Efektivní testování softwaru

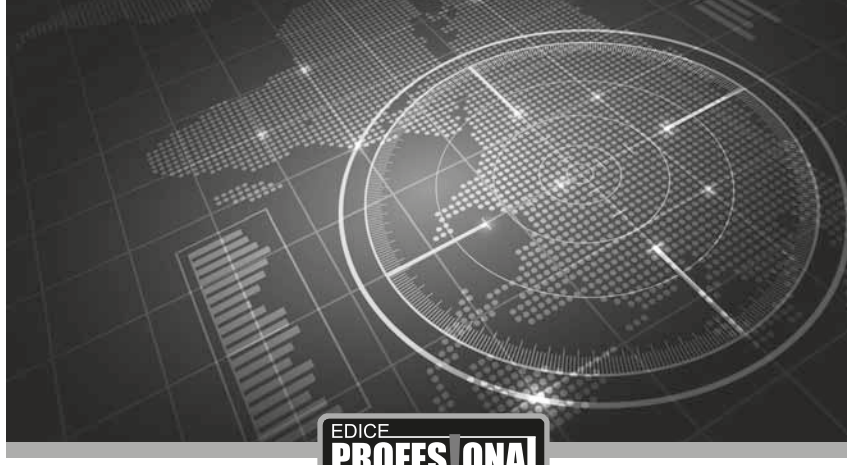
Klíčové otázky pro efektivitu testovacího procesu

Miroslav Bureš, Miroslav Renda
Michal Doležel a kolektiv

EDICE PROFESIONAL

GRADA®

- Organizace a řízení testování softwarových systémů
- Realistický plán testování a prevence možných rizik
- Příprava testovacího prostředí a testovacích dat
- Optimalizace regresních testů
- Statické testování a testování požadavků
- Odhadování testovacích aktivit
- Efektivní využití automatizovaných testů



GRADA®

Efektivní testování softwaru

Klíčové otázky pro efektivitu testovacího procesu

Miroslav Bureš, Miroslav Renda
Michal Doležel, Peter Svoboda
Zdeněk Grössl, Martin Komárek
Ondřej Macek, Radoslav Mlynář

Miroslav Bureš, Miroslav Renda, Michal Doležel, Peter Svoboda, Zdeněk Grössl, Martin Komárek, Ondřej Macek, Radoslav Mlynář

Efektivní testování softwaru

Klíčové otázky pro efektivitu testovacího procesu

Vydala Grada Publishing, a.s.
U Průhonu 22, Praha 7
obchod@grada.cz, www.grada.cz
tel.: +420 234 264 401, fax: +420 234 264 400
jako svou 6339. publikaci

Recenzovali:
doc. Ing. Valentino Vranić, Ph.D.
doc. RNDr. Jiří Barnat, Ph.D.
doc. Ing. Branislav Lacko, CSc.

Odpovědný redaktor Petr Somogyi
Sazba Jan Šístek
Fotografie na obálce fotobanka Allphoto
Počet stran 232
První vydání, Praha 2016
Vytiskly Tiskárny Havlíčkův Brod, a. s.

ISBN 978-80-271-9389-9 (pdf)
ISBN 978-80-247-5594-6 (print)

Obsah

Úvod

Držíte v rukou knihu, která se zabývá vybranými tématy z oblasti testování softwaru. Věříme, že vám informace a zkušenosti obsažené na následujících stránkách pomohou zefektivnit práci, nastavit nový úhel pohledu na věc nebo vyvolat následnou diskuzi.

Testování představuje průměrně 20 % až 40 % pracnosti v projektech vývoje softwaru. Pro vývoj řídicího softwaru s vysokými požadavky na spolehlivost může pracnost testování často přesáhnout pracnost vývoje. Pokud má být testování efektivní, musíme k němu přistupovat jako k samostatné odborné disciplíně – stejné, jako je analýza, návrh, architektura, vývoj nebo projektové řízení. Klíčem k úspěchu je vyvážená kombinace metodiky a praktických zkušeností.

Proč jsme se rozhodli o testování softwaru napsat další knihu? Na toto téma je k dispozici řada odborných publikací v anglickém jazyce, ale v češtině jich je pomálu. A právě místní kontext je podle našeho názoru podstatný – praktické zkušenosti a rady, které najdete na dalších stránkách, jsou pochopitelně ovlivněné projekty, z nichž pocházejí. Autoři jednotlivých kapitol sice pracovali na řadě zahraničních nebo mezinárodních projektů, ale velká část našich zkušeností pochází z českého prostředí.

Kniha poskytuje průřez oblastí, které jsou podle našeho názoru na současných testovacích projektech problematické, nebo představují rezervu z hlediska efektivity testování. Kniha ani zdaleka nepokrývá kompletní životní cyklus testování. Vědomě zanedbáváme zejména témata týkající se analýzy testování. Ta byla rámcově pokryta v knize [1] a jejich detailní popis by vydal na samostatnou publikaci.

Pro jaké typy projektů a domény se kniha hodí?

Testování softwaru je velmi různorodá disciplína. Jedno ze sedmi základních pravidel testování prezentovaných ve standardu ISTQB [2] říká, že testování je závislé na kontextu. Mobilní aplikaci elektronického bankovnictví vyvíjenou agilním způsobem budeme testovat podstatně jinak než řídicí software dopravního letadla.

Tato kniha je do jisté míry ovlivněna doménovými znalostmi a zkušenostmi jejích autorů. Většina projektů, na kterých jsme pracovali, byla řízena stylem vývoje typu vodopád (*waterfall*). Agilnímu testování se proto v knize systematicky nevěnujeme. Téma je pokryté v řadě jiných publikací, např. [3]. Nejvíce zkušeností máme z oblasti vývoje webových nebo desktopových aplikací. Projekty, které ovlivnily naše zkušenosti, byly realizované zejména v oblastech bankovnictví, pojišťovnictví a telekomunikačních služeb. Méně zkušeností máme v oblasti testování vestavěných systémů (*embedded systems*) a softwaru, kde jsou kladené vysoké požadavky na spolehlivost. Pro tyto oblasti nemusí všechny popsané metody stoprocentně odpovídat.

Jak je kniha strukturovaná

Kniha je uspořádána do 13 kapitol, každá z nich se snaží pokrýt jednu tematickou oblast bez větších závislostí na jiné kapitole. Můžete tedy pracovat s jednotlivými kapitolami „na přeskáčku" tak, jak se vyskytne potřeba pro rozšíření znalostí konkrétní problematiky. Jedinou výjimkou je kapitola *2*, *Role testování ve firmě a v projektu*, která vysvětluje základní pojmy o testování, z nichž vychází výklad v ostatních kapitolách. Tuto kapitolu doporučujeme přečíst vždy.

Každá kapitola obsahuje závěrečné shrnutí, které rekapituluje nejdůležitější informace k danému tématu. Pokud byste nenašli čas na přečtení všech kapitol, doporučujeme prolistovat knihou a přečíst alespoň tato shrnutí.

Dílčí kapitoly jsou psané různými autory, kteří jsou experty na probíranou oblast. Nevýhodou tohoto „sborníkového přístupu" může být mírně odlišný jazykový styl jednotlivých kapitol. Věříme, že kvalita informací tuto nevýhodu výrazně převáží.

Komu je kniha určená?

Knihu jsme psali pro začínající manažery testování, nebo pro zkušené testery či analytiky testování, kteří by se časem chtěli do role manažera testování profilovat. Kromě toho samozřejmě doufáme, že nové poznatky nebo nové úhly pohledu na jednotlivé problémy najdou v knize i zkušenější manažeři testování. Pro získání základního přehledu knihu doporučujeme i projektovým manažerům.

Použitá terminologie

Při psaní knihy jsme se snažili používat odborné termíny ve překladu použitém v glosáři ISTQB [4]. V několika málo případech jsme se od tohoto zdroje uchýlili, protože jsme buď považovali za vhodnější termín ustálený v testerské praxi, nebo glosář daný termín neobsahuje.

O autorech

Miroslav Bureš

Testováním softwaru se zabývá již přes deset let. V současné době působí na katedře počítačů ČVUT FEL, kde se zabývá výzkumem a inovacemi v oblasti testovacích metodik, efektivní automatizace testování a automatizace návrhu testovacích scénářů. Zde také vyučuje předměty týkající se testování softwaru a vede řadu experimentálních projektů s cílem zvýšit efektivitu současných testovacích metod. V předchozím období vedl testovací oddělení české a slovenské pobočky Capgemini, spolupracoval s tvůrci testovací metodiky TMAP a působil v řadě projektů z oblasti financí jako manažer testování a projektový manažer.

Miroslav Renda

Od roku 1995 prošel většinou projektových rolí v oblasti dodávky softwaru. Nejdéle působil v roli projektového manažera. Je držitelem mezinárodních certifikátů ISTQB, Prince 2, ITIL a dalších. V poslední době vedl testovací oddělení v konzultační společnosti Capgemini, v současnosti řídí testování nebo dodávky IT projektů klientům ve finanční sféře. Několikrát přednášel na mezinárodní konferenci Czech-Test. Ve volném čase se věnuje rodině, sportu a práci viceprezidenta odborné neziskové organizace CaSTB.

Michal Doležel

Má za sebou více než desetiletou technicko-manažerskou praxi v oblasti testování softwaru, především z odvětví telekomunikací a finančnictví. Dnes pracuje v pražském IT hubu farmaceutické společnosti Merck Sharp & Dohme, kde vede Testing Center of Excellence pro region EMEA. Rovněž spolupracuje s katedrou informačních technologií VŠE FIS, kde vyučuje v magisterském studiu a zabývá se výzkumem. Je autorem několika příspěvků publikovaných v odborných časopisech a na zahraničních konferencích. Mimo testování jsou předmětem jeho zájmu organizační koncepty pro vývoj informačních systémů a mezioborová problematika sociálních a kulturních faktorů v IT.

Zdeněk Grössl

Téměř patnáct let testuje software, prošel pozicemi od testera až po manažera testování, převážně v oblasti financí a utilit. Jeho hlavní doménou je oblast analýzy testování, což podporuje i fakt, že je jedním z nemnoha českých držitelů certifikátu ISTQB CTAL Test Analyst. V současné době vede testovací oddělení ve společnosti Ness Czech a současně přednáší na VŠE analýzu testování. Pravidelně také vede tutoriály o analýze testování na mezinárodní konferenci CzechTest a účastní se zájmových setkání proTest.

Martin Komárek

Specialista v oblasti business analýzy a specifikace požadavků na software. V současnosti pracuje ve firmě Cactoo Software, kde působí především jako lektor a konzultant. Kariéru začínal jako analytik bezpečnostně-kritických aplikací v AŽD. Od roku 2007 přednáší na ČVUT FEL tématiku softwarového inženýrství.

Ondřej Macek

Jako konzultant a školitel se podílí na řadě projektů z oblasti business analýzy a procesního řízení IT. Působil na ČVUT FEL, kde kromě realizace softwarových projektů vyučoval i předměty spojené s objektovým návrhem kódu a s řízením vývoje softwaru.

Radoslav Mlynář

V oblasti informačních technologií pracuje od roku 2001. Na začátku kariéry se zabýval implementací a administrací nástrojů pro řízení testů. Tři a půl roku strávil na projektech, kde měl na starosti implementaci technických požadavků od jejich analýzy, přes testování až po nasazení. Pět let vedl tým řízení kvality softwaru v České spořitelně. V současné době se v České spořitelně podílí na budování nové organizační jednotky – profesionálního testovacího centra v oblasti řízení testovacích prostředí a v oblasti řízení testů souhrnných celobankovních releasů.

Peter Svoboda

Testováním softwaru se zabývá od roku 2006, jeho specialitou je ekonomika testů a business pohled na testování. V minulosti působil v testovacím oddělení společnosti AVG a jako manažer testování a test konzultant ve společnosti Capgemini. V současné době buduje slovenskou větev společnosti tesena a věnuje se rozšiřování povědomí o nových trendech v testování na slovenském a českém trhu. Zároveň působí v roli manažera testování na IT projektech, především ve finančních institucích.

Poděkování

Na tvorbě knihy se podílela nejen osmička autorů. Na konečnou kvalitu díla měla velký vliv řada expertů z oblasti testování, kteří revidovali dílčí kapitoly. Za podnětné připomínky a náměty bychom chtěli poděkovat (v abecedním pořadí): Ladislavu Doubkovi, Karlovi Frajtákovi, Karolu Frühaufovi, Antonínu Gabrielovi, Anně Havlíčkové, Davidu Janotovi, Davidu Kristlovi, Štěpánu Květenskému, Tomáši Lisému, Martinovi Mičkalovi, Lubomíru Michalčákovi, Petru Neugebauerovi, Štěpánu Pelcovi, Slavoji Pískovi, Gáboru Puhallovi, Blance Rabhi, Zdeňkovi Samuelovi, Jaroslavu Strharskému, Pavlu Strnadovi, Milanovi Šterbovi, Romanu Štroblovi, Ivo Zelenkovi a Janě Zientkové. V neposlední řadě pak děkujeme recenzentům knihy doc. Ing. Valentinu Vranićovi, Ph.D., doc. RNDr. Jiřímu Barnatovi, Ph.D. a doc. Ing. Branislavu Lackovi, CSc.

1 Businessová hodnota testování

Peter Svoboda

Téměř každý IT projekt doprovází diskuze mezi sponzorem, businessovým vlastníkem zaváděného řešení, dodavatelem a manažerem testování o tom, jaká hloubka testování a potažmo jako množství vynaložených prostředků je žádoucí, potřebné a možné.

Cílem této kapitoly je poskytnout čtenáři argumentaci k obhájení potřeb testování vypočtených pomocí metod v kapitole 5, *Odhadování pracnosti testovacích aktivit*. Vysvětlíme si termín cena za kvalitu, podíváme se na její výpočet a postupně projdeme jednotlivými fázemi přípravy a použití argumentace nákladů na testování. Celý výklad bude proložen modelovými situacemi, které mohou manažerovi testování pomoci najít inspiraci při prosazení potřeb testování v organizaci.

1.1 Kam mohou vyšplhat náklady za nedostatečnou kvalitu

V této kapitole se nebudeme podrobně věnovat odhadování pracnosti testů, protože detaily k tomuto tématu najdete v kapitole 5, *Odhadování pracnosti testovacích aktivit*. Předpokládejme tedy, že pracnost stanovit umíme a dokážeme si tuto pracnost obhájit i před experty z oblasti testování. V praxi často narážíme na problém při obhajování ceny za testování u sponzora projektu a potažmo u dalších organizačních složek zapojených do projektu (business). Otázka, se kterou se každý manažer testování v takové chvíli potýká nejčastěji, zní: „**Proč máme tolik platit za testování? Cena za testy je příliš vysoká!**“ Vezměme si pro další výklad na pomoc událost, která se před nějakým časem skutečně stala.

V květnu 2015 vydal americký Úřad pro letectví (FAA) bezpečnostní bulletin pro všechny modifikace Boeingu 787 Dreamliner.[1] Bulletin se týkal softwarového problému v řídicí jednotce generátorů palubního napětí, ve kterém dojde po 248 dnech nepřetržitého běhu řídicí jednotky k přetečení rozsahu datového typu a následnému pádu softwaru do nouzového režimu. V praxi by to pak znamenalo okamžité vypnutí obou hlavních palubních generátorů a ztrátu napětí, což by znamenalo ohrožení bezpečnosti letu. Největší nebezpečí hrozí ve fázích vzletu a přistání, kdy by automaticky spuštěná náporová turbína díky nízké rychlosti letounu nezačala vyrábět dostatečné množství energie do šesti sekund od vypnutí generátorů, po kterou budou energii dodávat palubní baterie.

[1] FAA: Bulletin 2015-NM-058-AD, 2015 Dostupné také zde: http://rgl.faa.gov/Regulatory_and_Guidance_Library/rgAD.nsf/0/584c7ee3b270fa3086257e38004d0f3e/$FILE/2015-09-07.pdf.

Bezesporu se jedná o závažný defekt. K dubnu 2015 je v provozu 269 letadel typu Boeing 787 všech variant.

Podívejme se na detaily vnitřní implementace této funkcionality:

- V řídicím softwaru generátorů je jednoduché počítadlo, které narůstá každou setinu sekundy o 1.
- Počítadlo se vynuluje při vypnutí řídicí jednotky.
- Pro počítadlo byl použit datový typ o šířce 31 bitů. Rozsah počítadla je tedy 2^{31} milisekund.

To znamená, že po cca 248 dnech dojde k přetečení rozsahu počítadla a chybě v softwaru. V komerčním letectví je běžné, že se systémy letadla pří zastavení úplně nevypínají (hlavně ty, které mají na starost napájení, což je i tento případ). K úplné odstávce dochází zpravidla při servisních zásazích. Čas běhu 248 dnů je sice nepravděpodobný, nikoliv však vyloučený. Z tohoto důvodu FAA vydala nařízení k okamžitému restartu všech jednotek ve všech variantách modelu Boeing 787, aby se čas na odstranění defektu v softwaru prodloužil o (minimálně) 248 dnů.

Co to znamená z pohledu ekonomiky (ne)testování?

Nařízení FAA je pro aerolinky (minimálně americké) závazné, tudíž jej musí provést všechny, jež provozují daný typ stroje. Jeden restart byl vyčíslen na jednu pracovní hodinu technika hodnocenou 85 USD, takže pouhá **aplikace tohoto „workaroundu" stála americké provozovatele aerolinek 22 865 USD**.

Nepřímé finanční dopady, zejména nutnost stáhnout letadlo kvůli zásahu z provozu a ztráta reputace díky popularitě této zprávy v médiích, **mohou být o řád až dva vyšší**.

Vyřešení defektu bude nejspíše spočívat ve hloubkové analýze kódu a nahrazení všech nenulovaných proměnných s datovým typem integer 32 bit proměnnou typu integer 64 bit. Následovat budou rozsáhlé regresní testy, zátěžové testy a simulované testy dlouhodobého provozu (pokus o znovunalezení defektu). Jelikož systémy Dreamlineru obsahují přibližně 110 milionů řádků kódu, **tyto náklady se pravděpodobně vyšplhají na několik milionů dolarů**.

Zavedení testování dlouhodobé odolnosti (*soak testing*) ve vhodný okamžik přitom při podobně rozsáhlých projektech představuje náklady ve výši několika desítek tisíc USD. Jinými slovy – **investicí desítek tisíc USD do vhodného typu testu bylo možné předejít ztrátě několika milionů USD vyvolané defektem**.

Tabulka 1.1: Typické přímé náklady na opravu defektu v závislosti na fázi, ve které byla nalezena (zdroj: Capgemini)

	Analýza	Design	Vývoj	Vývojářské testy	Testování	Produkční provoz
Relativní počet zanesených defektů v dané fázi	10 %	40 %	50 %			
Relativní počet detekovaných defektů v dané fázi	3 %	5 %	7 %	25 %	50 %	10 %
Normalizované náklady na opravu defektů (EUR)	250	250	250	1 000	3 000	12 500

I v případě, že aktuálně nepracujete na projektu vývoje softwaru pro letectví, základní principy jsou obecně platné napříč odvětvími. Před několika lety provedla společnost Capgemini průzkum ceny za opravu defektů v různých fázích projektu,[2] zahrnující několik stovek integračních projektů z celé Evropy.

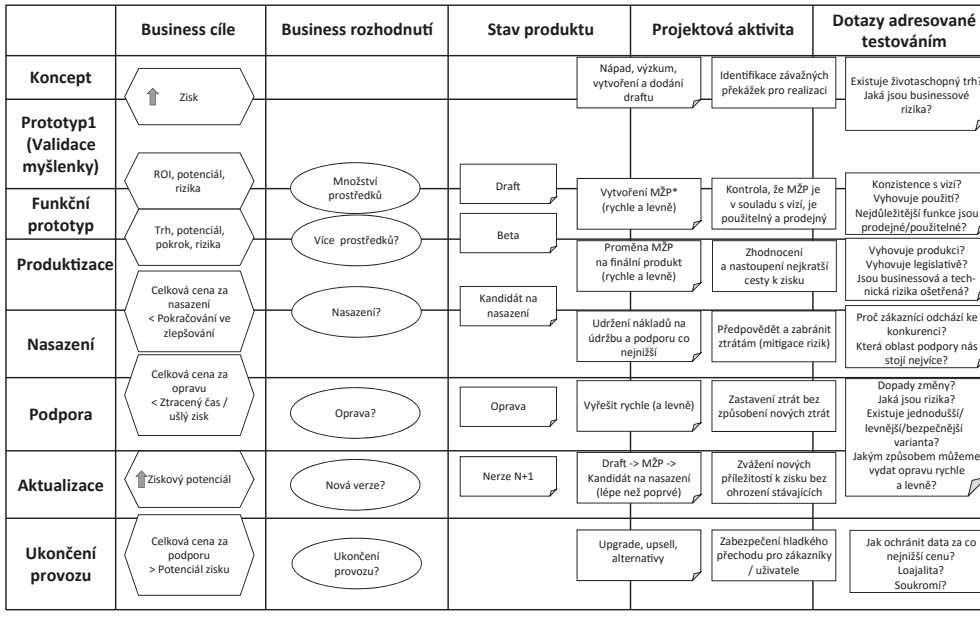
Přímé náklady na defekt, který je nalezen až v produkci, jsou tedy podle tohoto průzkumu v průměru 12,5× vyšší než náklady na odstranění defektu nalezeného v průběhu vývojářských testů. **Pokud do testování zapojíme např. revize analytických dokumentů, můžeme opravu části chyb zlevnit až 50×.**

1.2 Výpočet ceny za kvalitu

Větší IT projekty jsou ve většině případů financované z peněz, nad nimiž mají kontrolu výkonní manažeři společnosti. Mnozí manažeři testování a manažeři projektů si stěžují, že komunikace a jednání s výkonnými manažery je složité. V těchto stížnostech zaznívá, že výkonní manažeři nechtějí slyšet argumentaci a jediné, co je zajímá, je co nejnižší cena. To nemusí být pravda. Je nutné si uvědomit, že manažeři testování a manažeři IT projektů jsou většinou technicky uvažující lidé, upřednostňující rozdílnou argumentaci a vyznávající jiné profesní hodnoty než lidé ve výkonném managementu. Výkonný management často neslyší na argumenty typu „testování sníží rizika", „testování ušetří peníze" (něco nedělat je přeci 100% sleva) nebo na obsáhlou argumentaci popisující technickou složitost řešení a z toho plynoucí technické komplikace. Je třeba argumentovat jinak. Výkonný management zajímá hodnota, kterou za své peníze dostanou.

Kolik tedy má sponzor zaplatit za testy? Tato otázka je v principu špatně položená. **Lepší je ptát se, co chceme pomocí testů dosáhnout? Jaká je hodnota, kterou testy přinesou?**

Odpověď na tuto otázku bude dána namapováním produktů testování na businessové cíle při zohlednění fází životního cyklu produktu. Toto mapování je přehledně zobrazené na obrázku 1.1. Zkratka MŽP znamená minimální životaschopný produkt (angl. *minimal viable product*).



Obrázek 1.1: Mapování produktů testování na businessové cíle při zohlednění fází projektu

[2] Capgemini Europe: Cena za opravu defektů podle fáze, ve které jsou nalezeny, 2010. Dostupné na: http://www.capgemini.com.

*

V praxi se osvědčil při přípravě strategie testování následující postup:

1. Diskuze očekávané hodnoty testování se sponzory – tato aktivita by měla proběhnout jako první, jelikož očekávaná hodnota testování do určité míry definuje cíle testování.
2. Odhad pracnosti testování – jako vstup pro odhadování pracnosti jsou kromě jiného použity výstupy diskuze se sponzorem o hodnotě testování.
3. Validace odhadů se sponzorem – zde už můžete použít pro argumentaci výstupy z úvodní schůzky. Tedy domluvené businessové cíle a náklady na produkty testování, jež je pomáhají naplnit.

A jak můžeme popsat a kvantifikovat cenu za kvalitu (angl. *Cost of Quality*, COQ), zmíněnou v úvodu? Podle standardu ISTQB vstupují do ceny za kvalitu (či přesněji „ceny za nekvalitu") následující kategorie nákladů:

- Náklady využití na prevenci defektů – sem patří např. školení vývojářů, náklady na zavedení kódovacích standardů.
- Náklady na nalezení defektů – zahrnují veškeré testovací aktivity od plánování testů, přípravy testovacích případů, revize specifikací, přípravu testovacích prostředí, jejich vykonávání až po reporting.
- Náklady vyvolané řešením defektů v době testování (angl. *cost of internal failure*) – sem patří náklady na opravu defektů ve specifikacích, opravy defektů v kódu a na ně nabalené náklady dalších rolí (nasazování do prostředí, přetestování atd.).
- Náklady vyvolané selháními v produkčním prostředí (angl. *cost of external failure*) – zahrnují nejen přímé náklady na veškeré aktivity související s opravou **případných** produkčních defektů a distribucí opraveného systému uživatelům. Velkou položkou v této škatulce mohou být náklady na sankce (např. smluvní pokuta účtovaná odběratelem), náklady na ztrátu reputace atd.

Náklady na zajištění kvality před nasazením do produkčního prostředí jsou dané součtem první tří položek:

Cena zajištění kvality = Náklady na prevenci + Náklady na nalezení defektů + Náklady na opravu

Přínos těchto investic do kvality před uvedením systému do produkčního provozu vůči možným nákladům, které by bylo nutné vynaložit, kdybychom tyto investice/aktivity nerealizovali, pak ukazuje cena za kvalitu:

Cena za kvalitu = Náklady vyvolané možným produkčním selháním – Cena zajištění kvality

Pro argumentaci opodstatnění investic do aktivit pro zajištění kvality je tedy pochopitelně nutné, aby byla cena za kvalitu kladným číslem. Zjednodušeně řečeno: má smysl testovat jen do té míry, pokud jsou náklady na testování nižší než náklady na řešení případných produkčních incidentů.

1.3 Prostředky pro argumentaci hodnoty testování z praxe

Pro lepší ilustraci výše uvedeného uvádíme smyšlený příklad inspirovaný příběhem prezentovaným v úvodu kapitoly.

Projekt: Vývoj softwaru řídicí jednotky generátorů proudu v novém typu letounu.

1.3.1 Fáze získání informací

Při diskuzi s architektem řešení, vedoucím vývoje a analytickým týmem **se snažíme získat co nejvíce podrobností o technickém řešení, způsobu vývoje a případných specifikách**. Z těchto diskuzí získáme následující výstupy:

Očekávané parametry vývoje:

- 4,3 milionu řádků kódu,
- vývoj formou „Cleanroom Development",[3]

[3] Cobb & Mills: Engineering Software Under Statistical QualityControl,Science Alliance, 1990.

- statistická chybovost 3 defekty na 1000 řádků kódu (zohledňuje způsob vývoje a zkušenosti z minula),
- odhad počtu chyb přítomných v softwaru řídicí jednotky: 12 900,
- celkové množství kódu ve všech systémech: 110 milionů řádků,
- odhad celkového počtu chyb přítomných v softwaru letounu: 330 000.

Další podklady:

- průměrný nálet na stroj: 25 000 hodin za 10 let.

Při prvotní diskuzi se sponzory **procházíme business cíle projektu a vyjasníme si případné nedorozumění**, abychom byli schopni správně uchopit cíle. Z diskuze získáme následující výstupy:

Business cíle:

- ROI po 1000 prodaných letadlech daného typu,
- splnění regulace **DO-178B, DO-178C**,
- dodání 3000 strojů do deseti let.

1.3.2 Fáze přípravy argumentace

Argumentaci postavíme na **požadavcích regulátora na maximální počty selhání** dílčích závažností na letovou hodinu, které jsou prezentované v druhém sloupci tabulky 1.2.

Při zohlednění vstupních informací získaných v úvodní fázi dostáváme, že v prvních 10 letech bude souhrnný nálet **75 000 000 hodin**. Po vynásobení maximálních regulátorem stanovených přípustných počtů selhání v důsledku dílčích závažností defektů získáváme maximální počty defektů uvedené ve třetím sloupci tabulky. Protože vyvíjíme pouze kód pro řídicí jednotku, je třeba násobit maximální počty přípustných defektů poměrem mezi řádky zdrojového kódu řídicí jednotky (4,3 milionů) a celkovými řádky zdrojového kódu (110 milionů). Tato výsledná hodnota je pak uvedena v posledním sloupci.

Tabulka 1.2: Maximální počty přípustných defektů podle požadavků regulátora

Závažnost defektu	Max. počet selhání za letovou hodinu	Max. počet přípustných defektů pro celé letadlo	Max. počet přípustných defektů pro řídicí jednotku
Katastrofické defekty	0,000000001	0,075	0,00293
Nebezpečné defekty	0,0000001	7,500	0,29318
Závažné defekty	0,00001	750,000	29,31818
Drobné defekty	0,001	75000,000	2931,81818

Z podobných historických projektů známe statistické rozložení pravděpodobnosti výskytu defektů pro jednotlivé závažnosti. Tato hodnota je uvedena v druhém sloupci tabulky 1.3. Prostým součinem očekávaného procenta defektů dílčí závažnosti s celkovým počtem odhadovaných defektů v řídicí jednotce (12 900) získáme hodnoty uvedené ve třetím sloupci. Pro kategorii katastrofických a nebezpečných defektů nám předchozí tabulka ukazuje, že v softwaru řídicí jednotky nesmí být žádný defekt (čtvrtý sloupec). Nižší závažnosti připouští nenulový počet defektů. Ve čtvrtém sloupci tabulky 1.3 tedy vypočítáme, kolik defektů musíme odhalit pro dílčí závažnosti, abychom splnili požadavky regulátora.