádí, že jediným cílem testování softwaru je nalézat defekty. To však platilo v případě, kdy bylo prováděno čistě se záměrem prokázat chybovost produktu. Dnes však testování slouží především jako nástroj k získání informací o tom, zda produkt vyhovuje zamýšlenému použití – a to může samozřejmě i přes přítomnost odhalených defektů.

Poznámka: Ovšem stejně tak může naopak dojít k tomu, že aplikace s nulovým počtem nalezených defektů nebude akceptována, neboť požadavky nezachytily správně potřeby uživatelů.

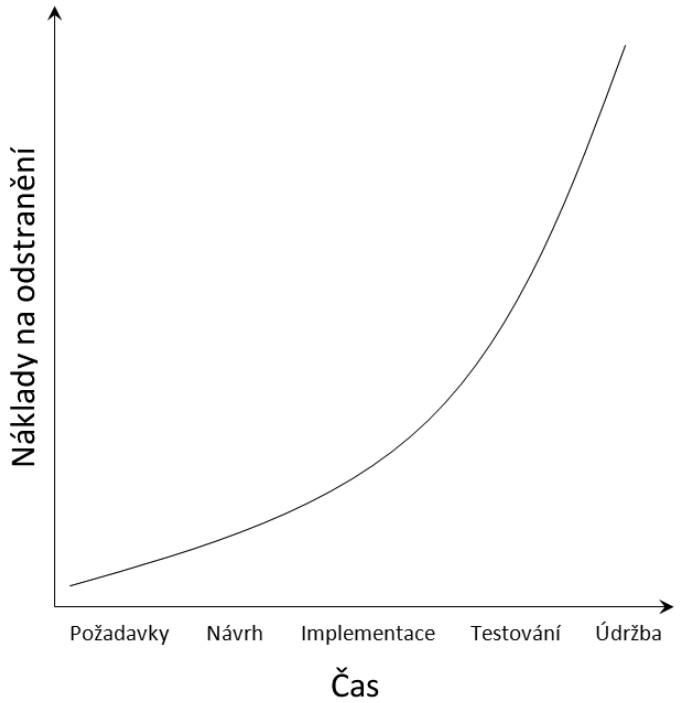
Výše naznačený posun v cílech, které testování sleduje, můžeme vidět na 5 fázích/úrovních vyspělosti testovacího procesu (respektive individuální mentality testera a jeho přístupu k testování) podle Borise Beizera:

- *Úroveň 0: Mezi testováním a laděním (debuggingem) není rozdíl.* Testování je chápáno jako aktivita pomáhající odstraňovat defekty, bez přílišného odlišení od ladění.
- *Úroveň 1: Smyslem testování je prokázat, že software funguje.* Testování má demonstrovat funkčnost softwaru a soulad se specifikacemi – zaměřeno na demonstraci.
- *Úroveň 2: Smyslem testování je prokázat, že software nefunguje.* Testování je způsobem, jak nalézat implementační defekty, nesoulad se specifikací. Zaměřeno na destrukci.
- *Úroveň 3: Smyslem testování není prokazování ničeho konkrétního, ale snížení rizika.* Role testování je rozšířena: kromě kódu se zaměřuje i na požadavky a návrh a stává se součástí celého cyklu vývoje softwaru.
- *Úroveň 4: Testování je duševní disciplína vedoucí k produkci softwaru s nízkým rizikem.* Tato úroveň již chápe testování a ostatní aktivity řízení kvality jako aktivity *preventivní* a především se tak systematicky zaměřuje na předcházení vzniku defektů v požadavcích, návrhu a konečně i kódu.

Protože tato problematika je pochopitelně mnohem rozsáhlejší, v několika bodech níže stručně uvádíme některá důležitá témata, kterými se detailněji zabýváme na jiných místech této knihy:

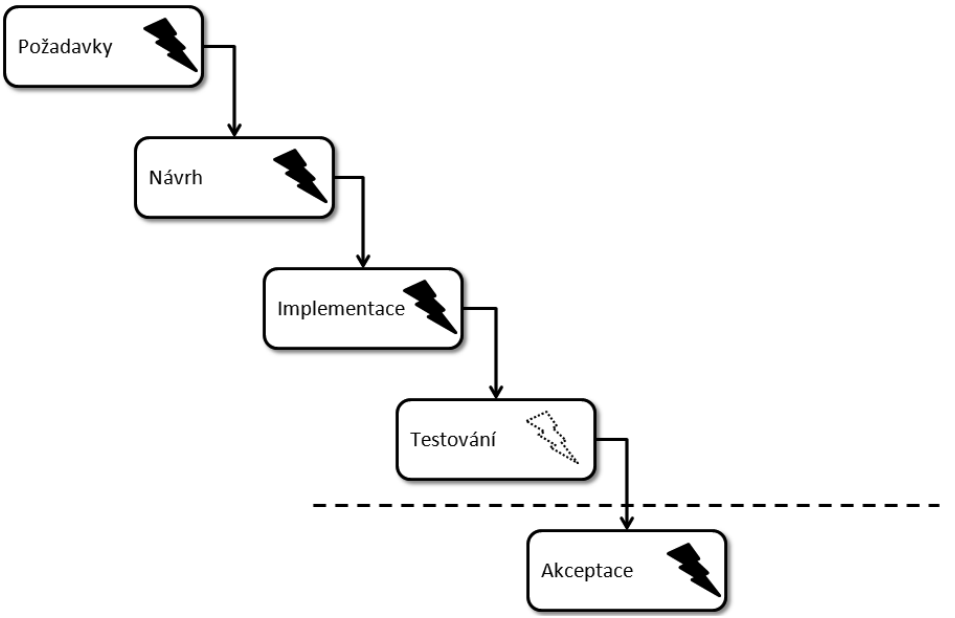
- Testování je jednou z aktivit spadajících pod řízení kvality (*Quality Control*).
- Testování není synonymem pro zajišťování kvality (*Quality Assurance*). Naopak, zajišťování kvality by mělo „prostupovat" všechny procesy vývoje systému, kam patří i samotné testování. Stanovuje metodiky, standardy, metriky a snaží se vést ke zlepšení procesů, a tak vyšší kvalitě produktu.
- Testování samo o sobě přímo nezvyšuje ani nezajišťuje kvalitu produktu, poskytuje však pro tuto činnost potřebné vstupy.
- Testování je závislé na kontextu nejen vyvíjeného produktu, ale i řady dalších faktorů, jako je vývojový model, úroveň testování, omezení zdrojů, akceptované riziko a tak dále.

- Testování je potencionálně nekonečný proces, a o jeho zastavení tak musí být rozhodnuto na základě uvážení různých faktorů, jako je jeho postupně se snižující efektivita, časové a rozpočtové omezení, provedení všech klíčových testů a podobně.
- Testování je kompromisem mezi kvalitou a časovým a rozpočtovým omezením – upřednostňování jednoho cíle má pochopitelně negativní dopad na zbylé dva.
- I testy by měly být otestovány, protože testeři chybují stejně jako ostatní. Vzájemná revize vytvořených testů mezi testery je často velmi efektivní.
- Zátěžové testování by nemělo být prováděno v posledních fázích vývoje produktu, přestože jde o poměrně rozšířený koncept. Identifikace problematických míst v kódu a jejich následná optimalizace by naopak měla probíhat průběžně během vývoje produktu.
- Testování a případné ostatní aktivity řízení kvality by měly začínat, co nejdříve je to možné, tedy již ve fázi specifikace požadavků. Zatímco v této fázi vyžaduje odstranění problému minimální úsilí, ten samý problém zjištěný až po nasazení systému do prostředí zákazníka může způsobit velké ztráty. Platí tedy, že čím dříve je defekt odhalen, tím nižší jsou náklady na jeho odstranění.



Obrázek 4.1 Vztah mezi fází, ve které je defekt identifikován, a cenou za jeho odstranění

Výše uvedené si lze demonstrovat na příkladu: Uvažujme situaci, kdy analytik nesprávně zachytí požadavek a tím do specifikací zanáší defekt. Následně je ze specifikace odvozen návrh systému a podle toho je i provedena implementace. Verifikace defekt samozřejmě neodhalí, neboť testy jsou navrhovány oproti chybné specifikaci, a tak se defektní systém může snadno dostat až k zákazníkovi, který v rámci validace rychle zjišťuje, že produkt nesplňuje očekávání.



Obrázek 4.2 Propagace defektu vzniklého ve fázi analýzy požadavků

Proč je testování důležité

Vývoj softwaru je velmi rozdílný proces oproti výrobě jiných produktů, neboť každý systém je odlišný a většinou vytvářený na míru konkrétnímu zákazníkovi nebo určité cílové skupině, pokud máme na mysli produkt určený pro pultový prodej. Variabilita je skutečně obrovská, a tak není překvapivé, že použité technologie, osvědčené postupy a lidé jako členové týmu s jejich znalostmi a dovednostmi mohou bez problému pracovat na jednom projektu, avšak už se nemusí hodit pro druhý. Z důvodu jedinečnosti každého projektu – a samozřejmě proto, že všichni lidé chybují – není možné jednoduše stále zlepšovat proces vývoje až do jeho ideální formy, produkující perfektní a bezchybný software. Testování je tak nezbytnou součástí každého projektu – bez něho bychom si nemohli být jisti kvalitou vyvíjeného produktu.

I přes zřejmý význam testování a jeho vliv na kvalitu je to právě tato činnost, která je nezřídka některými výrobci softwaru poněkud podceňována. Podívejme se tedy blíže na dopady, jaké pro výrobce testování či naopak netestování softwaru může mít.

V první řadě zmiňme roli testování při řízení rizik projektu. Je-li testování prováděno nesprávně, pak jeho výstupy ve formě informací o produktu budou nevalidní či nedostatečné, což může vést ke špatnému zhodnocení rizik a k následným chybným rozhodnutím projektového managementu. Je tak pochopitelné, že s kritičností vyvíjeného softwaru roste potřeba minimalizace rizik a případných, často velmi vysokých, ztrát. To lze však jen obtížně realizovat, není-li na procesy zajišťování a řízení kvality kladen dostatečný důraz.

Poznámka: Poměrně častým trendem především u větších společností je snaha minimalizovat potencionální riziko selhání vyvíjených produktů bez velkých počátečních investic do testovacích oddělení, takže využívají služeb externích dodavatelů specializujících se právě na testování či celkově na aktivity související se zajišťováním kvality.

Je-li uvolněna verze obsahující defekty v míře vyšší, než je pro obdobné systémy obvyklé (nebo očekávané zákazníkem), výrobce utrpí jednak škodu materiální ve formě finančních ztrát způsobených nákladnými opravami či pokutami, ovšem poškozována je i jeho reputace. To může mít za následek nejen ukončení spolupráce se stávajícími zákazníky, ale z dlouhodobého hlediska především nezájem potencionálních nových zákazníků či obchodních partnerů, kteří nechtějí být spojováni s výrobcem nekvalitních produktů. Totéž však platí také o zaměstnancích výrobce – kdo by chtěl pracovat ve společnosti, jejíž produkty jsou známé svou chybovostí, a tedy zřejmým nedostatečným zájmem o kvalitu, potažmo pak o samotné zákazníky? Schopní pracovníci se zde o práci jistě ucházet nebudou a ti, kteří sami odejdou, budou jen stěží svého bývalého zaměstnavatele chválit.

Je však třeba zmínit výjimky, kdy je produkt cíleně vypuštěn dříve i přes to, že nebyl dostatečně otestován a obsahuje pravděpodobně zvýšený výskyt defektů – může jít totiž o strategické rozhodnutí v rámci konkurenčního boje s cílem využít situace na trhu a získat co nejvíce zákazníků. Podobně tomu bývá u zábavního softwaru, kdy se nezřídka výrobce snaží dostat produkt na trh před Vánocemi. V takových případech ale výrobce kalkuluje s podstoupeným rizikem a velmi rychle poskytuje například bezplatné záplaty a aktualizace.

Z výše uvedeného jasně vyplývá, jak významný je pro výrobce softwaru správný přístup k zajišťování kvality při vývoji svých produktů, neboť kvalita a dobrá reputace jsou často silnou konkurenční výhodou.

Vliv a dopad softwaru na uživatele

Řada lidí, především pak vývojářů, mnohdy nerozumí nebo nesprávně chápe pojem *user experience* – UX (do češtiny bychom jej mohli pro zachování významu překládat jako prožitek z používaného softwaru). Tento pojem je často zaměňován s použitelností nebo uživatelskou přívětivostí, a přestože jsou zde určité společné rysy, nejde o pravou podstatu UX.

Co tedy UX vlastně znamená? Pochopitelně, že jediná správná odpověď neexistuje, ale v zásadě jde o celkový dopad našeho produktu na zákazníka jako člověka a to především jak

jej vnímá, jak na něj působí v emocionální rovině a ovlivňuje ho, zda se mu práce s ním líbí, těší ho a baví (nebo dokonce překvapí v kladném slova smyslu), což je mnohem více než jen splnění specifikovaných požadavků a uživatelská přívětivost. Produkt slouží určitému účelu, UX však zachází dále – snažíme se přizpůsobit tomu, kdo bude asi aplikaci typicky používat, kde a za jakých podmínek a co by mohlo být dále dobré a co by – v kontextu uvedených faktorů – mohlo naopak uživatele obtěžovat či frustrovat. Pro přesnost uveďme, že o UX mluvíme jen tehdy, existuje-li mezi uživatelem a produktem interakce – pokud vidíme statickou obrazovku programu, můžeme nanejvýš říct, zda se nám jeho vzhled líbí nebo ne.

Z výše uvedeného je patrně zřejmé, že UX je jen velmi okrajově věcí vývojářů systémů, neboť ti zřídka dokážou vnímat produkt současně z obou perspektiv. Ve skutečnosti se lidé mající na starost návrh UX rekrutují z řad marketingových specialistů, testerů, doménových znalců či dokonce sociologů.

Testování, především méně technicky zaměřenými lidmi z požadované cílové skupiny či s potřebnou doménovou znalostí, je jednou ze základních aktivit nutných ke zlepšování UX. Vzhledem k tomu, jak dnes technologie a interakce s ní prostupuje našimi životy prakticky ve všech oblastech, stalo se UX velmi významným elementem (nejen) z obchodního hlediska.

Stav testování v České republice oproti zahraničí

V letech 2010–2012 jsme (AH) prováděli výzkum zaměřený na úroveň testování a řízení kvality ve firmách vyvíjejících software na území České republiky, u nichž kvalita, efektivita a spolehlivost vývoje jsou důležitými aspekty konkurenceschopnosti.

Jako základ pro hodnocení výsledků průzkumu stavu testování v České republice byl použit obdobný průzkum v zahraničí. Počet a charakteristika účastníků tohoto průzkumu odpovídala počtu a charakteristice účastníků českého průzkumu, je však nutné zmínit dva rozdíly, které mohly v menší míře ovlivnit výsledky průzkumů:

1. Zatímco průzkum v České republice probíhal na konci roku 2010, průzkum v zahraničí proběhl o rok a čtvrt později, v lednu 2012. Tento rozdíl by však měl být zanedbatelný, jelikož se stav testování v České republice dle zkušenosti autorky za tuto dobu nijak nezměnil.
2. Zatímco v České republice byla oslovena velká část firem s žádostí o vyplnění dotazníku, zahraniční průzkum žádal o vyplnění dotazníku pouze na zahraničních fórech a diskuzních skupinách (náhodní Čeští respondenti nebyli do výsledků zahrnuti). Dá se tedy předpokládat, že větší část zahraničních respondentů má větší zájem o získávání znalostí v oblasti testování, než by tak mohlo být u průměrného českého vzorku.

Tyto rozdíly a jejich dopady byly vzaty při vyhodnocení a analýze výsledků do úvahy. Z výsledků průzkumů bylo zjištěno, že testování softwaru je v České republice *věnována menší pozornost než v zahraničí*, což se odráží zejména ve vzdělávání pracovníků a v používání moderních technik a nástrojů.

Pro zlepšení tohoto stavu byly navrženy následující kroky:

1. Rozšíření nabídky vysokoškolských a odborných kurzů zejména s důrazem na preventivní metody řízení kvality a na seznámení manažerů s ekonomickými aspekty investování do kvality vývoje softwaru.
2. Rozšíření praktických kurzů zacílených na použití teoretických znalostí o řízení kvality v praxi.
3. Rozšířit nabídku školení ve firmách pro zaměstnance aktivně se podílející na utváření kvality softwaru, případně motivovat zaměstnance k řízenému samostudiu s doporučeným seznamem kvalitních zdrojů.

Vzdělávání v oblasti testování a řízení kvality

Při posuzování kvality produktů firmy využívají v čím dál větší míře podrobné testovací scénáře a nástroje na automatizaci testů, čímž delegují provádění i značné části testů na nezkušené pracovníky nebo automaty. Na druhé straně se tím zvětšuje prostor pro specializaci odborníků v oboru testování. Znalosti, schopnosti a zkušenosti v oblastech byznys logiky produktu a testování softwaru tak nadále hrají významnou roli.

Merkel [Merkel a Kanij, 2010] ve své studii dokládá pozorování výrazných rozdílů ve výkonnosti testerů softwaru, kdy nejlepší z nich jsou přinejmenším o polovinu výkonnější, než je v daném týmu průměr. Z toho vyplývá, že individuální schopnosti a znalosti velkou měrou ovlivňují výsledky jednotlivců.

Pokud bychom se zajímali o to, co mají dobří testeři společného, pak následující vlastnosti byly společné u všech zkoumaných testerů s vysokou výkonností [Iivonen, 2009]:

- Zkušenost s produktem
- Zkušenost v dané byznys oblasti, doménová znalost
- Zkušenost s programováním
- Zkušenost se speciálními testovacími technikami
- Psaní dobrých hlášení chyb
- Udržení nadhledu/přehledu
- Nadšení z práce – skutečně je baví
- Pečlivost, svědomitost, trpělivost a výdrž

Poznámka: Vysoce výkonní testeři byli identifikováni analýzou obsahu databáze systémů pro hlášení chyb na základě počtu, závažnosti a podílu vyřešených chyb.

Ve výše uvedeném seznamu není zmíněna práce s nástroji, neboť chyby nalezené automatem není vždy možné přiřadit konkrétnímu objeviteli.

Obecně jsou využívány následující zdroje a způsoby získávání znalostí a zkušeností:

- Vysokoškolské studium
- Zaměstnání
- Samostudium
- Certifikace a rekvalifikace
- Školení a kurzy
- Konference
- Internet (blogy, odborné články, diskuzní fóra)
- Knihy
- Praxe

Pomineme-li získávání zkušeností z praxe, převládá jako hlavní zdroj znalostí v České republice Internet (46,4 %), následován kolegy (20,2 %) a knihami (17,9 %). Školení a kurzy, ať už odborné nebo univerzitní, mají nejmenší podíl na vzdělávání pracovníků softwarových firem v oblasti testování.

To je výrazný rozdíl oproti zahraničí, kde tito pracovníci softwarových firem čerpají znalosti v oblasti testování zejména ze školení (48,6 %) a dále z Internetu a knih. Zatímco samostudium je neřízená forma vzdělávání, jehož kvalita závisí na zvoleném způsobu získávání znalostí, absolvování speciálně zaměřených kurzů v rámci vysokoškolského studia by mělo poskytovat dostatečnou záruku kvality vzdělání.

V České republice však testování a řízení kvality zatím není plně integrovanou součástí výuky informatiky na všech vysokých školách, které tak i dnes produkují budoucí IT manažery, softwarové inženýry a analytiky, kterým chybí znalosti o utváření kvality a praktické dovednosti s nástroji pro její zajištění.

Základní axiomy testování softwaru

Axiom je tvrzení, jehož pravdivost je obecně přijímána, a není tak potřeba ji dokazovat. U testování softwaru, stejně jako u ostatních disciplín, lze formulovat řadu axiomů [Patton, 2002]. Čtyři nejvýznamnější a pravděpodobně i nejznámější z nich si představíme v bodech níže:

1. *Žádný netriviální program není možné otestovat úplně.* Nezřídka se lze setkat s názorem, že softwarový produkt lze zcela otestovat. Takové kompletní otestování dané aplikace by znamenalo, že budou odhaleny všechny defekty. To je však – s výjimkou triviálních aplikací – v praxi téměř nemožné. Testování by v takovém případě muselo zahrnovat všechny správné i nesprávné vstupní hodnoty, stavy programu a přechody mezi nimi či možné vlivy prostředí, které by bylo třeba nasimulovat. Je tak zřejmé, že ve většině případů je počet všech možných kombinací jednotlivých hodnot jednoduše příliš velký.

2. *Testování je věcí odhadu rizika.* Protože není možné úplné otestování softwaru, tester či analytik testování vybere k testování scénáře podle vlastního uvážení. Tím však samozřejmě riskuje, že nezachytí možný defekt ve scénářích, jež nevybral. Zvládnutí správně provedené analýzy rizik a způsobů jejich minimalizace tak patří k nezbytným dovednostem, které musí každý tester ovládat.
3. *Testování může prokázat jen přítomnost defektů, nikoli jejich absenci*. Testování umožňuje prokázat přítomnost defektů v softwaru, avšak nelze jím prokázat jeho bezchybnost, především pro nemožnost kompletního otestování (viz první axiom). A tak nehledě na to, kolik času věnujeme testování programu, nikdy jej nemůžeme označit za bezchybný. Výjimku tvoří triviální aplikace, se kterými se však jen stěží setkáme v rámci dnes vyvíjeného softwaru.
4. *Čím více defektů je nalezeno, tím více jich v produktu je*. Tento axiom vychází z toho, že defekty se vyskytují ve skupinách, neboť lidé často dělají stejné chyby a z podstaty mohou někdy chybovat více než jindy (každý může mít „špatný den"). Proto pokud je při testování nalezeno více defektů v určité části systému, je pravděpodobné, že se zde nachází ještě další.

Proč programátor nemá testovat svůj kód

Zkušenosti ukazují, že programátor by neměl testovat svůj vlastní kód. Tato zásada je většinou obecně dodržována s výjimkou testování jednotek. Jaké jsou ale pro tento přístup důvody?

Prvním problémem je situace, kdy programátor neporozumí specifikacím zcela správně – požadovanou funkcionalitu tak naimplementuje špatně, avšak při následném testování to nemusí vůbec odhalit, neboť vychází z předpokladů získaných nesprávným pochopením samotného zadání. Toto riziko je pochopitelně významně sníženo, jestliže je testování prováděno někým jiným.

Další potíž je v tom, že programátor se při testování přirozeně zaměřuje na prokázání, že jeho kód funguje správně. Protože kód zná, testuje logicky to, co implementoval. Pokud však zapomněl ošetřit určité případy či situace, nejspíše je zapomene i otestovat.

S tím také souvisí i výskyt mylných předpokladů, že určité věci se nemohou stát. Při vývoji i testování jsem se s tím setkal mnohokrát. „Tomu nevěřím, to přece není možné!" – jsem říkával sám nebo slýchal při nalezení defektu tam, kde se to nezdálo možné. Na rozdíl od testera bez znalosti fungování daného kódu může programátor určité případy opomenout se zdůvodněním, proč to z toho či onoho důvodu nemůže nastat.

Poznámka: Uveďme příklad z praxe týkající se chybných předpokladů programátora. Uživatelské rozhraní umožňovalo vyhledávat záznamy podle mnoha kritérií – vyhledávacích filtrů, jež se jednoduše přetahovaly do vymezeného prostoru a případně vymazávaly přetažením zpět či použitím klávesnice. Při testování však nastávala chyba, neboť pokud byl filtr zadán

a následně smazán, stále byl v platnosti. Programátor však nemohl chybu reprodukovat a byl si jistý, že vše funguje správně. Nakonec se však ukázalo, že tester k vymazání položky používal stisk klávesy Backspace, o jejíž podpoře použitou komponentou programátor vůbec nevěděl – vždy totiž automaticky používal pouze klávesu Delete!

Ostatní faktory negativně ovlivňující objektivitu programátora při testování vlastního kódu jsou spíše psychologického rázu. Ne každý je totiž schopný objektivně a vysoce kriticky přistupovat k vlastní práci (což platí obecně) a případně přiznávat pochybení před ostatními, obzvláště může-li nést určité následky. Navíc programátor ví, že nalezené chyby mu přidělají práci, protože je bude muset odstraňovat a poté danou část znovu otestovat.

Uvádí se, že tyto vlivy – většinou nevědomě – ovlivňují motivaci a objektivitu programátora testujícího vlastní kód. Z těchto důvodů je vhodnější a efektivnější, aby testování bylo prováděno nezávisle, s využitím jiného přístupu, a tak bez působení zmíněných faktorů.

Poznámka: Především dřívější praxí u velmi malých projektů bylo, že programátoři si testovali kód navzájem mezi sebou. Přestože nejde o ideální přístup, uvedené negativní vlivy jsou tím alespoň do určité míry redukovány.

Mýty a pravda o testování

Přestože v poslední době je na kvalitu kladen stále větší důraz, nezřídka se lze setkat se zkreslenými představami a někdy až předsudky, jde-li o chápání testování, jeho role v procesu vývoje produktu a lidí, kteří se jím zabývají. Podívejme se tedy na tyto „mýty" – jak je často zkušení (nejen) testeři nazývají – trochu blíže.

Poměrně rozšířený je názor, že testování celkově je jednoduchá, nudná a jednotvárná práce. Zde nelze začít jinak než citací z knihy The Art of Software Testing: „*Testování je neobyčejně tvůrčí a intelektuálně náročná úloha*". [G.Myers a kol., 2004] Připomeňme si, že testování je silně závislé na kontextu a vyžaduje navíc potřebnou analýzu. Pokud si pod pojmem testování představujeme prosté „proklikávání" tlačítek uživatelského rozhraní malých, nekomplikovaných aplikací, pak se toto tvrzení jistě může jevit jako přehnané. Ačkoli i v takových případech je třeba k odhalení logických chyb projevit dostatek invence při vytváření jednotlivých scénářů, skutečně může jít o poměrně monotónní práci (vynecháme-li exploratorní testování). Takové projekty však jen málokdy využívají služeb profesionálních testerů.

Pokud uvažujeme komplexnější aplikace, pak relativně jednodušší testování frontendu často vyvažuje mnohem náročnější testování backendu (včetně middleware), který může například sestávat z několika systému zajišťujících získávání dat, jejich transformaci, zpracování, aplikaci business logiky, ukládání a tak dále.

Poznámka: Obecně jako frontend označujeme součást softwaru, se kterou interaguje uživatel (uživatelské rozhraní v různé formě) a jež se také někdy označuje jako prezentační vrstva. Pod ní leží aplikační či servisní vrstva, tzv. middleware, a nejníže pak vrstva datová, nazývaná backend.

V takovém případě vyžaduje testování nejen znalost použitých technologií a velmi dobré pochopení požadavků k provedení analýzy, ale také schopnost pro odvozené scénáře vytvářet dostatečně variabilní a validní vstupní data, typicky simulující data skutečná.

Obecně nejsou neobvyklé případy, kdy je testování určité funkcionality celkově náročnější než její samotný vývoj. A nezapomeňme ani na automatizaci či zátěžové testování, kde je pro většinu složitějších scénářů nutné nejen upravovat zdrojový kód připravovaných testů, ale často také přímo programovat.

Mnoho lidí zastává názor, že testovat může každý, neboť testování nevyžaduje žádné zvláštní znalosti. Toto tvrzení ve své podstatě vychází z problematického předpokladu, že testování je snadné, což jsme rozebrali výše. Druhým důvodem je fakt, že zatímco tester může své znalosti a zkušenosti nabírat postupně během práce na stále složitějších projektech, programátor jimi musí – v nezbytné míře – disponovat již při práci na svém prvním projektu.

Poznámka: Málokdo si také uvědomuje, že zkušení testeři nepracují pouze s funkční aplikací, ale často také provádí statickou analýzu v rámci verifikace.

V určitých případech týkajících se čistě manuálního testování se může z praktického hlediska zdát dostačující, pokud osoba na pozici testera pouze nastuduje požadavky uživatelů a seznámí se s aplikací, avšak bez znalosti alespoň základních technik testování lze oprávněně předpokládat, že efektivita jeho práce bude nižší, a tedy riziko pro výrobce testovaného produktu a následně jeho zákazníky vyšší. Ve výsledku může takový přístup, většinou motivovaný snahou ušetřit na testování, ohrožovat kvalitu produktu. Často také dochází ke zbytečnému vytěžování ostatních členů týmu, kteří musí takovému testerovi pomáhat s techničtějšími úlohami (aktualizace kódu, databázové dotazy, nastavování prostředí a tak dále).

Při stále se zvyšujících očekáváních zákazníků týkajících se kvality produktů je tak v zájmu výrobce, aby zaměstnával dostatečně kvalifikované testery. V případě vývoje složitějších či kritických systémů například z oblasti finančního sektoru je pak samozřejmé, že nároky na znalosti, schopnosti i zkušenosti testerů bývají velmi vysoké. Porovnáme-li současnou situaci se stavem kolem roku 2000, je zde zřejmý výrazný posun: od testerů se očekává mnohem více než jen minimální znalost teorie a případně některých technik, což rozebereme níže v části Testeři a programování.

Dobrý tester je schopný dívat se na systém jako programátor, koncový uživatel a analytik zároveň, což vyžaduje kombinaci nejen technických, ale i řady jiných (typicky sociálních) dovedností, které často zúročí na vyšších pozicích – například při řízení týmu či vyjednávání

se zákazníkem. Asi proto není tolik překvapivé, že skutečně dobrých testerů je v současnosti velký nedostatek, přičemž samostatnou kategorií jsou specialisté na automatizaci a zátěžové či bezpečnostní testy.

Poznámka: Není neobvyklé, že testery se stávají programátoři a stejně to platí i naopak. Pokud zmiňujeme změnu profesní dráhy, tak schopní testeři s dostatečnými znalostmi nezřídka přechází na pozice například business analytiků.

Jeden z dalších a poměrně častých „mýtů" se týká role automatizace testování. Její rozvoj za poslední dekádu a fakt, že se stává na mnoha projektech neodmyslitelnou součástí testovacího procesu, vedou mnoho lidí (pochopitelně většinou těch, kteří se testováním sami nezabývají), k názoru, že automatizace by měla a může zcela nahradit manuální testování. Podle těchto úvah se tak testování stane rychlejším, levnějším, přesnějším, bez problému pracujícím přesčas…

Ve skutečnosti však lze bez zbytečné polemiky na toto téma říct, že v současnosti a ani blízké budoucnosti to možné nebude, pokud vynecháme regresní testování či specifické úlohy, kde je automatizace široce využívána už dnes. Ovšem i zde je to opět člověk používající inteligenci a kreativitu, kdo navrhuje a také udržuje scénáře testované automatizovanými skripty, počítač je pouze v roli bezduchého, ovšem silného vykonavatele.

Automatizace v současné situaci není nástupcem manuálního testování, je však jeho suplementární aktivitou, která v závislosti na povaze projektu může a nemusí být efektivnější. Více o výhodách a nevýhodách automatizace si povíme v samostatné kapitole *Automatizace testování*.

Na závěr se tak nabízí otázka: Kde je tedy pravda? Výše jsme se pokusili demonstrovat, že tyto „mýty" rozhodně nejsou pro testování obecně platné, jak si mnozí myslí. Na druhou stranu by ale bylo nespravedlivé tvrdit, že nemají žádný pravdivý základ, což není vzhledem k povaze a různorodým podobám testování nijak překvapivé.

Testeři a programování

Doby, kdy se souhrnně od pracovníků v oblasti testování neočekávala žádná znalost programovacích či skriptovacích jazyků, jsou už dávno za námi a dnes takové požadavky nejsou nikterak výjimečné, spíše naopak.

Neplatí to samozřejmě pro testery obecně a mnoho z nich při své práci tyto znalosti skutečně nepotřebuje a nevyužije.

Na druhou stranu, od profesionálů pracujících na významných a prestižních projektech je většinou vyžadována určitá úroveň znalostí, kam z řady důvodů patří i programování, a to bez ohledu na to, zda jej při své denní práci aktivně využívají. Nejde tak proto ani tolik o kon-

kurenční výhodu jako spíše o nutnost specializace. To lze přičíst především stále komplexnějším aplikacím a měnícímu se přístupu k testování, kdy se alespoň částečná automatizace či provádění základních zátěžových testů stává nezbytností, nehledě na snahu o usnadnění a zefektivnění práce pomocí různých (vlastních) nástrojů. Dalším důvodem je fakt, že znalost programovacího jazyka či dané technologie je přínosná i při samotném testování, neboť umožňuje předvídat možná pochybení programátora a slabá místa systému.

V následujících odstavcích rozebereme, co lze dnes obecně považovat za určité „programátorské minimum", které by měli zvládat univerzální testeři pracující typicky na vyšších, tedy nikoli juniorských pozicích a různorodých projektech, jež se mnohdy výrazně liší technologií, doménou i různou mírou technické náročnosti.

Na rozdíl od vývojářů nemusí samozřejmě tester běžně znát programovací (resp. skriptovací) jazyky zcela podrobně, avšak měl by ovládat alespoň jeden z nich na pokročilejší úrovni a základy několika dalších. Z praktické zkušenosti je pojmem „základy" myšleno zvládnutí základů syntaxe a sémantiky daného jazyka a jeho případných specifických vlastností na úrovni porozumění kódu. Z povahy dnes používaných nástrojů pro automatizované testování apod. lze vyvodit, že kupříkladu znalost principů objektově orientovaného programování již mezi běžně očekávané dovednosti testera nepatří, ovšem rozhodně je výhodou. Typicky tak jde o jazyky, jako jsou například Python, Perl či Ruby.

Výše uvedené se nevztahuje na jazyk SQL, jehož pokročilejší až pokročilá znalost je dnes, vzhledem k rozšířenosti relačních databází a jejich častému využití při testování, považována již prakticky za samozřejmost. K tomu je nezřídka počítána i základní znalost jeho procedurálních rozšíření, jako je PL/SQL či Transact-SQL (T-SQL).

Neméně důležitá je také znalost podpůrných nástrojů používaných při vývoji softwarových produktů, jako jsou systémy pro správu verzí (nejen) zdrojových kódů či systémy pro průběžnou integraci, se kterými by měl tester být schopen samostatně pracovat v rozsahu, jejž vyžadují jeho úkoly.

Role v testovacím týmu

Dnes se jen velmi vzácně setkáme s tím, že by testování a související aktivity byly svěřeny pouze jednomu pracovníkovi. Obvykle je podle potřeb projektu a daných omezení sestaven různě velký tým, jehož členové mají specifické role. Struktura týmu, názvy jednotlivých rolí a často i konkrétní úkoly jsou pochopitelně poměrně proměnlivé, a tak se podívejme na testovací tým a jeho členy obecně.

Jako první si ale řekněme něco o rolích s názvy jako QA inženýr či QA Analytik (*QA Engineer / QA Analyst*). Tato označení jsou totiž velmi často nepřesně používána pro pracovníky, jejichž skutečnou pracovní náplní je testování. Jak jsme ale uvedli již ve druhé kapitole, testování není QA. QA Engineer se z pohledu kvality zabývá sledováním, hodnocením a sna-

hou zlepšovat procesy, nikoli jejich výstupní produkty. U procesu testování může stanovovat a kontrolovat používané postupy, nastavovat procedury reportování defektů, provádět analýzu získaných dat s cílem odhalit příčiny defektů a podobně. Je však třeba dodat, že někdy se skutečně pracovní náplň testera a pracovníka QA může v určitých oblastech překrývat. Přesto ale samotné provádění testů není běžnou aktivitou QA.

Obecně jsou ve struktuře testovacích týmů středních a větších projektů zastoupeny následující role (u testerů a analytiků je testování navíc často rozlišované dále na juniorské a seniorské):

- *Tester* je zodpovědný především za provádění testů podle připravených scénářů, záznam získaných výsledků (tedy rozdílu mezi skutečným a očekávaným stavem), případně za vytváření hlášení o nalezených defektech. Po opravě defektu je jeho úkolem přetestování a správa stavu nahlášeného defektu včetně výsledků původního scénáře. Běžná je také zodpovědnost za generování zpráv o výsledcích provedených testů a správa testovacích prostředí – funkčnost, použitelnost, konzistence dat, přístupová práva a podobně.
- Analytik testování (*Test analyst* či *Test engineer*) má za úkol vytváření testovacích scénářů a jednotlivých testovacích případů. Jeho úloha je často obtížná. První krok spočívá v důkladné analýze požadavků a v následném vytyčení všech případů (negativních i pozitivních), které je nutné testováním ověřit. Následuje samotná tvorba jednotlivých testovacích případů se všemi náležitostmi. Dále je třeba zvážit riziko a stanovit prioritu pro jejich provádění – zdroje jsou omezené a odhalení problémů v klíčové funkcionalitě má pochopitelně přednost. Mezi další obvyklé činnosti analytika testování patří například vytváření sad regresních testů a jejich následná údržba, hlášení o pokrytí testů vzhledem k požadavkům či podpora zákazníka.

Poznámka: Bývá obvyklé, že role testera a analytika testování jsou sloučeny a zastává je jeden pracovník. Zkušený analytik také většinou provádí statickou analýzu. To však nemusí platit například pro projekty vyžadující velký objem regresního testování, a tedy i testerů, kteří se soustředí pouze na provádění testů.

- Tester – automatizované testování (*Test automator*). Projekty využívající automatizaci testování obvykle disponují specialisty zaměřujícími se výhradně na tuto činnost, především pokud uvažujeme robustnější komerční nástroje určené pro testování komplexních aplikací. Tito členové testovacího týmu spolupracují typicky s analytiky testování – dle technologie, strategie a požadavků volí vyhovující nástroj a automatizují vhodné testovací případy či celé scénáře. Kromě vývoje a údržby samotných testů vytváří potřebnou dokumentaci, zaznamenávají výsledky jednotlivých běhů (včetně hlášení nalezených defektů) či spravují prostředí, na kterém jsou automatizované testy prováděny. V podstatě totéž lze říci o specialistech na zátěžové testy.
- Vedoucí testování (*Test lead*). U velkých projektů je běžnou praxí, že jednotlivé části (pro jednoduchost si představme například backend a frontend) vyvíjeného systému jsou testovány samostatným týmem, za který odpovídá vedoucí testování. U menších

projektů pak jde o odpovědnost za testovací tým celkově. První úlohou vedoucího testování bývá vytvoření plánu testování (*test plan*) v souladu se strategií testování, kontrola jeho dodržování a případné korektivní akce. Dále nastavuje a spravuje systém pro hlášení defektů, zadává úkoly jednotlivým členům týmu a dohlíží na jejich plnění, z dílčích zpráv pravidelně generuje hlášení o celkovém stavu testování dané části systému a tak dále. Může se účastnit také samotného navrhování i provádění testů. Předpokladem pro tuto roli jsou kromě technických také dostatečné znalosti z oblasti teorie i praxe testování softwaru, zkušenosti s řízením lidí a organizační dovednosti.

- Manažer testování (*Test manager*). Tato role je na vrcholu pomyslné pyramidy testovacího týmu. Manažer testování je na projektu jako celku odpovědný za veškeré aktivity související s testováním (či řízením kvality vůbec) a odpovídá vedení projektu či vyššímu managementu. V určitých situacích spolupracuje s vedoucím testování či schvaluje některé jeho kroky – například při vytvoření plánu testování. Jeho obvyklá činnost zahrnuje volbu přístupu k testování a nastavení jeho cílů, stanovení kritérií kvality produktu a hodnocení jejich naplnění, správu klíčových dokumentů, vyjednávání s managementem, zákazníkem i vedoucím vývoje, koordinaci akceptačního testování, spolupráci s QA oddělením a tak dále. Významná je obvykle jeho pravomoc v oblasti personálních zdrojů – rozhoduje o sestavení testovacího týmu a jeho změnách v průběhu projektu, hodnotí jednotlivé členy, zajišťuje pro ně potřebné nástroje. U menších projektů bývá často tato role sloučena s rolí vedoucího testování.

Poznámka: Jestliže výrobce pracuje na několika projektech ze stejné oblasti (pro jednoho zákazníka), není neobvyklé, aby se manažer testování podílel současně na několika z nich.