VYSOKÁ ŠKOLA EKONOMIE A MANAGEMENTU
Nárožní 2600/9a, 158 00 Praha 5

BAKALÁŘSKÁ PRÁCE

Vysoká škola ekonomie a managementu
info@vsem.cz / www.vsem.cz

VYSOKÁ ŠKOLA EKONOMIE A MANAGEMENTU

Nárožní 2600/9a, 158 00 Praha 5

NÁZEV BAKALÁŘSKÉ PRÁCE/TITLE OF THESIS
Zavedení proaktivního přístupu ke kvalitě softwaru u vybrané společnosti / Implementing a proactive approach to software quality at a selected company

TERMÍN UKONČENÍ STUDIA A OBHAJOBA (MĚSÍC/ROK)
6/2024

JMÉNO A PŘÍJMENÍ STUDENTA / STUDIJNÍ SKUPINA
Radka Roudenská / KEMBC05

JMÉNO VEDOUCÍHO BAKALÁŘSKÉ PRÁCE
Ing. Kamil Hofrichter, MBA

PROHLÁŠENÍ STUDENTA
Odevzdáním této práce prohlašuji, že jsem zadanou bakalářskou práci na uvedené téma vypracoval/a samostatně a že jsem ke zpracování této bakalářské práce použil/a pouze literární prameny v práci uvedené. Jsem si vědom/a skutečnosti, že tato práce bude v souladu s § 47b zák. o vysokých školách zveřejněna, a souhlasím s tím, aby k takovému zveřejnění bez ohledu na výsledek obhajoby práce došlo. Prohlašuji, že informace, které jsem v práci užil/a, pocházejí z legálních zdrojů, tj. že zejména nejde o předmět státního, služebního či obchodního tajemství či o jiné důvěrné informace, k jejichž použití v práci, popř., k jejichž následné publikaci v souvislosti s předpokládanou veřejnou prezentací práce, nemám potřebné oprávnění. Datum a místo: 26.1.2024, Praha

PODĚKOVÁNÍ
Ráda bych tímto poděkovala vedoucímu bakalářské práce Ing. Kamilovi Hofrichterovi, MBA za metodické vedení a odborné konzultace, které mi poskytl při zpracování mé bakalářské práce.

VYSOKÁ ŠKOLA EKONOMIE A MANAGEMENTU

Nárožní 2600/9a, 158 00 Praha 5

SOUHRN

1. Cíl práce:

Posouzení stávajícího modelu životního cyklu doručování softwaru u vybrané společnosti a návrh jeho vhodných zlepšení za účelem redukce chyb (prostřednictvím předcházení jejich vzniku i včasné detekce) ve výsledném produktu a tím snížení nákladů spojených s jejich odstraňováním.

2. Výzkumné metody:

Literární rešerše, srovnávání, analýza dokumentů, syntéza, dedukce.

3. Výsledky výzkumu/práce:

Na základě identifikovaných nedostatků v současném podnikovém modelu a kořenových příčin chyb nalezených na referenčních projektech byla z navržených zlepšení z důvodu rozpočtového omezení sestavena množina 16 z nich, u nichž je předpokládán nejvyšší potenciál předcházet vzniku chyb anebo je včas odhalovat. Tento potenciál vychází především ze skutečnosti, že předcházení chybám i jejich nalézání musí být prováděno co nejdříve po zahájení projektu, neboť čím později jsou chyby objeveny, tím vyšší jsou náklady na jejich odstranění. Zároveň platí, že kvalita požadavků a jejich zpracování je významným faktorem, ovlivňujícím kvalitu softwaru v průběhu projektu.

4. Závěry a doporučení:

Konečná množina zlepšení byla posuzována vůči referenčním projektům, na nichž bylo zjištěno, že potřebná investice se vyplatí u těch projektů, kde náklady plánované na odstranění chyb přesahují 33 000 EUR. Pro využití získaných poznatků v praxi bylo navrženo revidovat podnikový model životního cyklu vývoje softwaru a začlenit navržená zlepšení tak, aby bylo jejich zavedení na vhodných projektech doporučeno.

KLÍČOVÁ SLOVA

Kvalita, software, zlepšení

VYSOKÁ ŠKOLA EKONOMIE A MANAGEMENTU

Nárožní 2600/9a, 158 00 Praha 5

SUMMARY

1. Main objective:

Assessment of the current software delivery lifecycle model of the selected company and design of appropriate improvements to reduce errors (through prevention and early detection) in the final product and thus reduce the costs associated with their rectification.

2. Research methods:

Literary research, comparison, document analysis, synthesis, deduction.

3. Result of research:

Based on the identified weaknesses and gaps in the current delivery model and the root causes of the errors found on the reference projects, a set of 16 improvements, which are assumed to have the highest potential to prevent and/or detect errors in a timely manner, was due to budget constraints compiled from all the proposed improvements. This potential is based primarily on the fact that error prevention and detection must be carried out as early as possible after the start of the project, as the later errors are discovered, the higher the cost of correcting them. At the same time, the quality of the requirements and their processing is absolutely crucial.

4. Conclusions and recommendation:

The final set of improvements was assessed against a set of benchmark projects where the required investment was found to be justified for those projects where the projected cost to rectify defects exceeds €33 000. In order to apply the findings in practice, it was proposed to revise the corporate model of the software development life cycle and to incorporate the proposed improvements so that their implementation on appropriate projects would be recommended.

KEYWORDS

Quality, software, improvement

JEL CLASSIFICATION

L15 - Information and Product Quality, L86 – Computer Software, M15 – IT Management

Vysoká škola ekonomie a managementu
Nárožní 2600/9a, 158 00 Praha 5

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

Jméno a příjmení:	Radka Roudenská
Studijní program:	Ekonomika a management (Bc.)
Studijní skupina:	KEMBC05
Název BP:	Zavedení proaktivního přístupu ke kvalitě softwaru u vybrané společnosti
Zásady pro vypracování (stručná osnova práce):	1. Úvod 2. Teoreticko-metodologická část 2.1. Kvalita softwaru a její ekonomický význam 2.2. Řízení a zajišťování kvality 2.3. Přístupy ke kvalitě dle modelů životního cyklu 2.3.1. Tradiční model 2.3.2. Agilní model 2.4. Metody práce 3. Analyticko-praktická část 3.1. Životní cyklus softwarového produktu u vybrané organizace 3.2. Analýza aktivit zajišťování a řízení kvality 3.3. Návrh zlepšení a posouzení aplikovatelnosti 4. Závěr
Seznam literatury: (alespoň 4 zdroje)	• BAUMGARTNER, M., KLONK, M., MASTNAK, C., PICHLER, H., SEIDL, R. a TANCZOS, S. *Agile Testing The Agile Way to Quality*. Cham: Springer International Publishing, 2021. 257 s. ISBN 978-3-030-73208-0. • GALIN, D. Software quality: *Concepts and practice.* Hoboken, NJ: Wiley, 2018. 720 s. ISBN 978111913449. • LEVIN, M.A., KALAI, T.T. a RODIN, J. *Improving product reliability and software quality:* Strategies, tools, process and implementation. Hoboken, NJ: Wiley, 2019. 456 s. ISBN 9781119179412. • O'REGAN, G. *Concise guide to software testing.* Charm: Springer, 2019. 317 s. ISBN 9783030284930.
Harmonogram:	• Zpracování cílů a metodiky do 1. 1. 2024 • Zpracování teoretické části do 1. 2. 2024 • Zpracování výsledků do 1. 3. 2024 • Finální verze do 1. 4. 2024
Vedoucí práce:	Ing. Kamil Hofrichter, MBA

prof. Ing. Milan Žák, CSc.
rektor

V Praze dne 5. 9. 2023

Prof. Ing. Milan Žák CSc.
Digitálně podepsal Prof. Ing. Milan Žák CSc.
DN: cn=Prof. Ing. Milan Žák CSc., c=CZ, o=Vysoká škola ekonomie a managementu, a.s., givenName=Milan, sn=Žák, serialNumber=JCA-10393535
Datum: 2023.09.05 10:50:32 +02'00'

Obsah

Seznam zkratek:

CMM	Capability Maturity Model
ČSN	České technické normy
DFW	Delivery Framework
IEEE	Institute of Electrical and Electronics Engineers
ISO	International Organization for Standardization
MD	Man Day
PPM	Portfolio and Project Management
SAP	Systeme, Anwendungen, Produkte
SEI	Software Engineering Institute

1 Úvod

V konkurenčním obchodním prostředí současného globalizovaného světa má kvalita dodávaných produktů pro výrobce mimořádný význam. Z krátkodobého hlediska představuje kvalita zboží klíčový faktor, rozhodující mezi vzájemně soupeřícími nabídkami na trhu. Zákazníci jsou stále náročnější a více informovaní, a proto jsou schopni rozpoznat a ocenit kvalitní výrobek, což může ve výsledku vést k rychlejšímu nárůstu prodejů a získání příznivější pozice ve srovnání s konkurencí.

Z hlediska dlouhodobého má investice do kvality výrobků strategický dopad. Kvalitní produkty pozitivně ovlivňují reputaci společnosti jakožto spolehlivého a odpovědného výrobce. Taková pověst pak může přitahovat nejen občasné zákazníky hledající jednorázový nákup určitého zboží, ale i zákazníky, kteří hledají spolehlivého výrobce a zůstavají mu dlouhodobě věrni. Opakované nákupy a věrnost zákazníků jsou tak do značné míry založeny na důvěře v kvalitu nabízených výrobků nebo služeb, což umožňuje výrobci vytvářet stabilní zákaznickou základnu a upevňovat svou pozici na trhu i v dlouhodobém horizontu.

Jedinečnou kategorií zboží je pak počítačový software, který se stává stále více neodmyslitelnou součástí každodenního života v moderní společnosti. Jeho vliv proniká prakticky do všech jejích sfér, zahrnující osobní zařízení, oblast vzdělávání, obchodu, zábavy, vědy, zdravotnictví až po kritickou infrastrukturu. Tato široká škála aplikací naznačuje, jak software ovlivňuje prakticky každý aspekt našeho života, a jak je tedy jeho kvalita pro nás, přímo či nepřímo, důležitá.

Oproti fyzickým produktům je však kvalita softwaru odlišná, což je dáno nemateriální, abstraktní povahou softwaru samotného. Zatímco u hmotných výrobků je možné měřit určité jejich fyzické charakteristiky a vlastnosti, u softwaru spočívá klíčové měřítko kvality v jeho schopnosti naplňovat požadavky, potřeby a očekávání uživatelů.

Ekonomický aspekt kvality softwaru je pro podniky zabývající se jeho výrobou velice důležitý, neboť kromě obecných přínosů diskutovaných výše vede vyšší kvalita produktu ke snižování nákladů spojených s opravami chyb a nedostatků, poskytovanou podporou a ztrátami způsobenými újmou na pověsti a případným ušlým ziskem.

Při vývoji softwarových produktů je klíčové zajištění kvality prostřednictvím proaktivních opatření, která předchází samotnému vzniku chyb, jež jsou ve větší či menší míře kombinována s aktivitami řízení kvality, které se naopak zaměřují na identifikaci existujících chyb tak, aby mohly být odstraněny s co nejnižšími náklady. Oba typy aktivit jsou v různé podobě začleněny do systematického přístupu, představovaného jednotlivými modely životního cyklu vývoje softwaru. Protože každý z těchto modelů je použitelný za určitých podmínek, je volba správného z nich – nebo vytvoření vlastního – klíčová pro efektivní realizaci projektu.

Tato bakalářská práce se zabývá posouzením aktivit zajišťování a řízení kvality v hybridním modelu životního cyklu vývoje softwaru u vybrané společnosti, přičemž jejím cílem je návrh takových zlepšení, jež by vedla k vyšší kvalitě výsledných produktů, respektive snížení počtu chyb a tedy i souvisejících nákladů, které jsou vynakládány za účelem odstranění chyb, ověření správnosti provedených oprav, přípravě opravného balíčku a jeho dodatečné poskytnutí zákazníkům.

Při zpracování práce byly nejprve v rámci teoretické části shromážděny informace týkající se samotného konceptu kvality a jejich specifik v kontextu nehmotných, abstraktních produktů jakým je počítačový software. U kvality softwaru bylo dále zkoumáno její vymezení a definice,

jednotlivé charakteristiky, z nichž se skládá a zejména pak její ekonomický význam pro organizaci, která se produkcí softwaru zabývá.

Následně byly charakterizovány procesy zajišťování a řízení kvality a jednotlivé činnosti, prostřednictvím kterých jsou během vývojového cyklu softwaru realizovány, kde byly uvažovány i odlišné přístupy mezi některými citovanými autory. Na závěr teoretické části byly popsány základní modely životního cyklu vývoje softwaru a jejich odlišný přístup k zajišťování a řízení kvality, kde byl kladen důraz rozlišení mezi tradičním a moderním, agilním, pojetím softwarového vývoje.

V následující analyticko-praktické části bylo nejprve provedeno zjištění stávajícího stavu v oblasti zajišťování a řízení kvality v dané společnosti. Za tímto účelem bylo nutné porozumět používanému specifickému, na míru vytvořenému modelu životního cyklu vývoje softwaru, který podnik používá a zhodnotit jej s ohledem na doporučenou praxi a principy platné pro kvalitu softwaru zjištěné v teoretické části práce. Dále byly shromážděny informace o kvalitě a nákladech s ní spojených na vzorku referenčních projektů, což umožnilo stanovení výchozího stavu.

Výše uvedené informace byly posléze použity pro vytvoření seznamu aktivit zajišťování a řízení kvality v podnikovém modelu, jehož součástí byly nedostatky identifikované jak ve vztahu k doporučené praxi, tak s ohledem na kořenové příčiny chyb zjištěných na referenčních projektech. Na základě uvedeného seznamu byla pro každou aktivitu navržena taková zlepšení, jejichž aplikace měla odstranit nebo zmírnit příslušný nedostatek. Při návrhu zlepšení byla uvažována i nákladová efektivnost jejich zavedení.

V závěru práce byla posuzována návratnost investice do navržených zlepšení na příkladech jednotlivých referenčních projektů.

2 Teoreticko-metodologická část práce

2.1 Kvalita softwaru a její ekonomický význam

Než bude blíže představena kvalita softwaru a její základní koncepty, je vhodné nejprve vymezit kvalitu v obecné rovině.

Samotný pojem „kvalita“ vychází z latinského slova *qualis*, jež bylo odvozeno z řeckého *poiótēs*. Tento výraz poprvé použil Platón, jeden z nejvýznamnějších antických filozofů, když jej vytvořil ze slova *poiós*, které lze do češtiny přeložit jako „jakého druhu“ či „jakého charakteru“ (Bausch 2022, s. 67).

Jde tedy o vyjádření toho, *jaké* povahy, *jakého* charakteru či *jakých* vlastností v souhrnu určitá věc je. A stejně jako je v českém jazyce kupříkladu ze slova „rychlý“ vytvořeno podstatné jméno „rychlost“, ze slova „jaký“ lze odvodit substantivum „jakost“, jež je v češtině výstižným synonymem pojmu kvalita.

Chápání a definice toho, co vlastně kvalita znamená, není univerzální, neboť jde o do značné míry subjektivní pojem, závislý na vnímání jednotlivce – co jeden považuje za dobrý produkt, může být pro jiného zcela nevyhovující (Filip 2019, s. 87).

Mnohotvárnost toho, co je kvalita a jak na ni lze nahlížet, lze demonstrovat na dobře známém rámci pro určování kvality od předního harvardského profesora Davida A. Garvina, který definoval pět různých aspektů kvality a přístupů, podle kterých lze postupovat (Sharma 2019, s. 664):

- transcendentální přístup: tento přístup považuje kvalitu výrobku nebo služby za vnitřní vlastnost, která je absolutní a všeobecně rozpoznatelná. Transcendentální kvalita je podobná Platónovu pojetí krásy jako „ideální formy", což však neposkytuje žádné praktické kroky pro její hodnocení;
- přístup založený na produktu: v tomto případě je kvalita výrobku nebo služby považována za kvantifikovatelnou na základě určitých složek nebo atributů. Garvin tento přístup ilustruje na zmrzlině a kobercích. Například zmrzliny lze hodnotit podle obsahu určité složky, přičemž vyšší obsah znamená vyšší kvalitu. Koberce lze zase hodnotit podle počtu uzlů na čtvereční palec, přičemž vyšší kvalitu pak představuje hustší vazba;
- přístup založený na uživateli: tento přístup vychází z předpokladu, že kvalita je „v očích pozorovatele", přičemž pozorovatelem je uživatel. Podle tohoto přístupu je kvalita mírou, do jaké výrobek nebo služba uspokojuje potřeby, přání nebo preference uživatele;
- přístup založený na výrobě: tento přístup, který Garvin označil jako „výrobní přístup", chápe kvalitu jako „shodu s požadavky". Podle tohoto přístupu tedy kvalitu snižuje jakákoli odchylka od určených požadavků;
- přístup založený na hodnotě: v tomto přístupu se kvalita posuzuje z hlediska nákladů a přínosů: čím více převažují přínosy nad náklady, tím více se zvyšuje hodnota výrobku nebo služby. Produkty nebo služby s vyšší hodnotou pak mají vyšší kvalitu.

Tyto přístupy by se měly vzájemně doplňovat, to však komplikuje skutečnost, že některé ze své podstaty působí proti sobě. Například podle přístupu založeného na výrobě určuje kvalitu prostá shoda s požadavky, zatímco u přístupu založeného na uživateli je za kvalitní považován takový produkt nebo služba, které uspokojují potřeby zákazníka, což je nezávislé na tom, zda je dosaženo shody s výrobcem určenými požadavky či nikoli. Kupříkladu deštník, jehož barva a

rozměr neodpovídá zcela výrobní specifikaci, může být zákazníkem považován za kvalitní, jestliže jej ochránil při dešti.

Na základě výše uvedeného pak není překvapivé, že jednoznačná a všeobecně platná definice kvality neexistuje. Významné osobnosti managementu kvality definovali v minulosti kvalitu následovně (Tsigkas 2021, s. 113):

- „kvalita je vhodnost pro zamýšlené použití“ – Joseph M. Juran;
- „kvalita je soulad s požadavky“ – Philip B. Crosby;
- „kvalita je to, co za ni považuje zákazník.“ – Armand V. Feigenbaum.

V dnešní době je mnohdy používána definice vycházející z mezinárodního standardu ISO 9000, který je v České republice dostupný jako ČSN EN ISO 9000:2016. Ten definuje kvalitu *jako stupeň splnění požadavků souborem inherentních charakteristik objektu* (Filip 2019, s. 87). Zde je vhodné zmínit, že nejde o požadavky pouze ve smyslu specifikací výrobce, protože samotný požadavek je touto normou definován jako „potřeba nebo očekávání, které je deklarované, obecně předpokládané nebo povinné” (Makiela, Stuss a Borowiecki 2022, s. 299), což tak zahrnuje i pojetí kvality s důrazem na očekávané potřeby zákazníka.

Norma ISO 9000 je použitelná všeobecně napříč všemi odvětvími a typy podniků, jež poskytují služby nebo vyrábí produkty, ovšem organizace zabývající se vývojem softwaru jsou od ostatních zásadně odlišné – software je jedním z mála lidských výtvorů, který je komplexní, neviditelný, nehmotný a je snadné jej modifikovat, ovšem zcela unikátním jej činí skutečnost, že bez zásahu (úprava kódu) zůstává jeho chování trvale neměnné (Rosen 2020, s. 18).

Tato jedinečnost zároveň znamená, že software nemůže být posuzován jako jiné produkty – při hledání odchylek jej nelze jednoduše změřit nebo zvážit, podrobit vizuální inspekci nebo rozebrat a posoudit opotřebení. Navíc oproti výstupní kontrole výrobků v továrně, kde je u každého z nich hodnocen předem soubor určitých vlastností oproti požadovaným hodnotám, je software specifický také tím, že žádné dva produkty nejsou totožné.

Je tak zřejmé, že klasické definice a přístupy ke kvalitě nelze na softwarové produkty efektivně aplikovat, což bylo důvodem vzniku různých norem a rámců, které se specificky zabývají kvalitou softwaru a jejím měřením.

První oficiální, mezinárodní normou zabývající se kvalitou softwaru se v roce 1991 stala norma ISO 9126, revidováno roku 2001, jež byla později nepřímo nahrazena řadou norem ISO 25000 (Delhaye 2021, s. 103), kde je kvalita softwaru definována jako *schopnost softwarového produktu splňovat stanovené a předpokládané potřeby při použití za stanovených podmínek* (Laporte a April 2018, s. 569).

Oproti dříve uvedené definici podle normy ISO 9000 tato nepracuje s pojmem „požadavky“ a výslovně zmiňuje naplnění stanovených a předpokládaných potřeb, což dokládá vyšší důraz na skutečnou vhodnost pro zamýšlené použití oproti pouhému souladu s požadavky, přičemž tento přístup je blízký výše uvedenému pojetí kvality dle Josepha M. Jurana.

Toto pojetí ve své podstatě reflektuje i pragmatický přístup ke kvalitě softwaru v praxi: softwarový produkt nemusí být perfektní, avšak musí být považován za dostatečně dobrý jeho uživateli a ostatními zainteresovanými stranami (Wiegers 2021, s. 33).

Přestože výše zmíněná definice dle normy ISO 25000 je srozumitelná, sama o sobě neříká nic o tom, jak lze vlastně onu „míru naplnění potřeb uživatele“ měřit a hodnotit. V praxi se tak produkt posuzuje s ohledem na dvě základní kategorie požadavků (Hilburn a Towhidnejad 2021, s. 126):

- funkční požadavky: výslovně stanovují, co musí systém vykonávat, respektive jakou konkrétní funkci musí poskytovat. Funkční požadavky tak definují, „co“ produkt poskytuje ve smyslu jeho funkčnosti, a při jejich hodnocení mohou nastat pouze dva stavy: požadavek je splněn anebo nikoli (Rosen 2020, s. 113). Funkční požadavky jsou dokumentovány typicky ve funkční specifikaci nebo zadávací dokumentaci. Jako příklad funkčních požadavků lze uvést následující věty: „Uživatel může zadat částku v rozmezí hodnot 1 až 999“ nebo „Je-li zadaná částka vyšší než nastavený limit platby, systém zobrazí varování“;
- mimofunkční požadavky: se netýkají toho, „co“ produkt poskytuje, ale „jakým“ způsobem – jak rychle, jak spolehlivě, jak bezpečně, jak uživatelsky přívětivě a podobně. Tyto požadavky, které jsou také někdy nazývány charakteristiky kvality, mnohdy nebývají formálně dokumentovány, protože zadavatel je považuje za samozřejmé, což může představovat zásadní problém (Hilburn a Towhidnejad 2021, s. 126). Na rozdíl od funkčních požadavků, mimofunkční požadavky může být obtížné až nemožné vyhodnotit, jestliže je zákazník nespecifikuje blíže nebo neposkytne měřitelná kritéria (Rosen 2020, s. 113). Je-li například v zadání uvedeno, že systém musí být rychlý a uživatelsky přívětivý, pak v podstatě není možné zaručit, že bude zákazník spokojen, protože pro každého může být představa rychlosti nebo přívětivosti odlišná.

Protože mimofunkčních požadavků je velké množství, výše zmíněná norma ISO 25000 poskytuje jejich následující členění, které umožňuje hodnotit kvalitu softwaru v jednotlivých kategoriích (Bierig et al. 2021, s. 3):

- funkční přiměřenost: míra, do jaké produkt zabezpečuje funkce, které splňují stanovené a předpokládané potřeby při použití za stanovených podmínek;
- bezpečnost: míra, do jaké jsou informace a data chráněna tak, aby nedošlo k jejich neoprávněné změně nebo přístupu neautorizovanou osobou nebo systémem;
- kompatibilita: míra, do jaké si dva (nebo více) systémy nebo jejich součásti mohou vyměňovat informace a/nebo vykonávat své požadované funkce během sdílení stejného hardwarového nebo softwarového prostředí;
- použitelnost: míra, do jaké může být výrobek používán určenými uživateli k dosažení stanovených cílů účinně, účelně a uspokojivě ve stanoveném kontextu používání;
- přenositelnost: míra, do níž může být produkt nebo jeho součást účinně a účelně převeden z jednoho hardwaru, softwaru nebo jiného prostředí do druhého;
- spolehlivost: míra, do jaké produkt nebo jeho součást plní stanovené úkoly v rámci stanovených podmínek na určitou dobu. Spolehlivost je vlastně vyjádřením trvanlivosti kvality v čase (Levin, Kalai a Rodin 2019, s. 51);
- udržovatelnost: míra účinnosti a účelnosti, s nimiž může být produkt upraven za účelem jeho opravy, vylepšení nebo přizpůsobení změnám prostředí nebo požadavků;
- efektivita výkonnosti: míra výkonnosti ve vztahu k výši prostředků využívaných za stanovených podmínek.

Každou z uvedených charakteristik pak norma ISO 25000 dále člení na skupinu měřitelných podcharakteristik.

2.1.1 Náklady na kvalitu softwaru

Vzhledem k tomu, že softwarové systémy jsou součástí všech aspektů života moderní společnosti, náklady spojené s chybami v nich jsou enormní – jen ve Spojených státech amerických byla odhadnuta každoroční ztráta způsobená chybným softwarem na 59,5 miliard

USD, přičemž u třetiny (asi 22.2 miliard USD) bylo potvrzeno, že jí bylo možné předejít provedením testování alespoň základních funkcí (Jun et al. 2021, s. 55).

Společnost dodávající vysoce kvalitní software získává důvěru svých zákazníků a nad ostatními podniky konkurenční výhodu, zatímco dodavatel nedostatečně kvalitních softwarových produktů může čelit ztrátě reputace a obchodních příležitostí, poklesu prodejů, vysokým nákladům na opravy a kompenzace zákazníků a v krajních případech i právním sporům (Bierig et al. 2021, s. 3).

Na obecné rovině ovlivňuje celkové náklady spojené s kvalitou softwarových produktů řada faktorů, z nichž mezi ty nejdůležitější patří následující (Hilburn a Towhidnejad 2021, s. 57):

- celková velikost, složitost a rizikovost vyvíjeného produktu;
- kritičnost produktu, od které se odvíjí ztráta způsobená jeho případným selháním, jež je samozřejmě nejvyšší v případech, kdy může dojít k ohrožení života a zdraví lidí nebo velkým škodám na majetku;
- kompetence vývojového týmu výrobce produktu, čili programátorů, analytiků, testerů a dalších rolí;
- efektivita technik, nástrojů a postupů použitých při vývoji a testování produktu, kdy například automatizované testování nebo analýza zdrojového kódu může ve výsledku náklady na kvalitu snížit.

Z perspektivy vynaložených nákladů lze veškeré činnosti, které souvisí s kvalitou softwaru, rozdělit do čtyř následujících kategorií (Hilburn a Towhidnejad 2021, s. 57):

- náklady na prevenci: do této kategorie spadají veškeré náklady na aktivity, které vývojový tým podniká s cílem zamezit vzniku a zavlečení chyb do produktu. Sem je možné řadit například náklady na návrh a nastavení procesů, směrnic a standardů, provádění interních auditů či revize činností, náklady na potřebné vybavení, licence používaných nástrojů a podobně;
- náklady na hodnocení: tato kategorie obsahuje náklady na všechny aktivity, které jsou prováděny za účelem posoudit kvalitu produktu a nalézt chyby. Nejtypičtější aktivitou z této kategorie je testování softwaru, mezi další patří například revize dokumentů či inspekce zdrojového kódu;
- vnitřní náklady spojené se selháním: tato kategorie obsahuje náklady, jež musí organizace vynaložit v souvislosti s chybami, které jsou nalezeny během vlastního testování. Sem jsou řazeny pochopitelně náklady na opravu samotné chyby, testování pro potvrzení správnosti opravy, případné úpravy testů a podobně;
- vnější náklady spojené se selháním: tato kategorie obsahuje náklady, jež musí organizace vynaložit v souvislosti s chybami, které se projevily po předání produktu zákazníkovi. Mimo nákladů na samotnou opravu sem bude patřit také cena dodání opravy zákazníkovi a jejího nasazení do produkčního prostředí, což pochopitelně může negativně ovlivnit provoz zákazníkova systému. Mezi tyto náklady tak bude řazen i případný finanční postih (pokuta) a újma na reputaci dodavatele, ačkoli tu zpravidla nelze přesně vyčíslit.

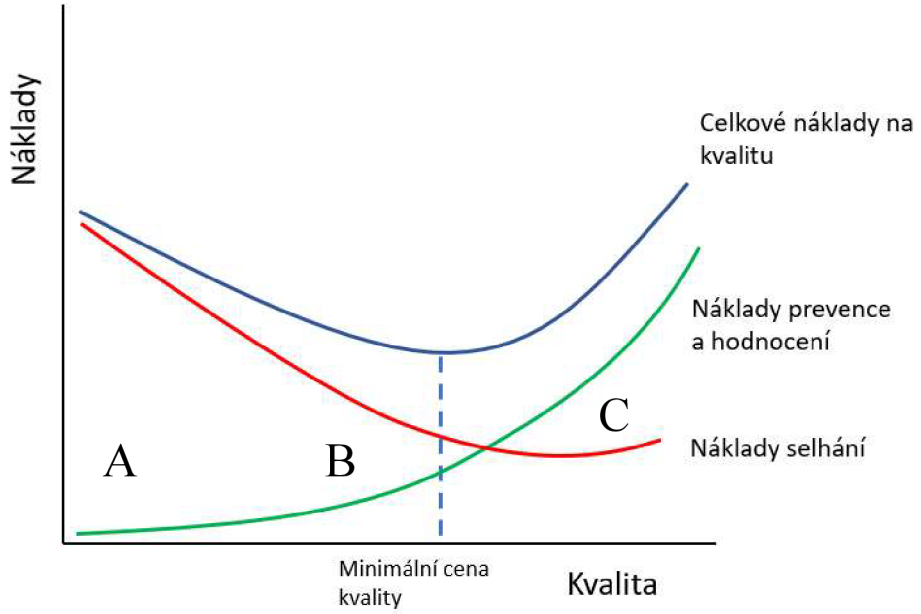
Pokud organizace nepřistupuje ke kvalitě jí vyvíjeného softwaru proaktivně a neinvestuje do preventivních opatření (náklady na prevenci), musí alespoň dostatečně důkladně ověřovat, že má produkt odpovídající kvalitu (náklady na hodnocení), aby se vyhnula ztrátám způsobeným chybami (Galin 2018, s. 167).

Prevence, tedy předcházení chybám, je ekonomicky výhodnější než následné hodnocení produktu, opravování nalezených chyb a následné potvrzování, že byly opravy úspěšné (Lewis

2020, s. 78). V praxi vede investice do preventivních opatření v důsledku k tomu, že produkt obsahuje méně chyb, a dochází tedy jak ke snížení nákladu na hodnocení, které již nemusí být tolik důkladné, tak i k redukci nákladů souvisejících se selháním, neboť i neodhalených chyb je méně.

Popsané vztahy mezi jednotlivými kategoriemi nákladů na kvalitu jsou zobrazeny na grafu 1 níže:

Graf 1 Kategorie nákladů na kvalitu a jejich vzájemný vztah



Zdroj: upraveno z Cassy et al. 2022, s. 383

Uvedené lze demonstrovat na příkladě produktu, který je předán zákazníkovi s množstvím chyb, jež nebyly objeveny během vývoje, protože podnik nevynakládá dostatečné prostředky na testování a další aktivity. Chybovost produktu a tedy jeho nízkou kvalitu tak zjistí zákazník v rámci zkušebního nebo ostrého provozu a požaduje nápravu, což vede k vysokým nákladům selhání (oblast A na grafu 1).

Pokud následně podnik začíná investovat do lepšího hodnocení kvality a především pak efektivního předcházení chybám, kvalita produktu se zvyšuje a náklady selhání klesají (oblast B). Tento pokles nákladů selhání však postupně zpomaluje a od určitého okamžiku může opět růst (oblast C), například protože opravy chyb jsou spojeny s již příliš nákladnými procesy údržby (Cassy et al. 2022, s. 383).

Náklady na prevenci chyb a hodnocení produktu lze teoreticky zvyšovat až do té doby, dokud není dosaženo produktu s maximální kvalitou. V praxi je však typicky rozhodujícím faktorem ekonomická výhodnost, tedy taková kombinace nákladů na selhání a nákladů na prevenci a hodnocení, která má v součtu nejnižší cenu. Na obrázku 1 výše představuje součet obou složek nejvýše položená křivka (Celkové náklady na kvalitu), přičemž její nejnižší bod je tedy minimální výše celkových nákladů, které je nutné na kvalitu vynaložit. Jakákoli jiná kombinace bude dražší za současně nižší anebo vyšší kvality.

Nízké náklady na prevenci chyb a hodnocení produktu však samy o sobě neznamenají, že podnik neprodukuje kvalitní software. Vzhledem k dříve diskutovaným faktorům ovlivňujícím kvalitu softwaru tak lze očekávat, že zkušený a dobře organizovaný tým, efektivně využívající dostupné zdroje, osvědčené techniky vývoje a nástroje (například automatizaci), dokáže na základě dostatečně podrobného zadání vytvořit kvalitní produkt bez dalších dodatečných

nákladů. A naopak pouhé navýšení nákladů na prevenci chyb a hodnocení produktu nezaručuje zlepšení jeho kvality, pokud není zajištěno jejich účelné a efektivní využití. Například mohou být zakoupeny nevhodné nástroje či nejsou používány správně, prováděné audity nenásleduje zavedení zlepšení anebo jsou automatizovány nevhodně zvolené testy (Baumgartner et al. 2022, s. 38).

U nákladů na hodnocení produktu je pro jejich efektivitu zásadní, jakým způsobem jsou příslušné hodnotící aktivity rozložené do vývojového cyklu, neboť platí následující: čím dříve je chyba odhalena, tím nižší jsou náklady na její odstranění (Hilburn a Towhidnejad 2021, s. 54). Z pohledu fází vývoje, které se standardně zobrazují chronologicky zleva doprava, je tedy důležité řešit kvalitu co nejdříve, čili ve fázích co nejvíce vlevo. Tento princip je obecně označován jako „posun vlevo“ (Spillner a Linz 2021, s. 37).

Tuto klíčovou skutečnost lze demonstrovat následujícím příkladem, kdy je v zadání chybně formulován požadavek zákazníka na funkci produktu, například obsahuje chybný vzorec nebo je v protikladu s jiným požadavkem.

Pokud je tato chyba odhalena ještě než před tím, než tým dodavatele začne pracovat na návrhu produktu, její oprava je možná prakticky bez jakýchkoli nákladů – zákazník pouze upraví požadavek. Jestliže je chyba zjištěna ve fázi, kdy již existuje návrh produktu, pak je nutné jej přepracovat tak, aby odpovídal opravenému požadavku. To s sebou již nese určité náklady, protože musí být přepracovány různé modely, dokumenty nebo diagramy, ovšem dosud neexistuje samotný produkt, který by bylo nutné upravit.

Není-li chyba zjištěna v předchozích fázích, produkt začne být vytvářen podle nesprávného požadavku. Jestliže je chyba odhalena během vývojových prací, pak je kromě přepracování návrhu nutné upravit i případnou již existující část produktu.

V případě, kdy není chyba zjištěna ani během vývoje, je dokončený produkt podroben testování. Pokud nyní dojde k odhalení chyby, pak je kromě úpravy požadavku a přepracování návrhu nutné opravit/přepracovat a znovu otestovat i produkt, což je podstatně náročnější, protože na dané funkci mohou být závislé jiné prvky a podobně.

Nejvíce nákladná je však oprava v situaci, kdy je chyba zjištěna až poté, co je produkt předán zákazníkovi a uveden do provozu. V takovém případě je nutné vynaložit náklady na všechny kroky uvedené v předchozích fázích (stále je nutné přepracovat návrh a opravit samotný produkt), k nimž je nutné připočítat dodatečné náklady spojené s novým předáním dodávky zákazníkovi a podporou nasazení do provozu. To pochopitelně vyžaduje i součinnost zákazníka a omezení jeho činnosti, což představuje další náklady, jejichž pokrytí může být požadováno po dodavateli.

Výše uvedené je navíc umocněno skutečností, že 50-60 % všech chyb odhalených v softwarových produktech má svůj původ právě v požadavcích (Hilburn a Towhidnejad 2021, s. 53), a bylo je tedy možné odstranit za minimálních nákladů. Podle některých autorů jsou náklady na opravu chyby, která má svůj původ v požadavcích a je odhalena až během provozu produktu, čtyřicetkrát vyšší, než kdyby byla opravena během revize požadavků (O’Regan 2022, s. 119).

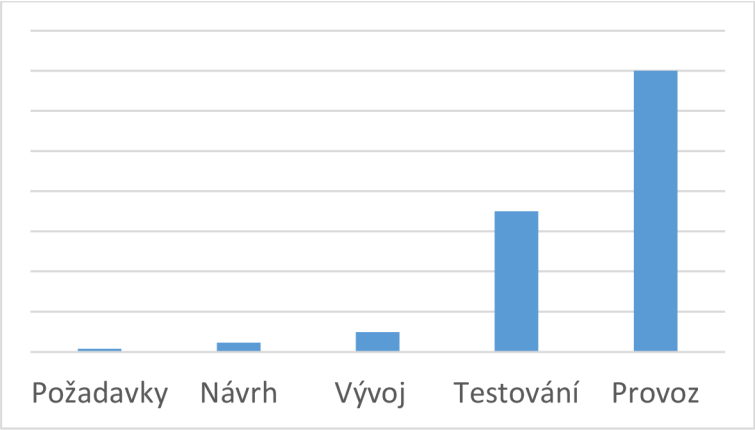
Ve výsledku tak lze říct, že kontrola správnosti požadavků, kam lze řadit například jejich úplnost, jednoznačnost a podobně, je z pohledu kvality a nákladů na ni klíčová.

U chyb, jež jsou do produktu zavlečeny pochybením při návrhu nebo samotném programování produktu pak přirozeně platí, že nejefektivnější je jejich odstranění ve stejné fázi, v jaké došlo k jejich vzniku, aby bylo zamezeno jejich šíření do navazujících aktivit a tím později nutnosti rozsáhlejšího přepracování. V momentě, kdy se například chyba návrhu dostane

k programátorovi, nebo chyba programátora k testerovi v rámci dokončeného produktu, oprava již vyžaduje i změnu v předchozím kroku, jak bylo popsáno výše.

Tento vztah mezi náklady na opravu chyby a fází vývoje produktu, ve které je chyba odhalena, demonstruje graf 2 níže:

Graf 2 Náklady na opravu chyby dle fáze vývoje



Zdroj: upraveno z Hilburn a Towhidnejad 2021, s. 54

Důležitým argumentem, který podporuje nutnost odhalit chyby v produktu co nejdříve, je také zjištění, že chyba v jedné fázi mnohdy vede k více chybám ve fázích následujících (Hilburn a Towhidnejad 2021, s. 53).

Popsaná zjištění týkající se ekonomického aspektu kvality softwaru lze shrnout následovně:

- optimálním přístupem je prevence, tedy předcházení vzniku chyb ve všech fázích vývoje softwaru;
- chyby, které jsou již přítomny v produktu (nebo v dokumentech potřebných pro jeho vytvoření) je nutné najít co nejdříve, přičemž náklady na jejich odstranění jsou nejnižší tehdy, jsou-li odstraněny vždy v té fázi, kdy došlo k jejich vzniku.

2.2 Řízení a zajišťování kvality

Veškeré aktivity a opatření v organizaci, které jsou prováděny v souvislosti s kvalitou poskytovaných produktů nebo služeb, souhrnně spadají do oblasti managementu kvality. Tato funkce zodpovídá za definici podnikových cílů kvality, její plánování, zlepšování a zajišťování a řízení (Spillner a Linz 2021, s. 46).

Poslední dva zmíněné procesy jsou klíčové pro měření, hodnocení a dosahování splnění požadavků na kvalitu u vyvíjeného produktu nebo služby, a jsou proto blíže popsány níže:

- řízení kvality (quality control): proces, kterým je u produktu ověřováno dosažení stanovených požadavků na kvalitu. Řízení kvality je tedy orientováno na potvrzení souladu mezi produktem a jeho specifikacemi a identifikaci veškerých odchylek – chyb. Jde o proces reaktivní, protože je prováděn až poté, co je produkt nebo jeho část dokončena a poskytuje tak pouze informace o tom, do jaké míry jsou požadavky splněny (Lewis 2020, s. 14), kvalitu však sám o sobě nijak ovlivnit nemůže;
- zajišťování kvality (quality assurance): proces zaměřený na poskytování jistoty, že požadavky na kvalitu produktu budou splněny (Lewis 2020, s. 14). Tuto definici lze více rozvést následovně: funkcí zajišťování kvality je poskytnout vedení podniku nebo zúčastněným stranám jistotu, že výrobní procesy jsou nastaveny a fungují tak, že výsledný produkt bude splňovat požadavky na kvalitu, jež jsou na něj kladeny. To

v praxi znamená, že zajišťování kvality cílí na nastavení adekvátních procesů a jejich zlepšování tak, aby umožňovaly a přispívaly ke kvalitnímu produktu. Z toho je patrné, že jde proces především proaktivní a preventivní, neboť prostřednictvím dobře nastavených, vhodných procesů lze nejen zlepšit nacházení chyb, ale především předcházet jejich vzniku.

Ačkoli je řízení kvality orientováno na produkt a zajišťování kvality na proces, úzce spolu souvisí, neboť chyby v produktu lze chápat jako nedostatky v procesu. To znamená, že výstup z řízení kvality může být zároveň cenným vstupem pro zlepšení procesů v rámci zajišťování kvality (Lewis 2020, s. 18). V praxi jsou tedy oba přístupy ke kvalitě často kombinovány a používány komplementárně.

V této oblasti existuje značné množství odlišných pojetí a přístupů mezi autory odborných publikací, technickými normami nebo procesními rámci. Některými je zajišťování kvality chápáno jako nadřazené a obsahující řízení kvality jako svou součást (Lewis 2020, s. 14), jiní ji v souladu s definicemi normy ISO 9000 chápou jako součást managementu kvality, rovnocennou s řízením kvality (Theisens 2020, s. 82).

2.2.1 Činnosti řízení kvality

Proces řízení kvality by neměl posuzovat pouze samotný výsledný produkt, ale i všechny dílčí meziprodukty, které jsou během vývoje vytvořeny – specifikace požadavků, návrh řešení, datové modely, zdrojový kód, uživatelský manuál a podobně (Lewis 2020, s. 17). Každý z těchto meziproduktů by měl být v ideálním případě zkontrolován před tím, než bude použit, aby bylo možné odstranit případné chyby s minimálními náklady, jak bylo diskutováno v předchozí podkapitole.

Jestliže má být například provedena důkladná revize specifikace požadavků, ověřuje se úplnost jejich zadání, jednoznačnost, vzájemná konzistence, proveditelnost a testovatelnost (Laplante a Kassab 2022, s. 148). Důsledkem toho lze odhalit řadu chyb, jež by se jinak rozšířily do návrhu řešení a případně až do samotného produktu, kde by bylo jejich odhalení a oprava výrazně nákladnější.

Řízení kvality je realizováno řadou činností, které jsou stručně popsány v následujícím seznamu:

- revize dokumentů: aktivita, která je zaměřena na přezkoumání obsahu a/nebo formy určitého dokumentu nebo jeho části. Pro dosažení optimálních výsledků by měl být použit systematický přístup, kdy jsou jasně stanovena posuzovaná kritéria a kontrolní seznamy, podle nichž je postupováno. Spolehlivost revize závisí na tom, zda je prováděna kompetentními pracovníky, kteří mají na její provedení dostatek času (Witte 2022, s. 90);
- inspekce zdrojového kódu: aktivita, během níž je posuzován zdrojový kód softwarového produktu za účelem odhalení chyb, ale i sdílení znalostí v týmu (Witte 2022, s. 91). Ve své podstatě jde o specifický typ revize dokumentů, který může být prováděn s různou mírou formálnosti – od zcela neformálního připomínkování přes strukturované prohlížení a hodnocení až po velmi formální inspekci s přesně stanovenými pravidly (O'Regan 2022, s. 117);
- statická analýza zdrojového kódu: aktivita, během níž je zdrojový kód produktu analyzován automaticky pomocí speciálních programových nástrojů – analyzátorů kódu. Výsledky a spolehlivost statické analýzy závisí na sofistikovanosti použitého nástroje (O'Regan 2022, s. 133);

- formální verifikace: aktivita, při níž je správnost chování softwarového produktu dokazována matematicky za použití formálních metod, což však vyžaduje formální specifikaci. Pro svou náročnost je tato aktivita využívána téměř výhradně u kritických systému, kde může nesprávná funkce ohrozit lidské životy či zdraví (O'Regan 2022, s. 292);
- testování: aktivita, při níž je softwarový produkt spouštěn za stanovených podmínek s cílem najít chyby, změřit kvalitu softwaru a vyhodnotit, zda jsou požadavky splněny (Deng 2020, s. 103). Software lze testovat na různých úrovních, prostřednictvím řady odlišných technik, manuálně nebo automatizovaně pomocí k tomu určených nástrojů (Pang 2020, s. 196). Podrobněji je testování diskutováno v podkapitole 2.2.3 níže.

Jak bylo diskutováno výše, nejnižší náklady na odstranění chyby jsou tehdy, je-li zjištěna během práce na daném produktu (specifikaci, dokumentu nebo kódu) či obecně před tím, než dojde k jeho použití. S ohledem na uvedené druhy aktivit řízení kvality je tak zřejmé, že provádění důkladných kontrol v počátečních fázích vývoje produktu, tedy revize specifikace požadavků, návrhu a případně dalších vstupních dokumentů, představuje vysoce efektivní způsob včasného odhalení významné části chyb.

I za předpokladu bezchybných požadavků a návrhu však při samotném vývoji dochází k chybám vinou pochybení programátora, jenž je možné včas odhalit a s nízkými náklady opravit prováděním časté inspekce kódu a jeho automatické analýzy.

Testování finálního produktu v jakékoli formě, automatizované nebo manuální, je pak v tomto ohledu samozřejmě nejméně výhodné.

Popsaný přístup je však v praxi omezen skutečností, že důkladné inspekce kódu nebo kvalitní nástroje pro jeho automatickou analýzou vyžadují značné náklady, které podniky často nejsou ochotny vynaložit i přesto, že obě uvedené činnosti jsou prokazatelně vysoce nákladově efektivní, snižující ve výsledku celkové náklady na kvalitu (O'Regan 2022, s. 119).

Zároveň platí, že s ohledem na efektivitu by aktivity řízení kvality měly být přiměřené s ohledem na vyvíjený produkt, kdy výukový program pro předškolní děti nemá smysl testovat stejně jako složitý bankovní systém. V praxi je tak při volbě vhodných aktivit nutné zohlednit kontext použití produktu, jeho složitost, rizika a stejně tak další faktory, jako je dostupný rozpočet a čas, kompetence členů týmu, nástroje a podobně (Stapp, Roman a Pilaeten, 2023, s. 175).

2.2.2 Činnosti zajišťování kvality

Protože zajišťování kvality je orientováno na nastavení a zlepšování veškerých procesů přímo souvisejících s výrobou produktu, pro své efektivní fungování musí být součástí všech fází jeho vývojového cyklu (Hilburn a Towhidnejad 2021, s. 52).

Primárně je zajišťování kvality realizováno prostřednictvím analýz procesů, revizí a procesních auditů, jež posuzují efektivitu procesů, jejich soulad s vnitřními a vnějšími standardy a na základě zjištění poskytují doporučení a návrhy na adekvátní zlepšení (Vallabhaneni 2021, s. 716).

Dle normy IEEE 730 (2014) lze konkrétní činnosti prováděné funkcí zajišťování kvality rozdělit do několika samostatných oblastí:

- posouzení souladu procesů životního cyklu vývoje produktu s předepsanými požadavky;
- posouzení souladu technického prostředí vývoje produktu s předepsanými požadavky;

- posouzení souladu procesů subdodavatelů s předepsanými požadavky;
- posouzení, zda měření procesů podporuje efektivní řízení procesů, je v souladu s procesy a plány projektu a odpovídá příslušným standardům a postupům;
- zhodnocení kompetencí a znalostí pracovníků s ohledem vůči požadavkům na role, které na svých pozicích v rámci vývoje produktu zastávají.

V rámci samotného posuzování jsou využívány různé podpůrné nástroje a techniky umožňující důkladnou a efektivní analýzu získaných dat a jejich kvality (Jones 2021, s. 112).

V oblasti procesů životního cyklu lze u tak například u posuzování procesů konfiguračního managementu uvažovat následující body, ověřující že:

- byla definována účinná strategie konfiguračního managementu;
- byly definovány všechny nutné role a odpovědnosti;
- byly definovány použité nástroje, techniky a postupy;
- byly definovány položky, jež jsou předmětem konfiguračního managementu;
- byl definován počáteční stav konfigurace.

Podle normy IEEE 730 (2014) výstupy z aktivit zajišťování kvality tak poskytují potvrzení, že procesy používané k vývoji, instalaci, provozu a údržbě softwaru odpovídají podmínkám smlouvy, jsou v souladu se všemi stanovenými předpisy a jsou přiměřené, účinné a efektivní, čímž umožňují produkovat software odpovídající požadavkům na kvalitu.

Ačkoli efektivní zajišťování kvality může do značné míry redukovat potřebu aktivit řízení kvality, jejich úplné nahrazení nebývá běžné. Zejména klíčové či kritické charakteristiky produktu jsou podrobovány testům nebo inspekcím, a slouží tak jak pojistka pro případ, že by zajištění kvality selhalo (Theisens 2020, s. 86).

2.2.3 Testování softwaru

Nejrozšířenější aktivitou řízení kvality softwaru zůstává i po několika desetiletích jeho testování v nejrůznějších podobách (Pang 2020, s. 216), které budou krátce diskutovány v této podkapitole.

Testování softwaru lze rozdělit podle mnoha kritérií, přičemž na nejvyšší úrovni lze rozlišit následující dvě skupiny (Chandrasekara a Pushpa 2019, s. 2):

- funkční testování: produkt je testován vůči funkčním požadavkům definovaným typicky ve funkční specifikaci;
- mimofunkční testování: je zaměřeno na hodnocení míry, do jaké produkt splňuje mimofunkční požadavky jako je spolehlivost, výkonnost nebo kompatibilita.

V závislosti na tom, jaké informace jsou pracovníkům řízení kvality k dispozici pro návrh testů, lze dále rozlišit tři následující přístupy (Stapp, Roman a Pilaeten, 2023, s. 170):

- testování černé skříňky: produkt je testován bez znalosti jeho vnitřního fungování, tedy testy jsou navrhovány čistě na základě chování definovaného ve specifikovaných požadavcích. Při provádění testu je tak produkt jakousi černou skříňkou, která na poskytnutý vstup zareaguje určitým výstupem;
- testování bílé skříňky: testy jsou navrhovány na základě znalosti vnitřního fungování produktu, tedy jeho zdrojového kódu, což umožňuje mnohem důkladnější, ale také náročnější a nákladnější testování;

- testování založené na zkušenostech: návrh testů probíhá na základě znalostí a zkušeností testera, což je vhodné jak v situacích s nedostatečnou nebo chybějící dokumentací, tak jako doplnění předchozích dvou přístupů.

Specifickou formou testování jsou testy související se změnami produktu, které jsou dvojího druhu (Baumgartner et al. 2022, s. 183):

- konfirmační testy: jsou prováděny za účelem potvrzení (konfirmace), že provedená oprava chyby byla úspěšná. Jde tak o opětovné spuštění testů, které původně selhaly důsledkem objevené chyby. Konfirmační testování se běžně označuje jako přetestování chyb;
- regresní testy: jsou prováděny s cílem potvrdit, že produkt je po provedených změnách stále plně funkční, tedy v jeho stavu nedošlo k regresi. To je velice důležité, protože každý zásah do produktu – nová funkce, úprava nebo oprava chyby – představuje riziko nežádoucích účinků, které se mohou projevit i v jeho nezměněných částech. Teoreticky by každá změna produktu vyžadovala, aby byly všechny testy spuštěny znovu, což je však v praxi většinou proveditelné pouze tehdy, jsou-li plně automatizovány (Spillner a Linz 2021, s. 88). V opačném případě musí testeři zvolit vhodnou strategii pro výběr takové množiny existujících testů, která je v daném případě nejefektivnější.

Při samotném vytváření testů je nutné zajistit, že budou navrženy tak, aby byly všechny požadavky produktu dostatečně ověřeny. Pro systematický přístup k návrhu testů lze využít řadu technik, podle nichž lze testy dále klasifikovat. Během testování přístupem černé skříňky mohou být pro návrh testů použity například následující techniky (O'Regan 2022, s. 120):

- analýza hraničních hodnot;
- rozdělení do tříd ekvivalence;
- testování přechodů stavů;
- testování scénářů;
- testování případů užití.

Jedním z klíčových podkladů pro návrh testů by měla být akceptační kritéria stanovená ve smlouvě se zákazníkem, neboť na jejich základě lze odvodit takové testy, jejichž úspěšné provedení je zásadní pro přijetí produktu (Witte 2022, s. 107).

V přístupu bílé skříňky vychází návrh testů ze struktury produktu, přičemž jednotlivé techniky jsou zaměřeny na různé úrovně pokrytí zdrojového kódu jednotkovými testy. Například může být požadováno, aby u 80 % podmínek v kódu byl otestován každý možný výsledek (Spillner a Linz 2021, s. 119).

U testování založeného na zkušenostech je klíčovou technikou exploratorní testování, při němž je daný produkt zkoumán intuitivně, bez předem definovaných testů (Chandrasekara a Pushpa 2019, s. 3). Nejde však o náhodné a nestrukturované testování, protože při jeho provádění testeři dynamicky určují cíl svého zkoumání, současně navrhují a provádí testy, přičemž na základě zjištěných skutečností se sami rozhodují o dalším postupu (O'Regan 2022, s.127). Exploratorní testování je velmi efektivní a důležitou složkou testování, především pokud je použito v kombinaci s ostatními přístupy a technikami (Amey 2022, s. 5). Určitou nevýhodou exploratorního testování je skutečnost, že kromě případných nalezených chyb neposkytuje žádnou dokumentaci o zkoumaných oblastech produktu a rozsahu provedených testů. Tento nedostatek je možné zmírnit použitím metod, které umožňují strukturovaný přístup k exploratornímu testování s minimální dokumentací, jejichž nejznámějším příkladem je testování v relacích (Spillner a Linz 2021, s. 130).

Dále je možné klasifikovat testy podle toho, s jakým cílem jsou navrhovány. V tomto smyslu pak lze rozlišovat následující dvě kategorie (Tockey 2019, s. 339):

- pozitivní testy: ověřují, zda se produkt chová podle specifikovaných požadavků, tedy na specifický vstup reaguje daným výstupem;
- negativní testy: ověřují, jak produkt reaguje na neočekávané nebo nevhodné vstupy, pro které není chování specifikováno.

Negativní testy jsou mnohdy opomíjeny, ovšem právě ony zásadně přispívají k důvěře v robustnost produktu (Baumgartner et al. 2022, s. 217). Negativní testy mají vyšší šanci odhalit chyby, proto někteří autoři doporučují, aby pro každý pozitivní test existoval alespoň jeden test negativní (Amey 2022, s. 114). To lze demonstrovat na jednoduchém příkladu – jestliže má políčko „částka“ přijímat hodnotu 1 až 999, pozitivní test bude pracovat s jakoukoli číselnou hodnotou, zatímco v negativním testu bude zadán text, mezera nebo speciální znak.

2.3 Přístupy ke kvalitě dle modelů životního cyklu

Životní cyklus vývoje softwarového produktu sestává z aktivit a fází, kterými produkt postupně prochází – od prvotního konceptu a specifikace požadavků, návrhu, samotného vývoje a testování až po nasazení do provozu a konečně jeho vyřazení. Model životního cyklu vývoje softwarového produktu (dále jen model životního cyklu) je pak obecným, vysokoúrovňovým znázorněním těchto aktivit, fází a vztahů mezi nimi (Hilburn a Towhidnejad 2021, s. 42).

V praxi umožňují modely životního cyklu přistupovat k vývoji softwarových produktů systematicky a disciplinovaně, což je obzvlášť důležité při práci v týmu (Rajib 2018, s.37).

Vzhledem k rozmanitosti softwarových produktů a specifik projektů, v jejichž rámci jsou dodávány, neexistuje žádný jediný, univerzálně platný a použitelný model životního cyklu. Z řady existujících modelů je tak nutné vybrat takový, který je vhodný s ohledem na typ, velikost, složitost a rizikovost vyvíjeného produktu, velikost vývojového týmu, omezení projektu daná časem, prioritami zadavatele, rozpočtem a zdroji a řadu dalších faktorů (Tockey 2019, s. 141).

Protože odlišnosti mezi jednotlivými modely životního cyklu se promítají i do přístupu k zajišťování a řízení kvality, volba správného z nich je nutná pro maximalizaci šance, že výsledný produkt bude kvalitní (Kohen, Kapoor a Bhatia 2021, s. 363).

Pro trvalý vliv na kvalitu procesů a tím i kvalitu produktu však není dostačující model životního cyklu pouze zvolit a aplikovat prostřednictvím zavedení do praxe, ale je nutné jej také dostatečně zdokumentovat. To zajišťuje nejen konzistenci procesů a jejich vztahů mezi různými projekty v organizaci, ale také prokazuje soulad s požadavky norem jako ISO 9000 nebo rámce SEI CMM (Rajib 2018, s. 39).

Na nejvyšší úrovni je modely životního cyklu možné rozdělit do dvou hlavních kategorií, kterými jsou (Kama, Basri a Mahmood 2020, s. 11):

- tradiční modely, kam patří například vodopádový model, v-model nebo spirálový model (Adlakha-Hutcheon a Masys 2022, s. 96);
- modely založené na agilních metodikách, jež představuje například Scrum, extrémní programování nebo agile unified process (Kama, Basri a Mahmood 2020, s. 18).

Tradiční modely životního cyklu jsou orientované na realizaci plánu, jež je připraven v počátečních fázích projektu. To vyžaduje značné úsilí, neboť požadavky na produkt či službu musí být kompletní a maximálně detailní, aby mohly být zahájeny následné, po sobě jdoucí

fáze návrhu, vývoje, testování a provozu (Kama, Basri a Mahmood 2020, s. 17). Klasickým příkladem je vodopádový model, jež bude blíže diskutován v podkapitole níže.

U modelů založených na agilních metodikách je na místo orientace na plán kladen důraz naopak na skutečnost, že v požadavcích mnohdy dochází k častým změnám i v průběhu samotného vývoje produktu (Adlakha-Hutcheon a Masys 2022, s. 97). Namísto toho, aby byl celý produkt doručen zákazníkovi na konci vývoje, agilní modely životního cyklu doporučují vytvářet jej postupně v kratších vývojových cyklech. Na konci každého takového cyklu je vždy funkční dílčí produkt, obsahující přírůstky funkčnosti, který je doručen zákazníkovi za účelem poskytnutí zpětné vazby (Adlakha-Hutcheon a Masys 2022, s. 97).

Uvedený přístup ve své podstatě představuje inkrementální a iterativní vývoj, což jsou dvě základní charakteristiky, jež jsou v určité podobě přítomné ve všech agilních modelech (Spillner a Linz 2021, s. 55). Oba zmíněné pojmy a jejich kombinaci lze vysvětlit následovně:

- inkrementální vývoj znamená, že produkt je vyvíjen postupně a v kratších cyklech, jejichž výstupem je potenciálně použitelný produkt rozšířený o další funkce. Oproti produktu z předchozí fáze je tedy vždy patrný dokončený přírůstek – inkrement – funkčnosti. Právě to je klíčovým aspektem tohoto přístupu, neboť v případě tradičního modelu mohou být sice rozpracovány všechny funkce zároveň, ovšem produkt jako celek je tím pádem nepoužitelný, dokud nejsou dokončeny. Oproti tomu inkrementální vývoj umožňuje minimalizovat čas nutný k uvedení produktu na trh, kdy je možné uvolnit nejprve verzi poskytující zákazníkům základní funkce a následně produkt rozšiřovat podle jejich přání a zpětné vazby (Spillner a Linz 2021, s. 52). V praxi si tak lze představit například situaci, kdy v období pandemie spustí firma službu umožňující uživatelům trasovat svůj pohyb, čímž rychle získá uživatelskou základnu. Teprve pak přidává další plánované funkce a rozvíjí aplikaci podle toho, jak se vyvíjí skutečná situace. Inkrementální procesní modely jsou používány již od 50.let minulého století, avšak teprve v období 90.let začaly být populární v oblasti softwarového vývoje (Adlakha-Hutcheon a Masys 2022, s. 98);
- iterativní vývoj znamená, že při vývoji produktu je postupováno v cyklech, během nichž dochází k opakování – iteraci – jednotlivých vývojových procesů, typicky návrhu, programování, testování a dokumentaci (Richards 2021, s. 38). V každém cyklu, iteraci, tak dochází k postupné úpravě produktu nebo jeho částí, a tedy jeho zlepšování, což je hlavním smyslem tohoto přístupu (Spillner a Linz 2021, s. 52). Jako příklad lze uvést vývoj produktu, kdy na konci první iterace jsou dostupné všechny funkce, ovšem fungují jen velmi omezeně a v minimálním uživatelském rozhraní. Po skončení druhé iterace poskytují všechny funkce použitelné výsledky, ovšem v neupravené podobě, a uživatelské prostředí již nabízí lepší rozložení. Ve třetí iteraci jsou funkce i uživatelské prostředí dokončeny;
- iterativní a inkrementální vývoj je pak kombinací obou výše uvedených přístupů, který lze demonstrovat na jednoduchém příkladu produktu, který má obsahovat 3 funkce. V čistě inkrementálním přístupu je v prvním cyklu doručena jedna kompletní funkce ve 100% kvalitě. V čistě iterativním přístupu budou v prvním cyklu doručeny všechny tři funkce, ovšem dokončené jen z 30 %. V iterativním a inkrementálním přístupu budou v prvním cyklu doručeny funkce (jedna či více), které sice nebudou zcela dokončené, ovšem budou již pro uživatele použitelné – inkrement. V dalších iteracích pak dojde k jejich dokončení nebo vylepšení, přičemž na výstupu je vždy použitelná funkce (Stephens 2022, s. 286).

Zpětná vazba od zákazníka a uspokojení jeho potřeb, i kdyby požadované změny znamenaly rozpor s původně očekávanými nebo již realizovanými požadavky, je srdcem filozofie agilních přístupů a jejich flexibility (Kama, Basri a Mahmood 2020, s. 22).

Tuto orientaci na kvalitu prostřednictvím poskytované hodnoty popisuje i první ze 12 základních principů agilního vývoje, které v roce 2001 zveřejnila skupina odborníků v takzvaném manifestu agilního vývoje softwaru (Layton, Ostermiller, Kynaston 2020, s. 13). Těmito principy jsou:

1. nejvyšší prioritou je uspokojit zákazníka včasným a průběžným dodáváním hodnotného (pro zákazníka užitečného) softwaru;
2. změna požadavků je vítána, a to i v pozdní fázi vývoje. Agilní procesy využívají změny pro získání konkurenční výhody zákazníka;
3. dodávat funkční software často, od několika týdnů do několika měsíců, přičemž se upřednostňují kratší lhůty;
4. obchodníci a vývojáři musí v průběhu projektu denně spolupracovat;
5. projekty jsou tvořeny motivovanými jednotlivci, kterým je poskytováno potřebné prostředí, podpora a důvěra ve splnění cíle;
6. nejúčinnější a nejefektivnější metodou předávání informací vývojovému týmu a v rámci něj je osobní rozhovor;
7. hlavním ukazatelem postupu prací je funkční software;
8. agilní procesy podporují udržitelný rozvoj. Sponzoři, vývojáři a uživatelé by měli být schopni udržet konstantní tempo po neomezenou dobu;
9. trvalá pozornost věnovaná technické dokonalosti a dobrému návrhu zvyšuje agilitu;
10. jednoduchost – umění maximalizovat množství nevykonané práce – je zásadní;
11. nejlepší architektury, požadavky a návrhy vznikají v samo se organizujících se týmech;
12. v pravidelných intervalech se tým zamýšlí nad tím, jak se stát efektivnějším, a podle toho upravuje své chování.

Jedním z nejznámějších a nejvýznamnějších agilních modelů životního cyklu je rámec Scrum (Adlakha-Hutcheon a Masys 2022, s. 98), který bude blíže představen v příslušné podkapitole níže.

Jak bylo již zmíněno, ani jeden z modelů není vhodný pro všechny typy projektů, ovšem některé jejich dílčí prvky, praktiky, mohou být použity samy o sobě. To v praxi znamená, že organizace nezřídka přistoupí k vytvoření vlastního, efektivního modelu životního cyklu, založeného na takové kombinaci prvků z tradičních a agilních modelů, která nejlépe odpovídá jejím potřebám (Tockey 2019, s. 776).

2.3.1 Tradiční model

Vodopádový nebo kaskádový model životního cyklu je považován za nejstarší a nejčastěji používaný strukturovaný tradiční model ve vývoji softwaru (Kama, Basri a Mahmood 2020, s. 18).

Ve své podstatě je vodopádový model sekvenční a lineární, sestávající z několika jasně definovaných, po sobě jdoucích fází: shromáždění a analýza požadavků, návrh budoucího řešení, vývoj, testování, provoz a údržba (Jun et al. 2021, s. 58).

Protože je každá fáze přímo závislá na výstupu z fáze předcházející, musí být vždy prováděny v daném pořadí a jedna po druhé, nikoli souběžně. Například dokud není zcela dokončena fáze shromáždění a analýzy požadavků, jejíž výstupem je úplná funkční specifikace nebo dokument se stejným významem, nemohou být zahájeny aktivity ve fázi návrhu.

Tato skutečnost v důsledku znamená, že požadavek na provedení jakékoli změny v již dokončené fázi představuje značné náklady spojené s nutným přepracováním všech odvozených meziproduktů v předcházejících fázích, proto nelze tento model efektivně využít v situacích, kdy je třeba flexibilita, například pokud se požadavky mohou změnit (Jun et al. 2021, s. 58).

Tento model je mnohdy kritizován a označován jako překonaný a nevyhovující (Khachatryan 2020, s. 202), což ovšem není zcela přesné – u projektů, kde je nutná důkladná příprava a analýza požadavků a jakákoli změna je nežádoucí (typicky software pro kritické systémy nebo systémy podléhající regulatorním požadavkům), je vodopádový model naopak stále velmi dobře použitelný a v praxi skutečně používaný, přinášející zejména následující výhody (Tockey 2019, s. 772):

- vysoce efektivní, dobře naplánovaný návrh, který od počátku zohledňuje všechny požadavky zadavatele;
- minimální náklady na přepracování, neboť návrh je založen na kompletních, detailně zpracovaných požadavcích;
- jasně definované vstupy a výstupy pro každou z jednotlivých fází, včetně případných kritérií;
- detailní dokumentace vytvořená během počáteční analýzy požadavků a návrhu technického řešení umožňuje později velice efektivní údržbu produktu, jeho rozšíření a také usnadňuje sdílení znalostí;
- od okamžiku schválení požadavků až do akceptačního testování není nutné zapojení zadavatele;
- na základě detailní specifikace a návrhu mohou být s předstihem vytvořeny a případně revidovány všechny potřebné testy, jež budou následně provedeny na dokončeném produktu ve fázi testování;
- škálovatelnost a jednoduchost vodopádového modelu umožňuje jeho aplikaci v týmech bez ohledu na jejich velikost;
- jasně vymezené milníky projektu.

Je tedy zřejmé, že vodopádový model lze efektivně využít tehdy, jsou-li v dostatečném předstihu před započetím prací všechny požadavky kompletně známé, stabilní a dostatečně kvalitně zpracované.

Kromě citlivosti na změny požadavků je třeba zmínit i vyšší technické riziko spojené s případným nevhodným technickým návrhem, který může vést k potřebě rozsáhlého přepracování.

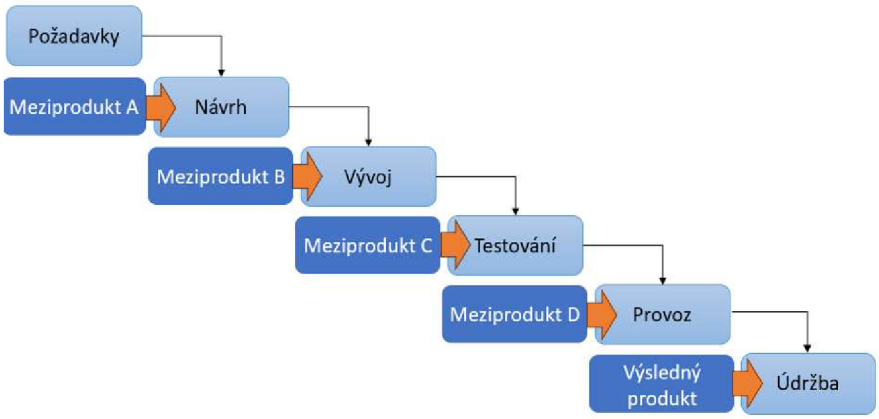
Další nevýhodou je skutečnost, že v případě dlouhotrvajících fází může být pro projektového manažera obtížné monitorovat aktuální stav projektu a adekvátně jej tak řídit (Tockey 2019, s. 773).

S ohledem na kvalitu softwaru je ve vodopádovém modelu patrný zřejmý nedostatek – testování je prováděno až poté, co je úplně skončena fáze vývoje. Ani bezchybné zpracovaní požadavků totiž nemůže zaručit, že nedojde k jejich chybné interpretaci anebo chybě při vlastní implementaci, které tak budou odhaleny až na dokončeném produktu, což s sebou nese vysoké náklady spojené s opravou.

Tento nedostatek byl později řešen navržením modifikované verze vodopádového modelu, v níž byla pro každou fázi vývoje plánována adekvátní aktivita testování, a z níž se nakonec vyvinul samostatný model životního cyklu, jež byl pro svůj tvar nazván V-model (Jun et al. 2021, s. 59).

Na obrázku 1 níže je znázorněn klasický vodopádový model životního cyklu se šesti po sobě následujícími fázemi:

Obrázek 1 Vodopádový model představující 6 fází vývoje a jejich vzájemnou závislost vycházející ze skutečnosti, že vstupem do jedné fáze je vždy výstup z fáze předcházející (naznačeno oranžovou šipkou)



Zdroj: vlastní zpracování

Vstupy a výstupy naznačené u jednotlivých fází na obrázku představují příslušné meziprodukty, kterými jsou typicky specifikace požadavků (A), návrh technického řešení (B), zdrojový kód (C) a funkční produkt připravený k otestování (D).

2.3.2 Agilní model

Za nejrozšířenější variantu agilního modelu životního cyklu je v současnosti považován Scrum, který jeho autoři definují jako „*jednoduchý rámec pro lidi, týmy a organizace, který umožňuje vytvářet hodnoty prostřednictvím adaptivních řešení složitých problémů*“ (Tockey 2019, s. 20). Zároveň jde o empirický přístup, který přijímá skutečnost, že řešený problém (vývoj produktu odpovídající požadavkům zadavatele) nemusí být možné dokonale pochopit a definovat, a proto se zaměřuje na maximalizaci schopností vývojového týmu doručovat často a pružně reagovat na změny požadavků, jakkoli pozdě se objeví (Hilburn a Towhidnejad 2021, s. 114).

Rámec Scrum vychází ze tří základních pilířů, které jsou důležité pro pracovní prostředí podporující týmovou spolupráci (Heath 2021, s. 20):

- transparence ve své podstatě znamená, že klíčové aktivity, způsob práce, vstupy, produkt i výstupy jsou otevřeny všem zúčastněným stranám, což umožňuje jejich následnou inspekci a případnou adaptaci;
- inspekce představuje důkladnou avšak nikoli formální kontrolu, na níž se podílí každý člen Scrum týmu a jež se může týkat produktu, procesů, personálních aspektů, postupů a průběžného zlepšování;
- adaptace je přirozeným krokem následujícím po inspekci, kdy jsou případná zjištění analyzována a v rámci průběžného zlepšování jsou přijata opatření k jejich nápravě. Adaptace je tedy vyjádřením jedné z klíčových hodnot agilních modelů – schopnosti a ochotě přizpůsobovat se změnám (Heath 2021, s. 21). Změna se může týkat technického aspektu produktu, používaných nástrojů, podpůrných procesů nebo třeba týmové spolupráce.

V souladu s dříve zmíněnými principy agilního vývoje je Scrum prostředkem, který týmům umožňuje optimalizovat své fungování prostřednictvím samoorganizace a porozumění vyvíjenému produktu.

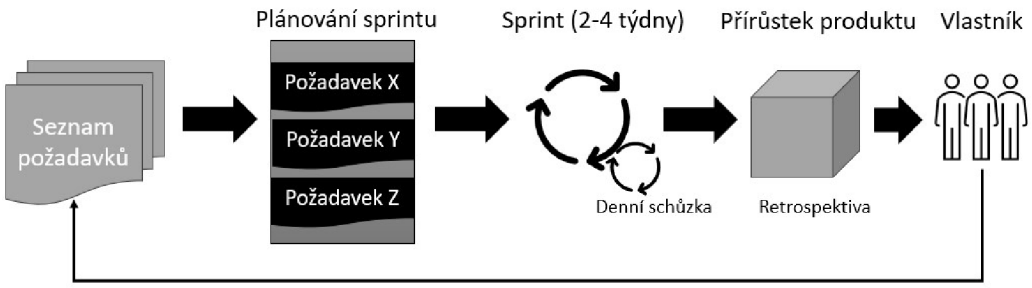
Základní prvky charakterizující fungování rámce Scrum lze shrnout do následujících bodů (Ransome a Schoenfield 2021, s. 30):

- vytvoření a udržování seznamu všech funkcí, vylepšení a oprav, které má produkt obsahovat (product backlog), a za který je zodpovědný jeho vlastník (product owner), který řazením jednotlivých položek určuje jejich relativní důležitost;
- jednotlivé položky seznamu jsou navrhovány tak, aby realizace každé z nich sama o sobě přispěla k celkové hodnotě poskytované produktem;
- u každé položky je jasně definováno, jaké podmínky musí být splněny, aby mohla být považována za splněnou. Tato forma výstupních kritérií je v rámci Scrum označována jako Definition of Done;
- vytyčení cílů, ve smyslu inkrementů produktu, kterých má být dosaženo během řady časově jasně ohraničených iterací, které jsou ve Scrumu označovány jako sprinty a typicky trvají mezi 2 až 4 týdny. Každý sprint začíná plánovací schůzkou, během které si tým – s přihlédnutím k prioritě dané pořadím – vybere položky (nové funkce, opravy a podobně) tak, aby je dokázal dokončit v požadované kvalitě do konce iterace. Toto rozhodnutí je výhradní pravomocí týmu;
- každý den se tým schází ke krátké schůzce za účelem inspekce postupu prací vůči vytyčeným cílům, což usnadňuje komunikaci, identifikaci překážek a podporuje včasné rozhodování;
- na konci každého sprintu je k dispozici funkční přírůstek produktu, který bude předveden vlastníkovi pro získání zpětné vazby (inspekce produktu) a zároveň se tým jako celek schází a hodnotí, jak sprint probíhal (inspekce procesu – retrospektiva).
- týmy se organizují samy, bez potřeby zapojení manažera, a sestávají z pracovníků se všemi nutnými dovednostmi;
- tým je podporován rolí Scrum Mastera, který pomáhá odstraňovat procesní překážky v týmu i mimo něj, moderuje týmové schůzky a zajišťuje, aby byly agilní principy a metody aplikovány efektivně.

Jednou z široce používaných, nikoli však povinných praktik je nastavení vstupních kritérií, jež jsou označována jako Definition of Ready (Hundhausen a Schwaber 2021, s. 51). Analogicky s výstupními kritérii jde o soubor podmínek, které musí položka v seznamu funkcí splňovat před tím, než ji tým může přiřadit do sprintu a pracovat na ni. V praxi to znamená, že tým nezačíná pracovat na něčem, co není zcela připravené, což předchází neefektivnímu využívání zdrojů. Příkladem Definition of Ready pro konkrétní položku je požadavek na existenci jasně definovaných akceptačních kritérií, odhadu pracnosti a proveditelnosti v rámci jednoho sprintu (Hundhausen a Schwaber 2021, s. 51).

Zjednodušené schéma rámce Scrum, včetně jeho klíčových aktivit popsaných výše, je vyobrazeno na obrázku 2 níže:

Obrázek 2 Model rámce Scrum



Zdroj: vlastní zpracování, upraveno z Layton 2018, s. 10

Rámec Scrum podporuje kvalitu již diskutovaným důrazem na principy inspekce, transparence a adaptace, přičemž v praxi jde o následující:

- definice výstupních kritérií (Definition of Done) umožňuje nastavit požadovanou úroveň kvality prostřednictvím podmínek, jež musí být splněny, a bez nichž není možné inkrement akceptovat;
- testování je nedílnou součástí vývoje (nikoli samostatnou fází), nutnou pro dokončení funkčního inkrementu. V rámci každého sprintu tak testování zároveň poskytuje rychlou zpětnou vazbu usnadňující opravu chyb;
- zodpovědnost vlastníka produktu za požadavky v seznamu položek umožňuje jejich dostatečně detailní zpracování i včasné úpravy, zajišťující soulad s jeho skutečnými potřebami;
- inspekce průběhu sprintu (retrospektiva) i inspekce přírůstku produktu, jež jsou prováděny po skončení sprintu, představují rychlý a efektivní prostředek zpětné vazby, který umožňuje rychle identifikovat procesní i produktové problémy.

Nevýhoda rámce Scrum spočívá v tom, že postupný vývoj jednotlivých požadavků může být problematický s ohledem na kvalitu kódu (upřednostnění rychlých avšak nepříliš vhodných řešení, která mohou později působit problémy) a celkový návrh produktu (Walsh, Sathiadev a Trumbach 2021, s. 117). Zároveň je třeba uvést, že efektivní využití rámce Scrum je do jisté míry omezeno velikostí týmu, který by neměl přesáhnout 10 členů (Layton 2018, s. 68).

V prostředí agilního vývoje softwaru vznikla (nebo začala být široce používána) řada praktik a technik, které lze aplikovat bez ohledu na použitý model životního cyklu a jež mají přispívat k vyšší kvalitě výsledného produktu. Mezi ně jsou řazeny následující (Layton, Ostermiller, Kynaston 2020 s. 338)

- párové programování: technika, kdy dva programátoři pracují současně na jednom počítači – jeden píše kód a druhý jej okamžitě reviduje. Kromě efektivnějšího řešení problémů umožňuje tento přístup také sdílení znalostí. V praxi však není přínos párového programování jednoznačný, a přestože někteří autoři poukazují na nižší chybovost produktu (Layton 2018, s. 291), jiní naopak poukazují na mírné zvýšení kvality za neúměrně vysokých nákladů (Jones 2021, s. 50);
- vzájemná revize kódu: je formou revize kódu, kdy si členové vývojového týmu navzájem revidují kód za účelem jeho zlepšení nebo nalezení chyb. Tato praktika je považována za efektivní, a v týmu se zkušenými programátory by měla vést ke zvýšení kvality produktu (Layton, Ostermiller, Kynaston 2020, s. 339);
- vývoj řízený testováním: je technika programování, kdy jsou před samotným kódem nové funkce vytvořeny automatické testy, které ji ověřují. Tyto testy samozřejmě nejprve selhávají, přičemž programátor následně napíše co nejjednodušší funkční kód tak, aby mohly testy úspěšně projít. Následně kód upravuje, vylepšuje a opakovaným spouštěním testů ověřuje, že jeho funkčnost zůstala nezměněna (Layton, Ostermiller, Kynaston 2020, s. 106). Tento přístup nutí programátory psát testovatelný kód a zároveň ověřuje testovatelnost požadavků (O'Regan 2022, s. 230). Tato technika tak ověřuje na nejnižší úrovni, že programátor vytvořil kód dané funkce v souladu s tím, jak interpretoval specifikované požadavky a jak vhodně zvolil testované podmínky, což však nutně nemusí být správně. Z toho důvodu je pro efektivní využití této techniky nezbytné, aby testy pracovaly s akceptačními kritérii samotného požadavku (Tockey 2019, s. 21).

2.4 Metody práce

Tato bakalářská práce je na nejvyšší úrovni členěna na dvě části, a to na část teoreticko-metodologickou a část analyticko-praktickou.

Při zpracování teoreticko-metodologické části byly využity metody literární rešerše, analýzy dokumentů a srovnávání. Za relevantní pro literární rešerši byly považovány takové zdroje, jež jsou dostupné v českém nebo anglickém jazyce a jejichž stáří nepřesahuje 5 let (rok vydání 2018 a později), což je v dynamické oblasti informačních technologií důležité kritérium. Výjimku představovaly technické normy, kde bylo v případě citace použito aktuálně platné revize.

Metodou srovnávání byly posuzovány rozdíly mezi různými přístupy k zajišťování a řízení kvality softwaru a doporučenými technikami a činnostmi pro jejich zavedení.

Analyticko-praktická část práce se zabývá rozborem vývojového cyklu softwaru u vybrané společnosti s cílem navrhnout zlepšení stávajících aktivit zajišťování a řízení kvality, jež by v důsledku vedla ke snížení nákladů souvisejících s nízkou kvalitou dodávaných softwarových řešení.

Vzhledem ke skutečnosti, že zpracovávané informace i kontext jejich prezentace poskytuje detailní vhled do fungování společnosti a jejího obchodního modelu, její skutečný název nebude uveden. Jedná se o nadnárodní podnik působící ve více než 50 zemích světa a zabývající se činností převážně v oblasti informačních technologií – hardwaru i softwaru – a poskytující mimo jiné služby v rámci kritické infrastruktury. Pro účely této bakalářské práce bude společnost nadále označována jako ABC, přičemž je uvažována pouze ta její část, která se zabývá vývojem softwarových řešení pro zákazníky.

Zpracování této bakalářské práce bylo oznámeno vedoucímu globální projektové kanceláře společnosti, který zároveň poskytl součinnost v přístupu k potřebným dokumentům a finančním údajům projektů, podmíněnou neuvedením názvu společnosti, jejích klientů, produktů ani konkrétních oborů činnosti.

Samotné zpracování analyticko-praktické části probíhalo v následujících krocích, jež tak zároveň představují pracovní postup autorky:

1. zjištění stávajícího stavu;
    a. rozbor životního cyklu softwarového řešení ve společnosti;
    b. shromáždění informací o nákladech spojených s nízkou kvalitou;
    c. popis aktivit zajišťování a řízení kvality;
2. návrh na zlepšení na základě získaných zjištění;
    a. identifikace nedostatků v aktivitách zajišťování a řízení kvality;
    b. návrh adekvátních zlepšení a odhad nákladů;
    c. posouzení aplikovatelnosti a přínosu pro projekt.

V této části práce bylo nejprve využito metody syntézy, kdy byly na základě zjištěných informací formulovány závěry týkající se popisu modelu a jeho charakteristik v kontextu aktivit souvisejících s kvalitou softwaru. Za použití metody dedukce pak byla z obecných doporučení a principů formulována konkrétní navržená zlepšení.

Během zjišťování stávajícího stavu byly informace získávány z několika zdrojů:

- firemní systémy (SAP PPM pro finanční údaje projektů, Atlassian Jira pro řízení dodávek na úrovni úkolů a hlášení chyb a Atlassian Confluence pro správu dokumentů a týmovou spolupráci);

- interní dokumenty a směrnice týkající se řízení projektů a vývoje softwarových řešení, spravovaných globální projektovou kanceláří;
- materiály sloužící k prokázání souladu procesů společnosti s požadavky normy ISO 9001;
- projektová dokumentace referenčních projektů.

V prvním kroku byl zkoumán životní cyklus softwarového řešení v organizaci, včetně technických prvků a nástrojů. Byly popsány jeho hlavní principy, struktura, jednotlivé fáze, podfáze i jejich členění a obsah.

Ve druhém kroku byly shromážděny informace o nákladech spojených s nízkou kvalitou, které byly získány na vzorku tří referenčních, již dokončených projektů. Ty byly zvoleny vedoucím projektové kanceláře tak, aby jejich velikost (rozsah) odpovídala co nejvíce průměrné velikosti projektu v organizaci, a to 550 člověkodnů, kam však společnost nezapočítává aktivity projektového řízení – čas na nich strávený není zaznamenáván v systému Jira. K těmto projektům byly k dispozici veškeré informace prostřednictvím udělení přístupu do systémů Jira a Confluence.

Pro každý z těchto projektů byl sestaven seznam validních chyb, za které byly považovány všechny chyby, jež nebyly ve firemním systému uzavřeny jako „Duplikát“ nebo „Není chyba“.

Mezi náklady spojené s nízkou kvalitou byly řazeny náklady na přepracování, opětovné testy a podpůrné činnosti spojené s opravou chyb, jež bylo možné odvodit z dostupných dat – počtu hodin zaznamenaných u jednotlivých úkolů a průměrného nákladu na hodinu práce.

U chyb bylo rozlišováno, zda jde o chyby nalezené během vývoje řešení anebo nahlášené zákazníkem ve fázi akceptace, pilotního provozu nebo produkce. U druhé skupiny je vyčíslení dodatečných nákladů komplikované, proto byla přičítána paušálně částka rovnající se nákladům na dva člověkodny práce, což bylo doporučeno vedoucím projektové kanceláře.

Následně bylo pro každý z referenčních projektů vyjádřeno, jaký podíl na celkových nákladech projektu tvoří náklady na opravy a jaká je jejich struktura.

Dále byly u chyb identifikovány nejčastější skupiny jejich příčin, což bylo nutné pro navržení efektivního způsobu, jak jim předcházet. Informace o kořenových příčinách byly součástí hlášení o chybách zaznamenaných v nástroji Jira. U chyb nalezených zákazníkem byly zkoumány důvody, proč nebyly odhaleny během interního testování, přičemž zjištěné informace sloužily jako podklad pro návrh vhodných zlepšení.

V dalším kroku byly v každé fázi životního cyklu softwarového řešení identifikovány stávající aktivity zajišťování a řízení kvality, na nichž byly následně popsány zjištěné nedostatky ve vztahu k doporučené praxi a principům definovaným v teoretické části této práce.

Na výstupu z této činnosti, tedy seznamu relevantních aktivit a popsaných nedostatků, bylo následně provedeno navržení vhodných protiopatření – zlepšení, vycházející z ošetření kořenových příčin chyb, příčin úniků chyb a obecných principů ekonomického aspekty kvality softwaru.

U každého navrženého zlepšení byla uvedena odhadovaná pracnost a tedy náklady nutné k jeho zavedení, přičemž tato hodnota byla konzultována s vedoucím projektové kanceláře a zástupcem vývoje řešení.

Maximální výše nákladů, které mohou být na zavedení navržených zlepšení vynaloženy, nazvaná hranice přijatelnosti, byla stanovena vedoucím projektové kanceláře na 11 % z původně plánovaných nákladů projektu.

Následně byla z navržených opatření vybrána ta, jejichž zavedení za náklady nižší nebo rovné hranici přijatelnosti mělo největší potenciál řešit kořenové příčiny chyb a jejich úniků.

V posledním kroku byly účinnost a návratnost investice do výsledné množiny zlepšení posouzeny na jednotlivých referenčních projektech. Za tímto účelem byla navržená zlepšení posouzena vedoucím projektové kanceláře a zástupci vývoje řešení, přičemž bylo odhadnuto, že v dané rozsahu lze očekávat redukci celkového počtu chyb přibližně o 2/3.

Na základě tohoto odhadu byly pro každý z referenčních projektů vypočítány náklady na opravy redukovaného počtu chyb a hodnoceno, zda investice do zlepšení v daných případech vede k přínosu nebo ke ztrátě.

Pro pomocné výpočty a tabulky uvedené v této práci byl použit program Microsoft Excel 365, diagramy pak byly vytvořeny v programu Microsoft Powerpoint 365.

3 Analyticko-praktická část práce

3.1 Životní cyklus softwarové dodávky ve společnosti ABC

Protože společnost ABC poskytuje své služby prostřednictvím regionálních poboček po celém světě, její dodávky zahrnují široké spektrum projektů nesourodé velikosti, složitosti a specifik, na kterých pracují mnohdy multikulturní týmy s odlišnými praktickými zkušenostmi a znalostmi, ale i přístupem k práci nebo k řešení problémů, což je navíc komplikováno rozdílnými časovými zónami a jazykovou bariérou.

Aby bylo zajištěno, že bez ohledu na výše uvedené aspekty budou všechny projekty napříč různými zeměmi a pro různé zákazníky dodávány systematicky, konzistentně a v určitém společném standardu, definovala společnost svůj vlastní, globálně platný rámec popisující model životního cyklu dodávky softwarového řešení, který interně nazývá Delivery Framework (DFW).

Kromě výše zmíněného je jeho cílem zajistit společný procesní jazyk všem zúčastněným stranám, poskytovat popis nejlepší praxe pro jednotlivé aktivity napříč celým životním cyklem dodávky a samozřejmě také poskytovat prostředek pro kontrolu a měření souladu skutečné projektové praxe s doporučenými procesy.

Za účelem zajištění co nejvyšší přístupnosti a jednoduchosti je celý rámec DFW prezentován ve formě moderně navržených, minimalistických webových stránek na intranetu podniku, což umožňuje efektivně využívat propojení mezi jednotlivými prvky, rychlou orientaci i vyhledávání.

Samotný obsah DFW byl dle sdělení zástupce projektové kanceláře koncipován tak, aby neobsahoval zbytečné teoretické informace, ale poskytoval prakticky aplikovatelné postupy a doporučení.

Koncept DFW vychází z následujících čtyř principů, jež jsou promítnuty napříč všemi jeho částmi:

- krátké cykly zpětné vazby, ve kterých jsou meziprodukty jednotlivých fází, podfází nebo i samotný částečně dokončený produkt poskytovány k průběžné kontrole či testování v rámci projektového týmu;
- krátké doručovací cykly, kdy je software vyvíjen po menších a samostatně funkčních částech, nikoli celý současně;
- podpora integrovaného týmu, kde testování je součástí vývoje a nikoli oddělenou činností;
- důraz na trvalé zlepšování prostřednictvím zlepšování procesů a praktik na základě poskytované zpětné vazby.

Protože převážná část projektů společnosti je realizována v předem definovaném rozsahu a za pevně stanovenou cenu, dodávky jsou z vnějšího pohledu typicky vodopádové: zákazník poskytne zadání požadovaného řešení a při dalším kontaktu očekává kompletní, plně funkční produkt. Výše uvedené principy, vycházející z prvků agilního vývoje, tak lze aplikovat pouze v omezené míře, a to pouze v rámci interních vývojových praktik, procesů a jejich dílčího zlepšování.

Vzhledem k povaze dodávaných řešení a konzervativnímu přístupu zákazníků je jejich aktivní participace během vývoje spíše výjimkou, což využití principů agilního vývoje na úrovni projektu jako celku v zásadě znemožňuje.

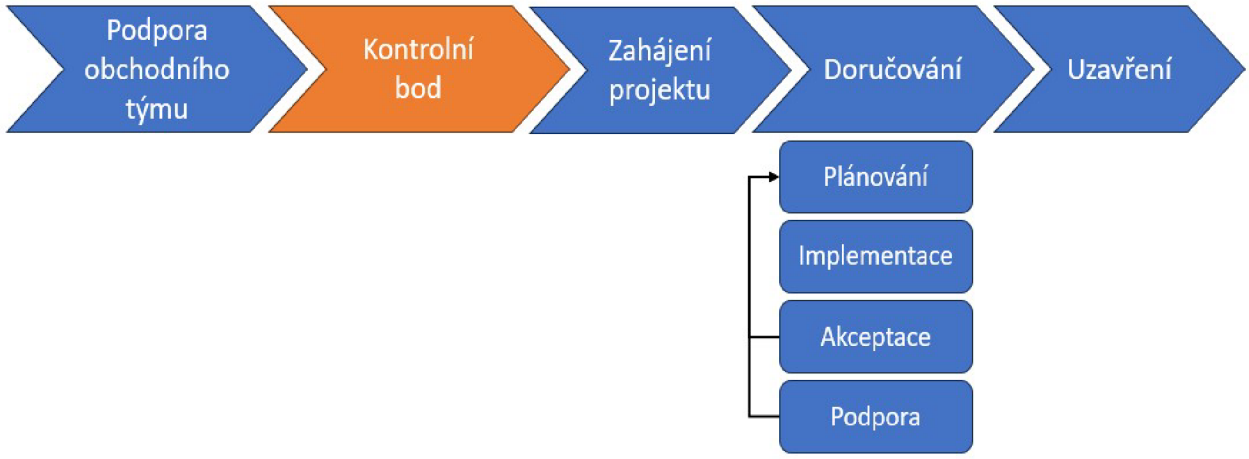
Služby jsou tak ve výsledku dodávány prostřednictvím hybridního modelu projektového řízení, kombinujícího klasický vodopádový přístup s některými agilními principy a praktikami, které jsou využívány pouze vývojovými týmy uvnitř společnosti a mezi jejími jednotlivými odděleními.

Dle struktury DFW lze životní cyklus softwarové dodávky, realizované jako projekt, rozdělit do následujících čtyř fází a jednoho kontrolního bodu, které budou diskutovány v jednotlivých sekcích níže:

1. podpora obchodního týmu před uzavřením zakázky;
2. kontrolní bod – rozhodnutí o zahájení;
3. zahájení projektu;
4. doručování
    a. plánování;
    b. implementace;
    c. akceptace;
    d. podpora;
5. uzavření.

Schéma jednotlivých fází a podfází rámce DFW je zobrazeno formou diagramu na obrázku 3 níže:

Obrázek 3 Fáze modelu životního cyklu vývoje softwarového řešení navržené podnikovým rámcem DFW



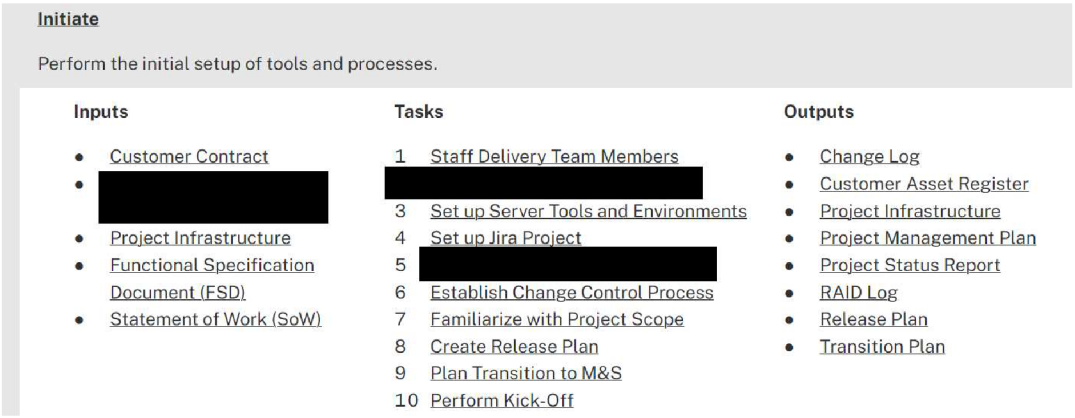
Zdroj: vlastní zpracování

Všechny fáze a podfáze DFW jsou navrženy a zobrazeny v jednotné struktuře, kdy jsou pro každou z nich definovány požadované vstupy, prováděné aktivity a očekávané výstupy, jež jsou blíže popsány níže:

- vstupy představují artefakty, které jsou povinné pro zahájení dané fáze, přičemž jde typicky o dokumenty, nastavené systémy nebo dostupné zdroje;
- aktivity poskytují seznam detailně rozepsaných činností, jež mají být v dané fázi vykonány;
- výstupy pak představují artefakty, jež musí existovat na konci dané fáze, přičemž povětšinou jde o výstupy jednotlivých činností.

Tato struktura je zobrazena v přehledné a dobře čitelné podobě, jak je možné vidět na obrázku 4 níže, představujícím fázi Zahájení projektu (Initiate):

Obrázek 4 Zobrazení fáze Zahájení projektu v rozhraní DFW (3 prvky byly odstraněny, neboť obsahovaly názvy proprietárních nástrojů)



Zdroj: vlastní zpracování (upraveno z DFW)

Vstupy a výstupy nejsou číslovány, protože na jejich pořadí nezáleží, ovšem u aktivit je tomu naopak.

3.1.1 Podpora obchodního týmu před uzavřením zakázky

Tato fáze popisuje podporu prodejních jednotek při vytváření návrhu řešení nebo služby, které lze zákazníkovi dodat v požadovaném čase, rozpočtu a kvalitě. Centrální a regionální projektové týmy tak poskytují podporu týkající se zejména technického návrhu daného řešení, jeho možností a omezení, odhadů pracnosti a možných rizik, na jejichž základě je vytvořena nabídka a případná smlouva.

Aktivity v této fázi zahrnují:

- analýzu požadavků;
- vytvoření návrhu řešení;
- validace návrhu řešení a jeho schválení;
- vytvoření podrobnějšího popisu řešení, které typicky představuje funkční specifikace a rozšířený popis technického návrhu;
- volba modelu dodávky – lokální tým, kombinace lokálních a vzdálených týmů, vzdálený tým;
- odhad celkové pracnosti;
- vytvoření rozšířeného popisu rozsahu projektu včetně akceptačních kritérií na základě návrhu zákazníka;
- schválení rozšířeného popisu rozsahu projektu.

Výstupem této fáze je funkční specifikace, odhad celkové pracnosti a schválený rozšířený popis zadání a rozsahu projektu, který se stane součástí smlouvy se zákazníkem.

3.1.2 Kontrolní bod – rozhodnutí o zahájení

Aktivity předchozí fáze lze do určité míry chápat jako investici, neboť společnost je provádí na vlastní náklady za účelem poskytnutí nabídky a získání dané zakázky. Zejména u velkých a komplexních projektů může jít až o několik týdnů trvající činnost, na níž participuje dočasně

vytvořený podpůrný tým složený z pracovníků potřebných odborností a která je řízena projektovým manažerem.

Po dokončení požadovaných aktivit nezřídka dochází k tomu, že lze a je výhodné (s ohledem na kupříkladu již nastavenou infrastrukturu, získaný přehled o navrženém řešení a kapacity potřebných pracovníků) pokračovat a zahájit práce na samotném projektu, ačkoli v dané chvíli neexistuje kontrakt nebo jistota, že bude uzavřen.

To s sebou však pochopitelně nese značná rizika, a proto rámec DFW předepisuje povinný postup ve dvou možných situacích:

1. jestliže není uzavřena smlouva, ale existuje dohoda o společném záměru nebo její ekvivalent, projekt lze zahájit s písemným souhlasem vedoucího dodávek pro danou zemi;
2. jestliže není uzavřena smlouva a neexistuje ani žádná dohoda se zákazníkem, projekt lze zahájit s písemným souhlasem vedoucího projektových dodávek pro danou oblast (například Evropu), což se může stát tehdy, je-li například strategický zájem na získání daného zákazníka vyšší než riziko.

Výstupem z tohoto kontrolního bodu je tedy dokumentované rozhodnutí o zahájení projektu, které je zásadní v případě absence smlouvy se zákazníkem, neboť poskytuje zdůvodnění vynaložení prostředků na projektové činnosti.

3.1.3 Zahájení projektu

Fáze zahájení projektu zahrnuje především zajištění veškeré potřebné infrastruktury, sestavení nebo doplnění projektového týmu a detailní seznámení s navrženým řešením. Klíčovými prvky této fáze jsou následující:

- příprava nebo rozšíření projektového plánu;
- nastavení projektu v systému Jira;
- nastavení a zajištění dostupnosti všech požadovaných prostředí;
- analýza navrženého řešení a vytyčení nejasností (v případě, že projektový tým je odlišný od toho, který řešení navrhl);
- tvorba harmonogramu dodávek (jaký rozsah bude doručen v jaké verzi);
- tvorba plánu předání do oddělení údržby a podpory, je-li součástí smlouvy;
- provedení zahajovací schůze.

V závěru této fáze by již neměly existovat významnější nejasnosti v požadavcích ani postupu realizace navrženého řešení.

Výstupy z této fáze zahrnují připravená vývojová prostředí a technické nástroje, projekt v systému Jira, aktualizovaný projektový plán, plán dodávek a další.

Uspořádáním zahajovací schůze projektu, které se účastní celý projektový tým (méně často pak také zákazník), projekt oficiálně přechází do následující fáze – doručování.

3.1.4 Doručování

Ve fázi doručování je navržené řešení detailně rozplánováno a následně vyvíjeno a testováno v kratších cyklech, představujících typicky určité ucelené části celkového řešení. Z toho důvodu je tato fáze jako jediná členěna do čtyř podfází, které se pro každý cyklus opakují:

- plánování – určení rozsahu dodávky pro daný cyklus, nastavení vstupních a výstupních kritérií, odhady pracnosti na úrovni nejnižších úkolů a určení strategie vývoje;
- implementace – vývoj, tvorba testů a jejich exekuce, řešení chyb, tvorba dokumentace;
- akceptace – asistence zákazníkovi při nastavení prostředí a instalaci dodávky, podpora akceptačního testování a řešení nalezených problémů;
- podpora – poskytování podpory dodávky během jejího nasazení v produkci a po dobu úvodní fáze produkčního provozu.

Zahrnutí podfází Akceptace a Podpory do cyklu doručování se může zdát nevhodné, ve skutečnosti jde však o vyjádření možnosti předat zákazníkovi k akceptaci (a následné poskytovat podporu) výstup z každého jednotlivého cyklu.

S tím se lze setkat především u projektů bez pevně stanoveného rozsahu nebo u takových, které jsou předmětem častých změn. V praxi jsou však takové projekty spíše výjimečné, a tyto podfáze jsou tak spuštěny pouze ve finálním cyklu, a tedy pro celé dodávané řešení.

Struktura fáze Doručování je koncipována s důrazem na využití agilního rámce Scrum, jehož principy využívá podfáze Implementace následovně:

- vývoj probíhá v iteracích, sprintech, jejichž délka je typicky kolem 2-3 týdnů;
- položky (typicky požadavky na nové funkce) jsou strukturovány do menších, samostatně fungujících celků, které lze dokončit v rámci jednoho sprintu;
- tým pracuje pouze na položkách, které splňují vstupní kritéria (Definition of Ready) definovaná v podfázi Plánování;
- dokončené položky lze uzavřít pouze tehdy, jsou-li splněna výstupní kritéria (Definition of Done) definovaná v podfázi Plánování;
- na konci sprintu je fungující produkt, který by bylo potenciálně možné doručit zákazníkovi;
- vývoj a testování každé položky probíhá společně v rámci jednoho sprintu, přičemž pouze plně otestována položka může být považována za dokončenou, v souladu s kritérii Definition of Done;
- tým provádí na konci každého sprintu retrospektivu.

Jak bylo uvedeno, vzhledem k nízkému zapojení zákazníků během vývoje je výše popsaný proces používán pouze interně v rámci společnosti. Ačkoli lze projektového manažera chápat jako určité zastoupení vlastníka produktu, zákazníka, přírůstek produktu na konci sprintu mu není demonstrován, protože nemůže poskytnout relevantní zpětnou vazbu – jde o součást týmu dodavatele, který má stejné informace jako vývojový tým.

3.1.5 Uzavření

Životní cyklus softwarové dodávky končí ve chvíli, kdy je řešení akceptováno zákazníkem a zároveň uplynula doba, po kterou se společnost zavázala nabízet podporu během a krátce po jeho nasazení do produkce.

Pokud je součástí smlouvy poskytovaní dlouhodobé podpory zákazníkovi, jde o rozšiřující službu poskytovanou odlišnou částí organizace a v jiném režimu, která nepatří do životního cyklu původní dodávky.

Aktivity této fáze se tak týkají především předáváním dokumentace do oddělení podpory (je-li poskytována), archivací dat, uzavíráním jednotlivých nástrojů a závěrečným hodnocením projektu realizačním týmem.

3.2 Analýza aktivit zajišťování a řízení

Posouzení stávajícího rozsahu a účinnosti aktivit zajišťování a řízení kvality bylo provedeno ve dvou krocích:

1. zjištění nákladů spojených s nízkou kvalitou a rozbor příčin, což poskytuje informace o účinnosti původního řešení;
2. rozbor relevantních aktivit a úkolů ve všech fázích a podfázích, které DFW pro dodávky řešení popisuje. Kontrolní bod je pouze procesním prvkem pro zahájení činnosti na projektu a neobsahuje žádné aktivity, nebude tedy posuzován.

Aby mohla být aktivita považována za relevantní z pohledu zajišťování nebo řízení kvality, musela splňovat jednu z následujících podmínek:

- zabývat se nastavením procesu nebo hodnocením jeho souladu s požadavky nebo jeho efektivitou;
- zabývat se hodnocením výstupu (produktu, dílčího produktu, dokumentu a podobně).

Výstupy z těchto aktivit byly porovnány s příslušnými daty v referenčním projektech, pokud byla pro danou aktivitu k dispozici.

3.2.1 Náklady spojené s nízkou kvalitou na referenčních projektech

U každého ze tří referenčních projektů, ke kterým byly dodány požadované informace, bylo nutné vypočítat celkové náklady vynaložené na činnosti související s opravami nalezených chyb.

To bylo možné provést relativně přesně, neboť firemní proces vyžaduje, aby každá nahlášená a validní chyba, kterou představuje záznam v systému Jira, povinně obsahovala následující tři úkoly:

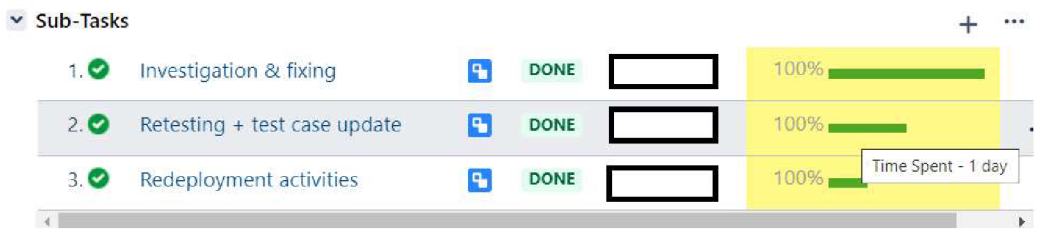
- vyšetření a oprava chyby;
- přetestování a doplnění testovacích scénářů;
- podpůrné aktivity.

Zároveň jsou všichni pracovníci, kteří na jednotlivých úkolech pracují, povinni u každého z nich zaznamenat vynaložený čas.

Určitým omezením je skutečnost, že systém umožňuje uzavřít úkol i tehdy, nezadá-li uživatel žádný vynaložený čas. Díky tomu se nezřídka stává, že kvůli opomenutí je tato hodnota doplněna zpětně, což může zvyšovat její zkreslení.

Popsanou strukturu úkolů vyžadovaných pro každé chybové hlášení v systému Jira demonstruje obrázek 5 níže:

Obrázek 5 Úkoly pro každou hlášenou chybu, jejich stav a doba trvání



Zdroj: vlastní zpracování (upraveno z podnikového systému Jira)

Tento požadavek ve výsledku poskytuje projektovému manažerovi lepší přehled o stavu opravných prací, nákladech vynaložených na opravy jednotlivých chyb a také představu o tom, kolik úsilí je celkově věnováno opravám v porovnání s běžnou projektovou činností.

Protože týmy jsou v praxi složené z pracovníků s různými sazbami (junior, medior, senior, expert), při výpočtu bylo kalkulováno s průměrnou hodinovou sazbou 50 EUR, a tedy denním nákladem 400 EUR na člověka.

Na základě shromážděných dat byly náklady oprav chyb následně rozděleny do tří kategorií – vývoj, testování a ostatní. Pro každou z nich byl pak vypočítán celkový náklad jako součin času ve dnech zaznamenaných na všech úkolech spadajících do dané kategorie a průměrné denní sazby.

Protože chyby nalezené po dodání produktu (a tedy uniklé internímu testování) s sebou nesou vyšší vedlejší náklady, v souladu s metodikou práce byly pro každou chybu nalezenou zákazníkem ve fázi akceptace, pilotu nebo produkčního provozu navýšeny náklady na její opravu o cenu dvou člověkodnů, tedy o částku 800 EUR. Ty byly poté připočteny do kategorie Ostatní.

Tabulka 1 poskytuje přehled vypočtených nákladů podle definovaných kategorií spolu s klíčovými údaji jednotlivých referenčních projektů, jež jsou níže označeny zkratkami P1,P2 a P3:

Tabulka 1 Přehled základních charakteristik poskytnutých referenčních projektů a odvozených nákladů na opravy chyb

Název	Rozsah (MD)	Chyb celkem	Po dodání	Opravy (MD)	Poměr oprav k nákladům projektu	Náklady oprav (EUR)			
						Vývoj	Testy	Ostatní	Celkem
P1	540	90	9	61	14,6 %	14 000	8 400	9 200	31 600
P2	590	111	16	71	17,5 %	16 200	11 000	14 200	41 200
P3	620	128	11	76	15,8 %	17 000	11 000	11 200	39 200

Zdroj: vlastní zpracování na základě firemních dat

Sloupec Rozsah udává skutečnou pracnost projektu zjištěnou jako součet časů zaznamenaných u všech položek v příslušném Jira projektu.

Pro účely výpočtu dodatečných nákladů na opravy chyb není mezi jednotlivými fázemi, ve kterých zákazník chybu objevil, rozlišováno, a jsou tak uvažovány souhrnně ve sloupci Po dodání.

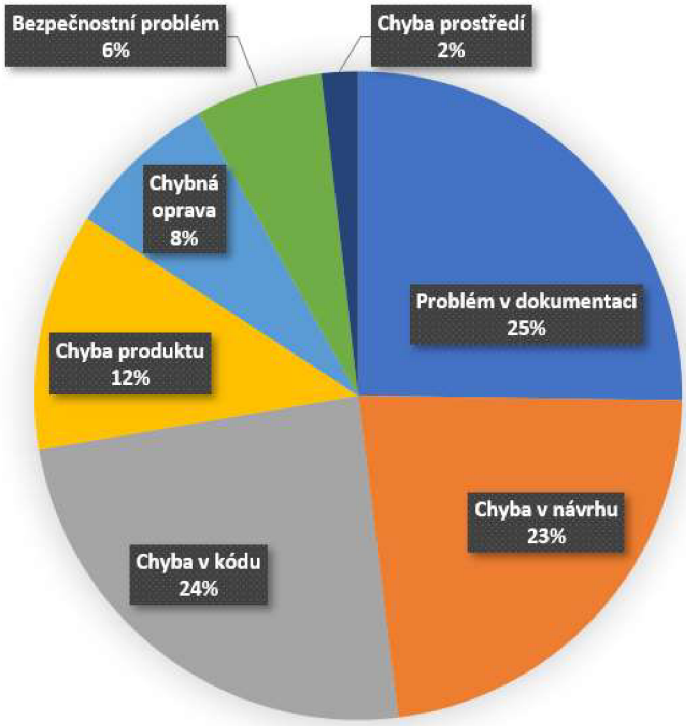
Náklady na opravy chyb vychází z jejich pracnosti navýšené o výše definovanou přirážku za každou chybu nalezenou po dodání projektu, přičemž následně je vyjádřen poměr těchto nákladů k celkovým nákladům projektu.

Systém Jira je ve společnosti nastaven tak, že každý záznam o nalezené chybě musí mít před svým konečným uzavřením povinně vyplněno pole Root Cause Analysis (Analýza kořenové příčiny), v němž je nutné vybrat jednu z řady předdefinovaných možností. To teoreticky umožňuje následně analyzovat časté příčiny chyb a zaměřovat se na jejich řešení v rámci průběžného zlepšování.

Příklady kořenových příčin, jež mohou uživatelé volit, zahrnují například problém v dokumentaci, chybu návrhu, chybu v kódu, chybnou opravu, selhání zavlečené produktem třetí strany a podobně.

V uvažovaných referenčních projektech bylo v součtu nalezeno 329 chyb, jejichž kořenové příčiny byly zaznamenány a seskupeny podle četnosti jejich výskytu. Zastoupení každé skupiny je zobrazeno v grafu 3 níže.

Graf 3 Přehled četnosti kořenových příčin chyb v referenčních projektech



Zdroj: vlastní zpracování na základě získaných dat

Zjištěných 7 skupin kořenových příčin bude nyní popsáno podrobněji s ohledem na typ chyb, jaké do nich byly řazeny (řazeno sestupně od nejpočetnější skupiny):

- problém v dokumentaci: chyby vzniklé důsledkem chybné interpretace, nedostatečné specifikace, nejasného nebo chybného vyjádření (včetně diagramů apod.), chybějícího nebo neúplného popisu;
- chyba v kódu: chyby způsobené při samotném vývoji, typicky chyby logické nebo datové, způsobené pochybením člověka – programátora;
- chyba v návrhu: chyby vzniklé důsledkem nesprávného, nekompletního nebo nevhodného návrhu řešení;
- chyba produktu: při dodávkách softwarových řešení zákazníkům společnost rozšiřuje a upravuje standardní produkty ze svého portfolia. Je-li vlivem nově přidané funkčnosti odhalena chyba přítomná v samotném základním produktu, je zařazena do této kategorie;
- chybná oprava: chyby zavlečené do zdrojového kódu řešení během opravy jiných chyb, kdy typicky jde o nežádoucí účinek provedené změny;
- bezpečnostní problém: chyba způsobená nevhodně zvoleným řešením při návrhu nebo implementaci, které je sice funkční, avšak nesplňuje interní nebo zákazníkem stanovené bezpečnostní požadavky;
- chyba prostředí: chyby způsobené nesprávným nastavením prostředí – servery, databáze, externí data a podobně. Tyto chyby jsou často způsobeny nedostatečnou dokumentací podpůrných procesů a úkonů.

Tabulka 2 níže zachycuje rozložení chyb na jednotlivých referenčních projektech podle kategorie kořenové příčiny:

Tabulka 2 Distribuce chyb podle projektu a kategorie kořenové příčiny

Název	Počet chyb celkem	Počet chyb podle kategorií kořenových příčin						
		Dokumentace	Návrh	Kód	Produkt	Prostředí	Bezpečnost	Oprava
Projekt 1	90	28	21	10	9	5	8	9
Projekt 2	111	22	24	25	19	0	9	12
Projekt 3	128	32	26	37	14	9	1	9

Zdroj: vlastní zpracování

Z celkového počtu 329 chyb bylo 36 z nich (11 %) hlášeno zákazníkem po dodání řešení a tedy uniklo aktivitám řízení kvality vývojovému týmu. Na základě analýzy chybových hlášení byly identifikovány následující příčiny:

- nedostatek v testovacích případech nebo testovacích datech (24; 66,7 %). Typickým problémem bylo opomenutí okrajových situací a hodnot ve specifikovaných požadavcích, méně často pak opomenutí celého požadavku nebo chybný zápis testu.
- netestovaný mimofunkční požadavek (6; 16,7 %). Dvě z těchto chyb souvisely s příliš pomalým zpracováním požadavku uživatele, jedna se týkala špatné čitelnosti/viditelnosti doprovodných textů na jedné z obrazovek a zbývající tři byly spojeny s nízkou robustností produktu;
- lidská chyba při exekuci testu (6; 16,7 %). U všech šesti případů bylo zjištěno, že příslušné testy byly pro danou verzi produktu nastaveny do stavu „Splněno“ ve stejný čas, pravděpodobně tedy vinou pochybení při hromadné operaci, kdy testeři v některých případech zpětně nastavují v jednom kroku určitý stav většího množství dříve provedených testů – například z důvodu dočasné nedostupnosti systému nebo opomenutí.

Výše uvedené tak představuje nedostatky procesu interního testování, jejichž odstranění nebo zmírnění by mělo vést ke snížení počtu chyb neodhalených vývojovým týmem a tedy nákladů na jejich opravu v případě, kdy dojde k jejich nahlášení zákazníkem.

3.2.2 Posouzení fáze Podpora obchodního týmu před uzavřením zakázky

V této fázi jsou s ohledem na podmínky definované výše relevantní následující dvě aktivity předepsané DFW:

1. validace návrhu řešení a jeho schválení;
2. schválení rozšířeného popisu rozsahu projektu.

Během validace návrhu řešení zkušení systémoví architekti potvrzují, že jednotlivé produkty lze použít a přizpůsobit v navržené kombinaci a v podmínkách stanovených požadavky projektu. Posuzován je pouze vysokoúrovňový návrh a jeho hrubý odhad pracnosti, který byl vytvořen v předcházející aktivitě této fáze, jež není součástí toho posouzení.

Pracnost testování je určena pouze paušálně jako 30 % z pracnosti vývoje, bez ohledu na jeho složitost. Tento poměr (70:30) je v podniku v souvislosti s plánováním pracnosti testování a souvisejících aktivit velmi častou praxí, přestože není nijak zdůvodněn.

Praktickým nedostatkem tohoto postupu je skutečnost, že obchodní týmy během přípravy nabídek požadují vytvoření podkladů od seniorních až expertních pracovníků v centrálních

odděleních, kteří však prakticky nikdy nebudou součástí týmu, který bude projekt doručovat. V praxi to znamená, že jimi odhadnutá pracnost projektu bude závazná pro zpravidla výrazně méně zkušený realizační tým, kterému bude projekt přidělen.

Tento problém vede k tomu, že projekty jsou mnohdy oproti původnímu odhadu pracnější, jak je demonstrováno v tabulce 3 níže, kde je porovnávána pracnost původně odhadnutá při tvorbě nabídky a následně pracnost skutečná, vycházející z celkového času zaznamenaného na položkách v příslušném Jira projektu.

Tabulka 3 Porovnání původně odhadované a skutečné pracnosti jednotlivých projektů

Název	Rozsah (MD)	Odhadnutá pracnost (MD)	Rozsah / odhadnutá pracnost
Projekt 1	540	500	108 %
Projekt 2	590	500	114 %
Projekt 3	620	580	106 %

Zdroj: vlastní zpracování na základě firemních dat

Přestože původní odhad pracnosti není dramaticky odlišný od skutečně vykázané pracnosti (nižší o 8 %, 14 % a 6 %), lze předpokládat, že projektový tým je pod značným tlakem, aby tento rozdíl minimalizoval, což může mít negativní dopad na kvalitu prováděných činností a následně tak kvalitu výsledného produktu.

Po vypracování rozšířeného popisu rozsahu řešení a funkční specifikace, jež budou součástí nabídky a následného kontraktu se zákazníkem, je vyžadováno jejich schválení projektovým manažerem, vedoucím dodávek pro danou zemi a vedoucím dodávek pro daný region, přičemž technická správnost a proveditelnost (s ohledem na časový rámec projektu) detailního návrhu a specifikace řešení se již neposuzuje.

Jedinou aktivitou řízení kvality je tak v této fázi validace vysokoúrovňového návrhu, neboť druhá relevantní aktivita obsahuje pouze proceduru pro schválení, aniž by byly schvalované dokumenty jakkoli hodnoceny.

3.2.3 Posouzení fáze Zahájení projektu

První relevantní aktivitou v této fázi je *Analýza navrženého řešení a vytyčení nejasností*. Klíčoví členové projektového týmu, který je typicky odlišný od skupiny expertů, která řešení navrhla, prochází zadání projektu, navržené řešení a funkční specifikaci, aby mohl být vytvořen plán dodávek na základě již stanovené doby trvání a odhadů pracnosti.

V rámci této aktivity mohou být odhaleny chyby a nejasnosti v navrženém řešení, neúplné zadání požadavků či jejich nejasnost. Případné nedostatky požadavků jsou předány zákazníkovi pro dodatečné objasnění.

Zde je problematická skutečnost, že na této aktivitě participují pouze klíčoví (většinou 1 až 3) členové týmu, kteří posuzují zásadní aspekty navrženého řešení i funkční specifikaci, a to včetně závazných odhadů pracnosti provedených již v rámci podpory obchodního týmu během přípravy nabídky. Zástupce funkce testování chybí, testovatelnost požadavků a řešení, stejně jako pracnost, tedy není explicitně posuzována.

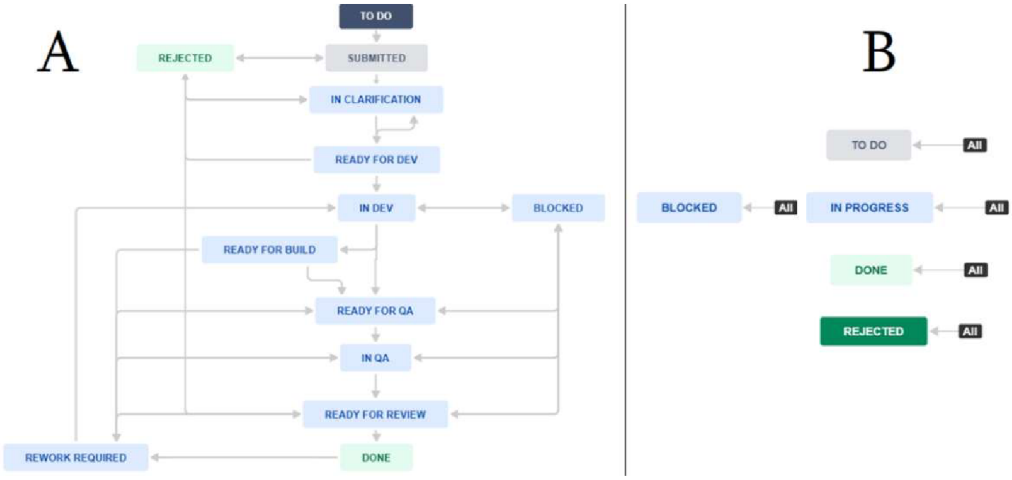
Druhou relevantní aktivitou je *Nastavení projektu v systému Jira*, kde na základě projektového požadavku tým spravující vývojové nástroje ve společnosti vytvoří nový projekt v systému Jira podle jedné z předdefinovaných šablon a přiřadí projektovému manažerovi (a mnohdy i dalším členům týmu) roli administrátora.

Tento postup má však značný nedostatek v tom, že DFW nespecifikuje projektovou šablonu a automaticky předpokládá, že tým projekt realizuje v jedné konkrétní z nich, která je sice oficiálně (mimo DFW) doporučena, její použití však není ověřováno.

V praxi to vede k tomu, že projekty mnohdy volí jinou, jednodušší šablonu, která má pro položky nových funkcí jen omezený počet stavů, a přechody mezi stavy nejsou omezeny žádnými podmínkami nebo nastavenými rolemi (kdokoli může udělat cokoli). To následně znamená, že dodržování procesu není nijak vynucováno nástrojem a závisí pouze na disciplíně týmu.

Na obrázku 6 níže jsou zobrazeny pro porovnání dvě šablony – komplexní, doporučená (A) a zjednodušená (B).

Obrázek 6 Projektové šablony v systému Jira



Zdroj: vlastní zpracování (upraveno z firemního systému Jira)

V šabloně A je patrná množina stavu a přechodů mezi nimi, které odpovídají životnímu cyklu vyvíjené funkce, zatímco šablona B umožňuje pouze nastavení základních stavů, mezi nimiž lze přecházet bez omezení.

Z referenčních projektů byla zjednodušená šablona nevhodně zvolena a používána u jednoho z nich.

3.2.4 Posouzení fáze Doručování

Ze čtyř podfází, do nichž je členěna fáze Doručování, jsou pro posouzení relevantní pouze Plánování a Implementace, neboť zbývající dvě popisují procesy následující po předání dodávky zákazníkovi.

V podfázi Plánování jsou relevantní následující aktivity:

1. nastavení vstupních kritérií: vytyčení a dokumentace kritérií, jež musí splňovat každá pracovní položka (typicky požadavek na funkčnost), než na ní může tým začít pracovat;
2. nastavení výstupních kritérií: vytyčení a dokumentace kritérií, jež musí pracovní položka splňovat, aby ji bylo možné považovat za dokončenou. Minimální výstupní podmínky zahrnují potvrzení, že byl kód otestován, zdokumentován a odeslán do repositáře;
3. nastavení strategie vývoje: obsahuje doporučení týkající se praktik vývoje, přičemž jednou z nich je požadavek na vytváření jednotkových testů, není však uveden ani typ pokrytí, ani požadovaná hodnota. Dále tato aktivita uvádí, že použití nástrojů pro

statickou analýzu kódu je třeba začlenit do každodenní praxe týmu, ovšem DFW nikde nevyžaduje revizi nebo hodnocení získaných dat.

S ohledem na výše uvedené aktivity byly na referenčních projektech identifikovány následující nedostatky:

- chybějící dokumentace vstupních kritérií (1 projekt) nebo jejich vágní znění (2 projekty) ve smyslu „Požadavek je připraven“;
- pouze minimální povinné výstupní podmínky (všechny 3 projekty);
- absence kontroly – v deseti případech na dvou projektech byla položka potvrzená jako kompletní, přestože nesplňovala jedno nebo více kritérií;
- žádný z projektů neměl k dispozici výsledky statické analýzy zdrojového kódu, přestože byl správně nastaven v nástroji SonarQube, který je ve společnosti dostupný všem projektům globálně;
- minimální, nepravidelné nebo zcela chybějící pokrytí jednotkovými testy.

Možným nedostatkem v samotném členění dané podfáze DFW je skutečnost, že neobsahuje aktivitu týkající se nastavení strategie testování.

V následující podfázi Implementace pak byly jako relevantní označeny tyto aktivity:

1. tvorba a dokumentace testů: nastavení nástroje pro správu testování, příprava testovacích plánů a návrh testovacích scénářů, pro což tato aktivita poskytuje určitá doporučení;
2. exekuce testů: provádění testovacích scénářů a dokumentace výsledků;
3. provedení konfirmačního testování: přetestování opravených chyb pro potvrzení úspěšné opravy a aktualizace nebo rozšíření testovacích scénářů tak, aby pokryly případný opětovný výskyt dané chyby.

Zmíněné aktivity mají několik nedostatků, jež jsou rozepsány níže:

- ačkoli u tvorby a dokumentace testů poskytuje DFW řadu doporučení, tato se týkají jmenné konvence testů, struktury jejich uspořádání, kategorizaci a podobně, neobsahují však žádné instrukce týkající se kvality testů, jejich typů nebo technik, které by testeři měli používat pro jejich návrh tak, aby bylo zajištěno optimální pokrytí produktu. Vzhledem k tomu, že při rozboru úniku chyb byl nedostatek v testu nebo v testovacích datech identifikován jako příčina u 66,7 % chyb, lze toto považovat významný problém;
- v případě exekuce testů DFW uvádí, že je vhodné doplnit ji exploratorním testováním, neposkytuje však opět žádné další informace nebo doporučení, například kolik úsilí by mělo být této aktivitě věnováno, jak ji měřit nebo zda a jak ji dokumentovat. U referenčních projektů tak nebylo možné ověřit, zda bylo exploratorní testování vůbec provedeno;
- zcela chybí informace týkající se potřeby regresního testování a doporučení pro jeho plánování a provádění;
- žádná z uvedených aktivit se ani okrajově nezabývá mimofunčními požadavky a jejich hodnocením.

U všech podfází je shodně poslední aktivitou „Provedení retrospektivy“, v jejímž rámci projektový tým jako celek hodnotí její průběh z pohledu procesního, technického i týmového. Cílem je pokračovat v tom, co je považováno jako pozitivní a žádoucí, a naopak omezit nebo zlepšit to, co se neosvědčilo nebo se jeví jako problematické či nevhodné. Jde tedy o efektivní mechanismus zpětné vazby.

K této aktivitě však není přidružen žádný očekáváný a dokumentovaný výstup, a nikde v DFW ani není odkazován. Z toho důvodu tedy nelze hodnotit, zda a do jaké míry je tato aktivita prováděna a její výstup – případná navržená zlepšení v procesech fungování týmu – reflektován v praxi.

3.2.5 Posouzení fáze Uzavření

V této konečné fázi je relevantní aktivita „Provedení projektové retrospektivy“, během níž všichni členové realizačního týmu diskutují projekt jako celek s ohledem na úspěchy i neúspěchy, překážky procesní i technické, vyřešené i trvající problémy, fungování týmu samotného a také možná zlepšení, což v souhrnu představuje důležitý prvek zpětné vazby pro průběžné zlepšování.

Výstup z této aktivity je formalizován ve formě dokumentu a následně umístěn do projektové knihovny, dostupné globálně všem projektovým manažerům ve společnosti, kteří tak mohou těžit ze sdílení znalostí a získat cenné informace a řešení, užitečné zejména na počátku projektu.

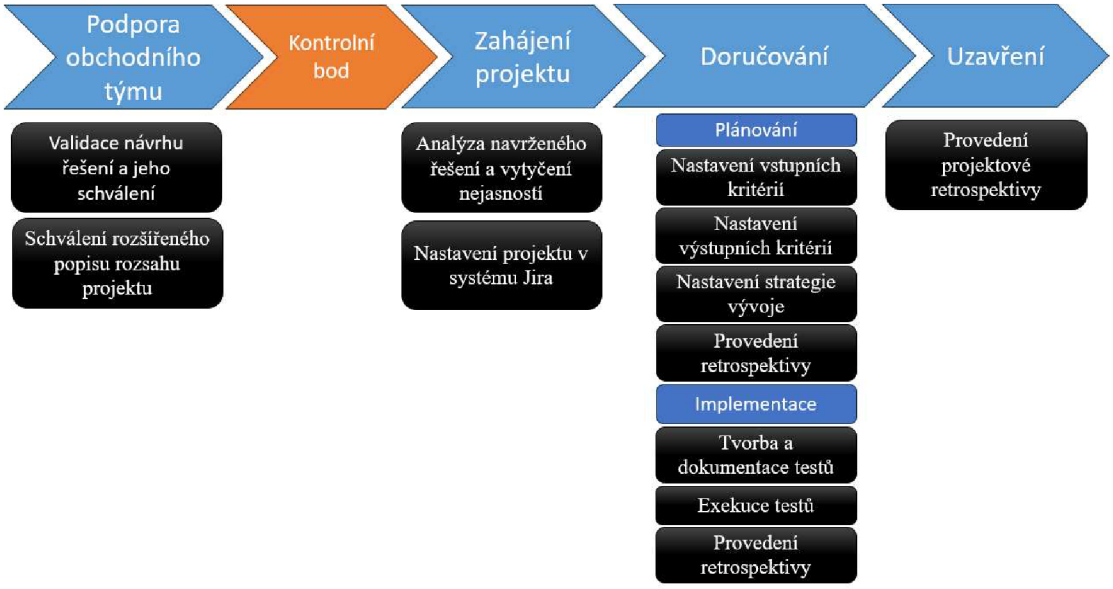
Určitým nedostatkem je však skutečnost, že rámec DFW nikde na tento dokument neodkazuje a nedoporučuje jeho použití, které se tak stává čistě závislé na tom, zda jednotliví projektoví manažeři o jeho existenci a možném přínosu vědí.

3.2.6 Výsledné zhodnocení přístupu ke kvalitě v rámci DFW

Z provedeného rozboru jednotlivých fází, podfází a jejich aktivit je patrné, že navzdory některým prvkům agilního vývoje přistupuje celkově rámec DFW ke kvalitě převážně reaktivně, tedy přístupem řízení kvality. Ten je navíc z většinové části zaměřen na finální produkt z jednotlivých iterací.

Na schématu DFW zobrazeném na obrázku 7 níže jsou znázorněny všechny relevantní aktivity identifikované v předcházejících podkapitolách a fáze, v nichž se nachází. Zmíněný trend převážně reaktivního přístupu ke kvalitě produktu je patrný z uspořádání daných aktivit, které jsou početnější směrem napravo, tedy později v cyklu vývoje.

Obrázek 7 Aktivity zajišťování a řízení kvality v rámci DFW



Zdroj: vlastní zpracování (upraveno z rámce DFW)

Hlavní činností řízení kvality je v DFW testování produktu, respektive jeho přírůstku na konci každé iterace ve fázi Doručování, podfázi Implementace, tedy z pohledu modelu životního cyklu velmi pozdě. Ačkoli DFW popisuje validaci návrhu řešení ve fázi Podpora obchodního návrhu a následně analýzu navrženého řešení ve fázi Zahájení projektu, v obou případech jde o omezené posouzení několika jednotlivci, které je provedeno zcela bez zapojení zástupců funkce testování.

Za aktivity zajišťování kvality lze považovat ty, jejichž předmětem je nastavení nebo kontrola procesů vývoje a testování (Nastavení projektu v systému Jira, Nastavení vstupních a Nastavení výstupních kritérií, Nastavení strategie vývoje) a provádění retrospektiv na konci iterací i celého projektu.

Zde jsou ovšem patrné nedostatky, které výrazně omezují dopad těchto aktivit a jejichž hlavní příčinou je úplná absence nebo nedostatečná míra kontroly souladu nastavených procesních požadavků se skutečným stavem, která je nedílnou součástí procesů zajišťování kvality.

3.3 Návrh zlepšení a posouzení aplikovatelnosti

Nedostatky identifikované v přístupu DFW ke kvalitě i v jednotlivých aktivitách jejího řízení a zajišťování představují možné příležitosti ke zlepšení, prostřednictvím kterých lze dosáhnout vyšší kvality produktů a tedy nižších nákladů spojených s chybami a jejich odstraňováním.

Při návrhu zlepšení byly zohledňovány kořenové příčiny chyb zjištěné na referenčních projektech, přičemž obecně bylo vycházeno z několika základních principů ekonomického aspektu kvality softwaru, popsaných v teoretické části této práce:

- prevence vzniku chyb je optimálním a nejvíce nákladově-efektivním přístupem, neboť od počátku projektu je kladen důraz na kvalitu vstupů, meziproduktů a tedy i výsledného produktu;
- existující chyby, jimž nebylo možné předejít, lze s nejnižšími náklady odstranit v té fázi, kdy došlo k jejich vzniku, přičemž čím později je chyba objevena, tím vyšší budou náklady na její odstranění. To se týká chyb v produktu, dílčích produktech i veškerých vstupech (specifikace, návrh a podobně);
- výhodnou strategií pro zlepšení kvality vyvíjeného softwaru v organizaci je „posun vlevo“, tedy snaha zavést preventivní i vhodné reaktivní aktivity řízení kvality co nejdříve v životním cyklu produktu.

Opatření navržená k odstranění zjištěných nedostatků budou pro každou relevantní fázi a podfázi rámce DFW diskutována v následujících podkapitolách. U jednotlivých opatření je pro snazší odkazování použito průběžné číslování.

3.3.1 Podpora obchodního týmu před uzavřením zakázky

Pro činnost *Validace návrhu řešení a jeho schválení* byla navržena následující dvě zlepšení:

1. zapojení seniorního pracovníka funkce testování do tvorby hrubého odhadu pracnosti řešení, který na základě vlastní interpretace a zkušeností poskytne odhad týkající se aktivit testování, případně potvrdí proveditelnost paušálního odhadu 30 % z pracnosti vývoje. Jako součást této činnosti budou dokumentována potenciální rizika spojená s testováním, takže je bude možné zohlednit již během přípravy zahájení projektu. Toto zlepšení bude začleněno do aktivity jako úkol s názvem *Nezávislé posouzení požadavků na testování*;

2. zavedení kontroly požadavků specificky cílené na jejich úplnost, jednoznačnost a mimofunkční aspekty, aby mohly být případné nejasnosti vyřešeny co nejdříve. Toto zlepšení bude začleněno do aktivity jako úkol s názvem *Revize připravenosti požadavků*;
3. zohlednění skutečnosti, že odhady pracnosti poskytované seniorními až expertními pracovníky jsou v praxi obecně nižší než úsilí vynaložené realizačním týmem, složeným z pracovníků s různými úrovněmi znalostí a dovedností. Na základě dat z referenčních projektů je proto navrženo, aby v případech, kdy na projektu nebude pracovat expertní tým nebo tým, který původní odhady poskytuje, byla odhadnutá pracnost navyšována o 6 %, což představuje nejnižší z vypočítaných hodnot, o kterou byla skutečná pracnost vyšší než původní odhad. Toto zlepšení bude začleněno do procedury výpočtu pracnosti jako nový koeficient s názvem *Faktor rozdílných zkušeností*.

Pro druhou relevantní aktivitu této fáze, *Schválení rozšířeného popisu rozsahu projektu*, bylo navrženo následující:

4. zavedení kontroly akceptačních kritérií a specifikace požadavků s ohledem na jejich úplnost, jednoznačnost, vzájemnou konzistenci, proveditelnost, testovatelnost a mimofunkční aspekty. Toto zlepšení bude začleněno do aktivity jako úkol s názvem *Revize akceptačních kritérií;*
5. rozšíření schvalovací procedury o krok zabývající se technickou revizí detailního návrhu řešení jinými pracovníky pro nezávislé posouzení jeho úplnosti, proveditelnosti, správnosti, rizik a mimofunkčních charakteristik včetně bezpečnosti. Tento krok bude začleněn pod názvem *Revize detailního návrhu.*

Tato zlepšení mají za cíl především včasnou identifikaci nedostatků v požadavcích a funkční specifikaci, zlepšení přesnosti odhadů pracnosti testování a zajištění revize detailního návrhu před jeho konečným schválením, což v součtu zvyšuje šanci, že případné problémy budou odhaleny v dostatečném předstihu.

Dále byla navržena nová aktivita, jejíž předmětem bude hodnocení souladu aktivit této fáze s jejich specifikacemi (rámcem DFW), dokumentace výsledků, zavádění zlepšení a posuzování úspěšnosti. Tato aktivita byla nazvána *Hodnocení procesu podpory obchodního týmu.*

3.3.1 Zahájení projektu

Pro činnost *Analýza navrženého řešení a vytyčení nejasností* byla navržena následující dvě zlepšení:

6. zapojení širšího okruhu členů projektového týmu, například prostřednictvím sdíleného přístupu k dokumentům a připomínkování, což umožňuje flexibilní sběr a dokumentaci vstupů i ve větším a distribuovaném týmu. Toto zlepšení bude začleněno do aktivity jako úkol s názvem *Týmové posouzení navrženého řešení*;
7. zavedení samostatné kontroly zástupci funkce testování z projektového týmu, jejichž zodpovědností je potvrdit testovatelnost funkčních a zejména mimofunkčních požadavků, analyzovat dříve zjištěná rizika související s testováním a navrhnout jejich ošetření. Toto zlepšení bude začleněno do aktivity jako úkol s názvem *Posouzení testovatelnosti navrženého řešení.*

U činnosti *Nastavení projektu v systému Jira* bylo navrženo následující:

8. zavedení dodatečného kroku kontroly, během níž bude ověřováno, že projektová šablona v nástroji Jira je zvolena správně, tedy odpovídá doporučení z rámce DFW.

Pokud je zvolena jiná, bude vyžadováno dokumentované rozhodnutí, které tuto volbu ospravedlní. Tento krok bude začleněn pod názvem *Potvrzení správnosti nastavení.*

Vzhledem k zásadnímu významu akceptačních kritérií byla jejich analýza ustanovena jako samostatná aktivita s názvem *Analýza akceptačních kritérií*. Jejím smyslem je zajištění, že jednotlivá akceptační kritéria týkající se funkčnosti a mimofunkčních charakteristik jsou jasně navázána na specifikované požadavky, takže realizační tým může těmto oblastem věnovat zvýšenou pozornost během vývoje i návrhu testů.

Smyslem navržených opatření je v tomto případě zajištění, že projektový tým před započetím prací dostatečně analyzoval navržené řešení z více perspektiv, s důrazem na testovatelnost požadavků a s vyhodnocením rizik souvisejících s testováním, aby mohly být případné sporné nebo nejasné body vyjasněny s obchodním týmem nebo přímo zákazníkem, je-li to možné.

Kontrola zvolené šablony v nástroji Jira slouží k tomu, aby projekt mohl být zahájen pouze s takovou šablonou, která plně podporuje a vyžaduje procesní přístup k řízení projektu a správu úloh dle doporučení DFW.

Stejně jako u předchozí fáze byla navržena nová aktivita, jejíž předmětem bude hodnocení souladu aktivit této fáze s požadavky DFW, dokumentace výsledků, zavádění zlepšení a posuzování úspěšnosti. Tato aktivita byla nazvána *Hodnocení procesu zahajování projektu.*

3.3.2 Doručování – Plánování

V podfázi Plánování byla shodně pro činnosti *Nastavení vstupních kritérií* a *Nastavení výstupních kritérií* navržena tato zlepšení:

9. zavedení kontroly, zda jsou příslušná kritéria nastavena účelně a proveditelně, v souladu s doporučeními DFW, a řádně na projektu zdokumentována. Uvedená kontrola bude do aktivity začleněna pod názvy *Potvrzení vstupních a Potvrzení výstupních kritérií*;
10. začlenění kontroly splnění podmínek do procedury správy úloh v systému Jira. V praxi může jít například o postup, kdy před zahájením prací musí u dané úlohy pracovník potvrdit, že vstupní podmínky jsou skutečně splněny, což bude viditelné v záznamu aktivity. Podobně při dokončení prací by uzavření úlohy mohlo vyžadovat potvrzení, že jsou výstupní kritéria splněna, což může být ustanoveno jako součást závěrečné revize u každé položky. Tato kontrola bude do aktivity začleněna pod názvy *Ověření splnění vstupních* a *Ověření splnění výstupních podmínek*.

Pro následující činnost, *Nastavení strategie vývoje*, byla navržena tato zlepšení:

11. ustanovení jasných požadavků na typ a minimální akceptované pokrytí zdrojového kódu jednotkovými testy. Protože projekty se mohou značně odlišovat v míře rizika a složitosti, kritéria by mohla být členěna do několika kategorií, kdy rizikové nebo komplexní projekty budou vyžadovat vyšší úroveň pokrytí než projekty, u nichž je riziko nízké. Toto zlepšení bude do aktivity začleněno pod názvem *Nastavení požadavků na jednotkové testy*;
12. ustanovení kontroly výstupu automatické analýzy zdrojového kódu jako samostatné úlohy, pravidelně prováděné zástupcem vývoje nebo projektovým manažerem, který za ni bude zodpovědný. Uvedená kontrola bude do aktivity začleněna pod názvem *Průběžná kontrola výsledku statické analýzy kódu*;
13. rozšíření DFW o praktiky agilního vývoje a jejich doporučení. Těmi může být například provádění vzájemné revize kódu u rizikových části produktu nebo párové programování s juniorními členy týmu, což usnadňuje přenos znalostí i podporuje žádoucí praktiky

vývoje. Toto zlepšení bude do aktivity začleněno jako doporučení pod názvem *Doporučené praktiky vývoje*.

Zlepšení navržená pro tuto podfázi jsou zaměřena převážně na zajištění, že jsou nastavené procesy a postupy – jejichž vliv na kvalitu je významný – dodržovány. Zavedení agilních praktik vývoje pak cílí na zlepšování samotného procesu vývoje.

Z pohledu podfáze jako takové je navržena nová činnost s názvem *Nastavení strategie testování*, která by měla popisovat požadavky na testování, kritéria, časové rámce a doporučení týkající se provádění regresních a konfirmačních testů. To umožní zástupcům funkce testování systematický a plánovaný přístup.

Kontrolu procesu lze začlenit do existující aktivity *Provedení retrospektivy*, kde je tak navrženo následující:

14. je periodicky hodnocen soulad prováděných aktivit s požadavky specifikovanými v DFW, výsledek je dokumentován, jsou navrhována opatření pro zjištění nedostatků a jejich účinnost je posuzována. Toto zlepšení bude do aktivity začleněno jako úkol s názvem *Ověření souladu procesů fáze s požadavky*;
15. struktura retrospektivy je rozšířena o dodatečný bod, v jehož rámci jsou diskutována zjištění z retrospektivy předchozí a ověřováno, že byly případné procesní změny skutečně provedeny. Toto zlepšení bude do aktivity začleněno jako úkol s názvem *Nastavení struktury a bodů povinné agendy*;
16. záznam o provedení retrospektivy je dokumentován spolu s jejími výsledky a je přístupný všem. Toto zlepšení bude do aktivity začleněno jako úkol s názvem *Dokumentace a sdílení výsledků*.

3.3.3 Doručování – Implementace

U podfáze Implementace byla pro aktivitu *Tvorba a dokumentace testů* navržena následující zlepšení:

17. rozšíření doporučení DFW o specifické požadavky na kvalitu testů a techniky používané při jejich návrhu – povinné členění na testy pozitivní a negativní, důraz na testování hraničních hodnot a nevalidní vstupy. Uvedené zlepšení bude začleněno jako nová sekce doporučení s názvem *Požadavky na návrh testů*;
18. rozšíření DFW o doporučení týkající se specifik a přístupů k testování mimofunkčních požadavků. Uvedené zlepšení bude začleněno jako nová sekce doporučení s názvem *Testování mimofunkčních požadavků*;
19. ustanovení kontroly kvality navržených testů a výsledků jejich exekuce prováděné periodicky seniorním pracovníkem funkce testování na projektu. Tato kontrola bude začleněna do aktivity pod názvem *Revize navržených testů a stavu exekuce*.

U následující činnosti, *Exekuce testů*, bylo navrženo následující:

20. rozšíření doporučení DFW o vyšší důraz na exploratorní testování a jasné požadavky na jeho provádění a dokumentaci. U rozsahu lze jako obecné, výchozí pravidlo aplikovat 80:20, tedy 20 % z celkové pracnosti testování by mělo být vyhrazeno na provádění exploratorních testů. Plánování a dokumentace exploratorního testování by měla být řízena přístupem testování v relacích. Uvedené zlepšení bude začleněno jako nová sekce doporučení s názvem *Požadavky na exploratorní testování*.

Pro činnost *Provedení konfirmačního testování* bylo navrženo níže uvedené zlepšení:

21. začlenění doporučení pro plánování a provádění konfirmačního testování do nově navržené aktivity *Nastavení strategie testování*. Toto zlepšení bude do této aktivity zahrnuto jako sekce s názvem *Požadavky na konfirmační testování*.

Navržená opatření vychází z příčin úniků chyb zjištěných na referenčních projektech a doporučené praxe, jsou proto zaměřena na více systematický přístup k návrhu testů i provádění samotného testování, důrazu na mimofunkční požadavky a exploratorní testování.

Aktivita *Provedení retrospektivy* je rozšířena stejným způsobem jako v předcházející podfázi, takže obsahuje procesní kontrolu i dokumentaci výsledků.

3.3.4 Uzavření

V závěrečné fázi rámce DFW, Uzavření, jsou pro aktivitu *Provedení projektové retrospektivy* navržena následující zlepšení:

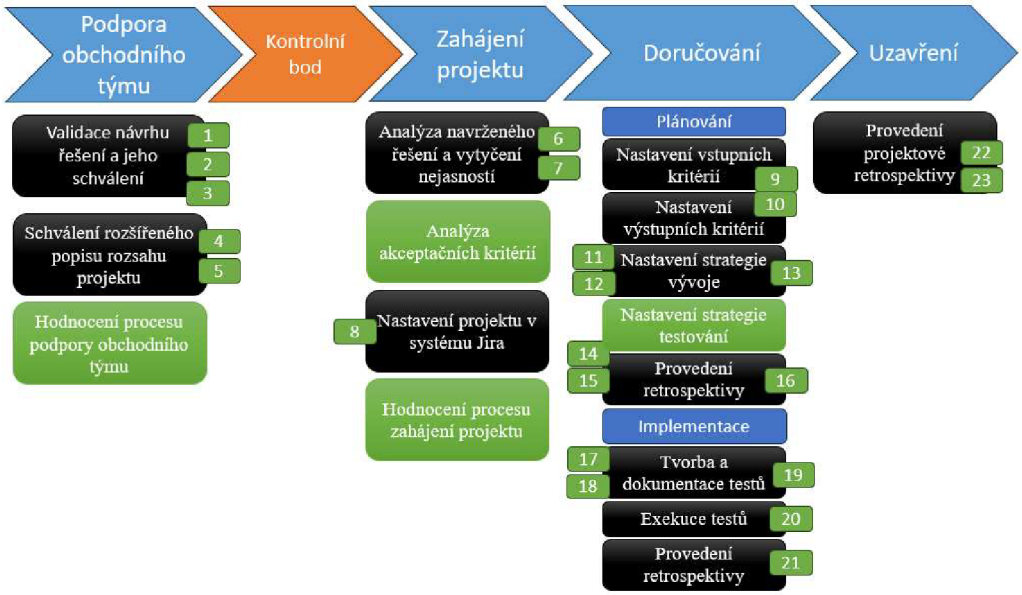
22. do dokumentovaného výstupu budou zařazeny (odkazem či přímo) i výsledky retrospektiv z jednotlivých iterací. Toto zlepšení bude začleněno do této aktivity jako úkol s názvem *Zahrnutí průběžných retrospektiv*;
23. ve fázi Zahájení budou výstupy z projektových retrospektiv odkazovány jako potenciální zdroje užitečných znalostí, a bude doporučeno jejich využití pro projekty s podobným kontextem. Uvedené zlepšení bude začleněno do stávající aktivity *Příprava nebo rozšíření projektového plánu* jako doporučení s názvem *Zohlednění informací z historických projektů.*

Smyslem výše uvedených zlepšení je trvalé zlepšování skrze zajištění využívání získaných znalostí mezi jednotlivými projekty a sdílení užitečných praktik a řešení problémů.

3.3.5 Náklady na navržená zlepšení

Navržená zlepšení jsou koncipována tak, aby podporovala proaktivní přístup ke kvalitě prostřednictvím nastavení vhodných procesů, kontrolou jejich provádění a včasného hodnocení vstupů a výstupu, zejména před započetím fáze Doručování, jak je patrné na obrázku 8 níže:

Obrázek 8 Aktivity zajišťování a řízení kvality v rámci DFW rozšířené o nové úkoly a aktivity



Zdroj: vlastní zpracování

Lze si všimnout, že oproti původnímu stavu je na obrázku výše patrný „posun vlevo“.

Odhadovaná pracnost (v člověkodnech nebo procentech) na provádění jednotlivých zlepšení je spolu se zdůvodněním uvedena pro každou fázi a podfázi níže.

Ve fázi Podpora obchodního týmu před uzavřením zakázky:

- na provedení úkolu *Nezávislé posouzení požadavků na testování* byly naplánovány 4 člověkodny, což představuje třetinu z průměrného trvání aktivity *Validace návrhu řešení a jeho schválení*. Tento odhad vychází z dříve popsané podnikové praxe (70:30);
- na provedení úkolu *Revize připravenosti požadavků* byly naplánovány 2 člověkodny pro projekty průměrného rozsahu, tedy podobného referenčním projektům, kdy se dokumentace požadavků (včetně vizuálních prvků) zadání pohybuje mezi 60-120 stranami;
- zavedení koeficientu *Faktor rozdílných zkušeností* navýší pracnost o 6 %, navýšení nákladů je tedy závislé na rozsahu projektu;
- na provedení *Revize akceptačních kritérií* byl naplánován 1 člověkoden;
- na provedení úkolu *Revize detailního návrhu* bylo naplánováno 18 člověkodnů, což odpovídá pracnosti revize vysokoúrovňového návrhu navýšené o 50 %, protože se jedná o složitější činnost;
- na provedení nově navržené aktivity *Hodnocení procesu podpory obchodního týmu* byl naplánován 1 člověkoden.

Pro fázi Zahájení projektu:

- na provedení úkolu *Týmové posouzení navrženého řešení* bylo naplánováno 8 člověkodnů;
- na provedení úkolu *Posouzení testovatelnosti navrženého řešení* byly naplánovány 2 člověkodny;
- na provedení aktivity *Analýza akceptačních kritérií* bylo naplánováno 0,5 člověkodne;
- na provedení kontroly *Potvrzení správnosti nastavení* nebyl plánován žádný čas, protože jde o procesní krok s trváním v řádu minut;
- na provedení nově navržené aktivity *Hodnocení procesu zahajování projektu* byl naplánován 1 člověkoden.

Pro podfázi Plánování ve fázi Doručování:

- pro kontroly *Potvrzení vstupních* a *Potvrzení výstupních kritérií* byl naplánován v součtu 1 člověkoden, neboť se jedná o revizi několika odstavců textu, kterou je nutné provést po každé úpravě, která se děje nejvýše jednou během každé iterace;
- pro kontroly *Ověření splnění vstupních* a *Ověření splnění výstupních podmínek* byl naplánován v součtu 1 člověkoden. Tento odhad je založen na skutečnosti, že případné dva nové stavy u položek v nástroji Jira budou ve výsledku znamenat, že při jejich údržbě (aktualizaci stavu a podobně) zde členové vývojového týmu v součtu stráví více času;
- pro *Nastavení požadavků na jednotkové testy* nebyl plánován žádný čas, neboť jde o jednorázovou aktivitu, během níž by měla být při zahájení projektu přejata kritéria z rámce DFW, upravena na základě projektových specifik a zadokumentována jako povinná;
- pro úkol *Průběžná kontrola výsledků statické analýzy kódu* byly naplánovány 2 člověkodny. Tento odhad byl odvozen z času plánovaného pro projektového manažera na ostatní, pravidelně prováděné kontroly (například kontrola postupu prací);

- pro zavedení doporučení *Doporučené praktiky vývoje* nebyl plánován žádný dodatečný čas, protože čas spojený s tréninkem juniorních členů týmu a prováděním agilních praktik by měl ve výsledku přinést vyšší efektivitu vývoje, méně opravných prací a tyto náklady tak pokrýt;
- pro aktivitu s *Nastavení strategie testování* bylo na základě pravidla 70:30 naplánováno 1,5 člověkodne;
- pro změny v provádění retrospektivy (úkoly *Ověření souladu procesů fáze s požadavky*, *Nastavení struktury a bodů povinné agendy* a *Dokumentace a sdílení výsledků* nebyl žádný čas, protože se jedná o požadavky na strukturu a ukládání výsledků.

Pro podfázi Implementace ve fázi Doručování:

- pro zavedení doporučení *Požadavky na návrh testů* a *Testování mimofunkčních požadavků* nebyl plánován žádný čas, protože správný návrh a testování mimofunkčních požadavků musí být součástí odhadů pracnosti;
- pro úkol *Revize navržených testů a stavu exekuce* bylo naplánováno 10 člověkodnů, což u referenčních projektů odpovídá zhruba 20 % pracnosti plánované pro vytváření testů;
- pro provádění doporučení ze sekce *Požadavky na exploratorní testování* nebyl plánován žádný čas, protože se jedná o aplikaci systematického postupu a zajištění, že z plánovaného času je odpovídající část vyhrazena pro exploratorní testování;
- pro provádění doporučení ze sekce *Požadavky na konfirmační testování* nebyl plánován žádný čas, protože se jedná o aplikaci systematického postupu pro aktivitu, která je již rutinně prováděna.

Pro fázi Uzavření:

- pro úkoly *Zahrnutí průběžných retrospektiv* a *Zohlednění informací z historických projektů* nebyl čas plánován, neboť se jedná o pokyny upřesňující již existující aktivity.

Všechna zlepšení, jejichž zavedení vyžaduje navýšení časového odhadu pro dokončení projektu, jsou spolu s odhadovanou pracností uvedena v tabulce 4 níže:

Tabulka 4 Přehled zlepšení a potřebného času v člověkodnech (MD)

Zlepšení	MD	Zlepšení	MD
Nezávislé posouzení požadavků na testování	2	*Analýza akceptačních kritérií*	0,5
Revize připravenosti požadavků	4	*Hodnocení procesu zahajování projektu*	1
Revize akceptačních kritérií	1	*Potvrzení vstupních* a *Potvrzení výstupních kritérií*	1
Revize detailního návrhu	18	*Ověření splnění vstupních* a *Ověření splnění výstupních podmínek*	1
Hodnocení procesu podpory obchodního týmu	1	*Průběžná kontrola výsledku statické analýzy kódu*	2
Týmové posouzení navrženého řešení	8	*Nastavení strategie testování*	1,5
Posouzení testovatelnosti navrženého řešení	2	*Revize navržených testů a stavu exekuce*	10

Zdroj: vlastní zpracování

V součtu tak činí celková pracnost všech navržených zlepšení 53 člověkodnů, v níž však není započítán vliv koeficientu *Faktor rozdílných zkušeností*, jehož zavedení by zvýšilo plánovanou pracnost projektu o 6 %.

Pro výpočet nákladů je pracnost vynásobena průměrným nákladem na jeden člověkoden, jež byl ustanoven v podkapitole 3.2.1 výše.

Výsledné náklady na jednotlivá navržená zlepšení (vyjádřené v EUR) jsou uvedeny v tabulce 5 níže:

Tabulka 5 Přehled nákladů na jednotlivá zlepšení

Zlepšení	EUR	Zlepšení	EUR
Nezávislé posouzení požadavků na testování	800	*Analýza akceptačních kritérií*	200
Revize připravenosti požadavků	1 600	*Hodnocení procesu zahajování projektu*	400
Revize akceptačních kritérií	400	*Potvrzení vstupních* a *Potvrzení výstupních kritérií*	400
Revize detailního návrhu	7 200	*Ověření splnění vstupních* a *Ověření splnění výstupních podmínek*	400
Hodnocení procesu podpory obchodního týmu	400	*Průběžná kontrola výsledku statické analýzy kódu*	800
Týmové posouzení navrženého řešení	3 200	*Nastavení strategie testování*	600
Posouzení testovatelnosti navrženého řešení	800	*Revize navržených testů a stavu exekuce*	4 000

Zdroj: vlastní zpracování

Níže jsou pak uvedena všechna zlepšení, která teoreticky nevyžadují žádný dodatečný čas, neboť poskytují upřesňující pokyny nebo doporučení pro nastavení procesů či kritérií v aktivitách, které jsou již prováděny:

- *potvrzení správnosti nastavení;*
- *nastavení požadavků na jednotkové testy;*
- *doporučené praktiky vývoje;*
- *požadavky na návrh testů;*
- *změny v průběžných retrospektivách;*
- *testování mimofunkčních požadavků;*
- *požadavky na exploratorní testování;*
- *požadavky na konfirmační testování;*
- *zahrnutí průběžných retrospektiv;*
- *zohlednění informací z historických projektů.*

V praxi však provedení úprav stávajících praktik, postupů a kontrola jejich dodržování bude určitý čas vyžadovat, proto byl na každé z výše uvedených zlepšení naplánován paušálně 1 člověkoden. Tato zlepšení v součtu pracnosti 10 člověkodnů tedy navyšují konečný odhad nákladů o 4 000 EUR.

Náklady na navržená zlepšení tak činí 25 200 EUR + 6 % z původního odhadu pracnosti projektů.

Celkové náklady na navržená zlepšení v případě referenčních projektů jsou prezentovány v tabulce 6 níže spolu s původními odhady.

Tabulka 6 Přehled nákladů na zlepšení pro referenční projekty, poměr k původním nákladům pracnosti a hranice přípustnosti

Název	Náklady původní pracnosti (EUR)	Celkové náklady na zlepšení (EUR)	% z plánovaných nákladů	Hranice přípustnosti (EUR)
Projekt 1	200 000	37 200	18,6	22 000
Projekt 2	200 000	37 200	18,6	22 000
Projekt 3	232 000	39 120	16,9	25 520

Zdroj: vlastní zpracování

Protože hranice přijatelnosti nákladů byla stanovena na 12 %, nelze zavést všechna navržená zlepšení, neboť jak je patrné z tabulky 6 výše, u všech referenčních projektů by výše potřebných nákladů tuto hodnotu překročila.

3.3.6 Výběr zlepšení a posouzení aplikovatelnosti

Za rozpočtového omezení daného hranicí přijatelnosti je proto nutné vybrat takovou množinu zlepšení, která přinese nejvyšší užitek.

Pro sestavení nákladově-efektivního seznamu použitelných zlepšení tak bylo nejprve hodnoceno, která z nich mají nejvyšší potenciál předcházet vzniku chyb zaznamenaných u referenčních projektů podle toho, zda řeší jejich kořenovou příčinu.

Pro každou kategorii kořenových příčin chyb (uvedených v podkapitole 3.2.1), seřazených sestupně od nejpočetnější, jsou níže uvedena použitelná zlepšení:

1. problém v dokumentaci:
   a) *revize připravenosti požadavků;*
   b) *revize akceptačních kritérií;*
   c) *revize detailního návrhu;*
2. chyba v kódu:
   a) *nastavení požadavků na jednotkové testy*;
   b) *průběžná kontrola výsledku statické analýzy kódu*;
   c) *doporučené praktiky vývoje*;
3. chyba v návrhu:
   a) *revize detailního návrhu*;
   b) *týmové posouzení navrženého řešení;*
4. chyba produktu:
   a) *požadavky na návrh testů*;
5. chybná oprava:
   a) *nastavení požadavků na jednotkové testy*;
   b) *doporučené praktiky vývoje*;
6. bezpečnostní problém:
   a) *revize detailního návrhu;*
7. chyba prostředí:
   a) *ověření splnění výstupních podmínek*.

Následně bylo posuzováno, která zlepšení se týkají příčin úniků chyb, takže jejich zavedení může zlepšit účinnost nacházení chyb před tím, než bude řešení předáno zákazníkovi. Pro každou skupinu příčin úniků, seřazených sestupně od nejpočetnější, jsou níže uvedena použitelná zlepšení:

1. nedostatek v testovacích případech nebo testovacích datech
   a) *požadavky na návrh testů;*
   b) *revize navržených testů a stavu exekuce;*
   c) *požadavky na konfirmační testování;*
2. netestovaný mimofunkční požadavek
   a) *testování mimofunkčních požadavků*;
   b) *požadavky na exploratorní testování*;
3. lidská chyba při exekuci testu
   a) *revize navržených testů a stavu exekuce*;
   b) *požadavky na exploratorní testování*.

Výše uvedené tak představuje minimální množinu navržených opatření, jež mohou být zavedena s celkovými náklady 20 600 EUR. Protože je tato částka pod hranicí přijatelnosti pro všechny tři referenční projekty, je možné seznam ještě rozšířit.

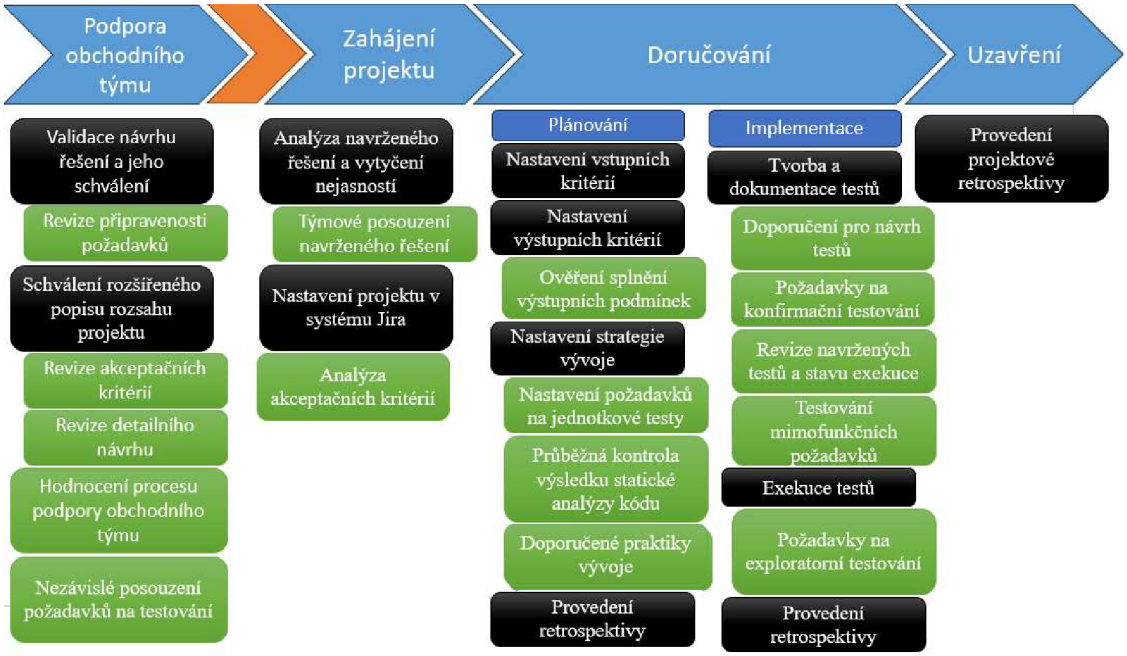
S ohledem na snahu o „posun vlevo“ je výhodné přidávat zlepšení z počátečních fází tak, aby jejich kombinace nepřekročila náklad 1 400 EUR. To splňují následující tři:

- *analýza akceptačních kritérií*;
- *posouzení testovatelnosti navrženého řešení*;
- *hodnocení procesu podpory obchodního týmu*.

Ačkoli je hranice přijatelnosti pro projekt 3 lehce vyšší než u projektů 1 a 2, nebudou pro něj plánována dodatečná zlepšení, aby byl navržený seznam aplikovatelný pro všechny z nich.

Výsledná množina zlepšení, které lze na referenčních projektech zavést za náklady nižší nebo rovné hranici přijatelnosti 22 000 EUR, tak obsahuje 16 prvků, zobrazených na obrázku 9 níže:

Obrázek 9 Konečná množina zlepšení v rámci DFW, vyznačených zeleně



Zdroj: vlastní zpracování

Oproti původnímu stavu, jež je zachycen na obrázku 7, je patrný větší důraz na včasné zahájení činností zajišťování a řízení kvality, tedy „posun vlevo“ diskutovaný výše.

3.3.7 Posouzení účinnosti zvolených zlepšení na referenčních projektech

Při posouzení účinnosti výsledné množiny navržených zlepšení bylo zjišťováno, zda úspora dosažená snížením počtu chyb převyšuje náklady nutné k jejich zavedení.

Navržená zlepšení řeší jednotlivé kořenové příčiny chyb i hlavní příčiny jejich úniku, v souladu s metodikou této práce tak bylo uvažováno, že aplikace zlepšení povede k redukci chyb o 2/3, tedy 66,6 %.

Při výpočtu tak byl původní počet chyb dle tabulky 1 v kapitole 3.2.1 (včetně chyb nahlášených po dodání) snížen na jednu třetinu, vypočtena pracnost v člověkodnech (MD) a potřebné náklady. Při výpočtu nákladů oprav bylo uvažováno s hodnotou 0,65 MD na jednu chybu, získanou z původních dat ve výše odkazované tabulce.

Následně byly náklady na opravy redukovaného počtu chyb odečteny od původních nákladů v tabulce 1, výsledek zaznamenán a odečten od nákladů na zavedení zlepšení o výši 22 000 EUR.

Data a jednotlivé vypočítané hodnoty pro každý ze tří referenčních projektů jsou uvedeny v tabulce níže.

Tabulka 7 Náklady na opravy redukovaného počtu chyb ve vztahu k nákladům na zlepšení

Název	Počet chyb (z toho po dodání)	Pracnost oprav (MD)	Náklady oprav (EUR)	Rozdíl proti původním nákladům (EUR)	Po odečtení nákladů na zlepšení	Úspora proti původním nákladům
Projekt 1	29 (3)	19	10 000	21 600	-400	-1,3 %
Projekt 2	37 (6)	24	14 400	26 800	4 800	11,7 %
Projekt 3	42 (4)	27	14 000	25 200	3 200	8,2 %

Zdroj: vlastní zpracování

Z tabulky je patrné, že zavedení dané množiny zlepšení s celkovými náklady 22 000 EUR a odhadovaným přínosem ve formě redukce počtu chyb na jednu třetinu je výhodné pro projekty 2 a 3, kde dojde ke konečné úspoře ve výši 4 800 a 3 200 EUR, a naopak nevýhodné pro projekt 1, kde je dosažená úspora nižší než náklady, jež je nutné vynaložit.

Výše uvedená data lze využít k vytvoření obecného pravidla, respektive doporučení, jak velký (ve smyslu pracnosti a tedy nákladů) by měl projekt být, aby bylo zavedení daných zlepšení ekonomicky výhodné.

Dle tabulky 1 v kapitole 3.2.1 lze uvažovat, že průměrný podíl nákladů na opravy k celkovým nákladům projektu je 16 %, přičemž aplikace zlepšení by měla vést k jejich snížení o přibližně 2/3. Jestliže tedy hodnota 2/3 nákladů na opravy (což představuje přibližně 10 % z celkových nákladů) přesáhne 22 000 EUR, je výhodné navrženou množinu zlepšení zavést. Pro snazší výpočty je vhodnější vypočítat celkovou výši nákladů na opravy, od které se zlepšení vyplatí, což je tedy 33 000 EUR.

Z těchto zjištění vyplývá, že bylo dosaženo cíle práce, neboť zavedení výsledné množiny 16 zlepšení vede u projektu splňujících stanovenou podmínku k úspoře prostřednictvím snížení počtu chyb.

Zjištěné výstupy, představované navrženými aktivitami nebo kroky soustředěnými primárně do úvodních fází projektu, mohou být využity pro rozšíření a úpravy firemního rámce DFW tak, aby mohly být aplikovány v praxi.

Závěr

V dnešním moderním světě, který je poháněn technologickým pokrokem, prostupuje software v nejrůznější podobě prakticky všechny aspekty života společnosti i jednotlivce, od soukromé komunikace, zábavy, vzdělání a obchodu přes infrastrukturu, zdravotnictví až po oblast národní bezpečnosti.

Rostoucí závislost na všudypřítomných softwarových systémech, ať už na mobilních telefonech, počítačích, v automobilech nebo samoobslužných pokladnách, klade stále vyšší nároky na kvalitu softwaru jako takového, neboť jeho bezchybný, bezpečný a efektivní provoz je nutným předpokladem k plynulému fungování moderní, digitální společnosti.

Vzhledem ke konkurenčnímu obchodnímu prostředí současného globalizovaného světa se kvalita dodávaného softwaru stává pro jeho výrobce zásadním faktorem, určujícím míru udržitelnosti a úspěchu na trhu. Investice do opatření souvisejících s kvalitou softwaru jsou tak strategickými rozhodnutími, jež ve výsledku přispívají ke zvyšování počtu spokojených, loajálních zákazníků, pozitivní reputaci dodavatele a konkurenční výhodě.

Zavádění takových opatření je ve své podstatě realizováno prostřednictvím proaktivních aktivit zajišťování kvality a reaktivních aktivit řízení kvality, jež jsou spjaty s přístupem, jakým organizace software vyvíjí, tedy modelem jeho životního cyklu.

V této bakalářské práci byl u vybrané společnosti zkoumán hybridní model životního cyklu vývoje softwaru a hodnoceny jeho aktivity zajišťování i řízení kvality za účelem identifikace nedostatků a navržení odpovídacích zlepšení. V tomto ohledu byl kladen důraz na to, aby byla navrhována zlepšení především proaktivní, což je ekonomicky nejvýhodnější. Cílem práce pak bylo navržení takové skupiny zlepšení, jejíž zavedení by za přijatelných nákladů vedlo ke snížení počtu chyb ve výsledném produktu a tedy úspoře vycházející z nižších nákladů nutných k jejich odstranění.

Praktická část práce se zabývá konceptem kvality obecně a v kontextu softwarových řešení, přičemž jsou uvedeny základní principy, definice a charakteristiky, které kvalitu softwaru tvoří. Zvlášť je pak diskutován ekonomický aspekt kvality softwaru ve formě nákladů, který se s ní pojí.

Dále byly zkoumány klíčové procesy zajišťování a řízení kvality, u nichž byly popsány hlavní činnosti, jimiž jsou prováděny. Závěr teoretické části se zabývá modely životního cyklu vývoje softwaru a jejich přístupu ke kvalitě.

V podkapitole Metody práce jsou uvedeny specifika zkoumaného podniku, metody výzkumu, zdroje dat a kritéria jejich výběru, metody zpracování práce, podmínky, omezení a předpoklady stanovené zástupci podniku pro vypracování práce.

Analyticko-praktická část práce se v prvním kroku zaměřuje na zjištění stávajícího stavu ve zkoumané oblasti v daném podniku. To sestávalo z rozboru specifického, na míru vytvořeného modelu životního cyklu, kombinujícího prvky tradičního a agilního vývoje. Jednotlivé prvky daného modelu byly hodnoceny s ohledem na obecné principy a doporučenou praxi.

Za tímto účelem bylo nutné porozumět používanému specifickému, na míru vytvořenému modelu životního cyklu vývoje softwaru, který podnik používá a zhodnotit jej s ohledem na doporučenou praxi a principy platné pro kvalitu softwaru zjištěné v teoretické části práce. Na základě tohoto šetření byl vytvořen seznam aktivit zajišťování a řízení kvality v podnikovém modelu, jehož součástí byly i identifikované nedostatky.

V rámci zjištění stávajícího stavu bylo provedeno také shromáždění informací o kvalitě na vzorku referenčních, již dokončených projektů, u kterých byly zaznamenány počty nalezených

chyb, jejich kořenové příčiny, členěné podle fáze vývoje a náklady vynaložené na jejich odstranění.

Druhým krokem pak bylo navržení takových opatření, zlepšení, která odstraní nebo zmírní nedostatky stávajících aktivit zajišťování a řízení kvality, tedy v ideálním případě odstraní nebo alespoň zmírní příslušnou kořenovou příčinu. Aby mohl být výsledek použitelný v praxi, odhadované náklady na zavedení zlepšení nesměly v součtu přesáhnout částku stanovenou jako hranice přijatelnosti. Z toho důvodu byla původně navržená množina redukována na takovou kombinaci zlepšení, jejichž zavedení mělo největší potenciál řešit kořenové příčiny chyb a jejich úniků za akceptovatelných nákladů.

Na závěr byla na jednotlivých referenčních projektech hodnocena návratnost investice do zavedení výsledné množiny navržených zlepšení, jejichž účinnost (ve smyslu předpokládané míry redukce chyb) byla odhadnuta zástupci podniku. Na základě těchto informací byly pro každý referenční projekt vypočítány náklady na odstranění zbývajícího počtu chyb a porovnáno, zda úspora proti původním nákladům na opravy převyšuje výši investice do navržených zlepšení.

Na základě výše uvedeného lze tedy konstatovat, že cíl práce byl splněn, neboť byla identifikována množina zlepšení sestávající z 16 prvků, jejíž zavedení na projektech splňujících stavěnou podmínku (na opravy chyb je plánováno alespoň 33 000 EUR) lze dosáhnout snížení počtu chyb a tedy úsporu v nákladech.

Navržená zlepšení mohou být začleněna do firemního modelu životního cyklu, jež je popsán prostřednictvím rámce DFW.

Literatura

Monografie

ADLAKHA-HUTCHEON, G. a MASYS, A. *Disruption, ideation and Innovation for Defence and Security*. Cham, Switzerland: Springer, 2022. 300 s. ISBN 9783031066368.

AMEY, S. *Software test design: Write comprehensive test plans to uncover critical bugs in web, desktop, and mobile apps*. Birmingham: Packt Publishing, 2022. 426 s. ISBN 9781804614730.

BAUMGARTNER, M., STEIRER, T., WENDLAND, M., GWIHS, S., SEIDL, R., HARTNE R, J. *Test Automation Fundamentals: A Study Guide for the Certified Test Automation Engineer Exam – Advanced Level Specialist – ISTQB® Compliant*. Paderborn: dpunkt.verlag, 2022. 330 s. ISBN 9783969108703.

BAUSCH, M. *Intercultural Transfer of Management Practices of German MNC to Brazil: The interplay of translation and recontextualization*. Wiesbaden: Springer Fachmedien Wiesbaden GmbH, 2022. 389 s. ISBN 9783658380564.

BIERIG, R., BROWN, S., GALVAN, E. a TIMONEY, J. *Essentials of software testing*. Cambridge: Cambridge University Press, 2022. 318 s. ISBN 9781108833349.

CASSY, B., HOLDERNESS, K., Olsen, K., SWAIN, M., YUSCHAK, M. a PHELAN, T. *Wiley CMA Exam Review Study Guide 2022*. Hoboken, NJ: John Wiley & Sons, Inc., 2022. 544 s. ISBN 9781119849407.

CHANDRASEKARA, C. a PUSHPA, H. *Hands-on Functional Test Automation: With visual studio 2017 and Selenium*. Berkeley, CA: Apress, 2019. 252 s. ISBN 9781000392777.

DELHAYE, J.-L. *Inside the world of Computing: Technologies, uses, Challenges*. Londýn: ISTE Ltd, 2021. 256 s. ISBN 9781786306654.

DENG, C. *Modern Intelligent Instruments: Theory and Application*. Singapore: Bentham Science Publishers, 2020. 291 s. ISBN 9789811460241.

FILIP, L. *Efektivní řízení kvality*. Praha: Pointa, 2019. 248 s. ISBN 9788090753051.

GALIN, D. *Software quality: Concepts and practice*. Hoboken, NJ: Wiley, 2018. 720 s. ISBN 978111913449.

HEATH, F. *The professional scrum master (PSM I) guide: Successfully practice scrum with real-world projects and achieve your PSM I certification with confidence*. Birmingham: Packt Publishing, 2021. 174 s. ISBN 9781800200494.

HILBURN, T. a TOWHIDNEJAD, M. *Software engineering practice: A case study approach*. Boca Raton: CRC Press, 2021. ISBN 9781466591677.

HUNDHAUSEN, R. a SCHWABER, K. *Professional scrum development with azure devops*. Redmond, WA: Microsoft Press, 2021. 356 s. ISBN 9780136789147.

JONES, C. *Software development patterns and antipatterns*. Boca Raton, FL: CRC Press, 2021. 512 s. ISBN 9781000414745.

JUN, W., LI, X., ALLISON, C. a HOHORST, J. *Nuclear power plant design and analysis codes: Development, validation, and application*. Duxford, United Kingdom: Woodhead Publishing, 2021. 608 s. ISBN 9780128181911.

KAMA, M., BASRI, S. a MAHMOOD, Y. *Software Requirement Change Effort Estimation: An Algorithmic Approach*. Kalkata: Exceller Books, 2020. 200 s. ISBN 9788194951773.

KHACHATRYAN, G. *Instruction modeling: developing and implementing blended learning programs*. New York, NY: Oxford University Press, 2020. 272 s. ISBN 9780190910709.

LAPLANTE, P. a KASSAB, M. *Requirements engineering for software and systems*. Boca Raton, FL: CRC Press, 2022. 428 s. ISBN 9781000593792.

LAPORTE, C. a APRIL, A. *Software quality assurance*. Hoboken, NJ: IEEE Press, 2018. 624 s. ISBN 9781118501825.

LAYTON, M., OSTERMILLER, S.J. a KYNASTON, D.J. *Agile Project Management for dummies*. Hoboken, NJ: Wiley, 2020. 419 s. ISBN 9781119677062.

LAYTON, M. *Scrum for dummies*. Hoboken, NJ: Wiley, 2018. 416 s. ISBN 9781119467649.

LEVIN, M.A., KALAI, T.T. a RODIN, J. *Improving product reliability and software quality: Strategies, tools, process and implementation*. Hoboken, NJ: Wiley, 2019. 456 s. ISBN 9781119179412.

LEWIS, W. *PDCA/Test*. Boca Raton FL: CRC Press, 2020. 448 s. ISBN 9781000170139.

MAKIELA, Z., STUSS, M. a BOROWIECKI, R. *Sustainability, Technology and Innovation 4.0*. Abingdon, Oxon: Routledge, Taylor & Francis Group, 2022. 360 s. ISBN 9781000439632.

O'REGAN, G. *Concise guide to software engineering: From Fundamentals to Application Methods*. Cham: Springer, 2022. 444 s. ISBN 9783031078163.

PANG, C. *Software engineering for agile application development*. Hershey, PA: Engineering Science Reference, 2020. 330 s. ISBN 9781799825333.

RAJIB, M. *Fundamentals of Software Engineering*. Delhi: PHI Learning Private Limited, 2018. 612 s. ISBN 9789388028035.

RANSOME, J. a SCHOENFIELD, B. *Building in security at Agile Speed*. Boca Raton, FL: CRC Press, 2021. 346 s. ISBN 9781000392777.

RICHARDS, R. *DSDM - agile project management – A (still)unknown alternative full of advantages: An Introduction to the AgilePM® Method, which Combines the Best of Classical Project Management and Agile Product Development*. Norderstedt: BOOKS ON DEMAND, 2021.112 s. ISBN 9783754396599.

ROSEN, C. *Guide to Software Systems Development connecting novel theory and current practice*. Cham: Springer International Publishing, 2020. 201 s. ISBN 9783030397302.

SHARMA, J. K. *Business statistics*. India: Vikas Publishing, 2019. 780 s. ISBN 9789353387273.

SPILLNER, A. a LINZ, T. *Software testing foundations;a study guide for the certified tester exam – foundation level – ISTQB compliant*. S.l.: DPUNKT VERLAG, 2021. 330 s. ISBN 9783969102985.

STAPP, L., ROMAN, A. a PILAETEN, M. *Istqb Certified Tester Foundation Level: A self – study guide syllabus v4.0*. S.l.: SPRINGER INTERNATIONAL PU, 2023. 406 s. ISBN 9783031427664.

STEPHENS, R. *Beginning Software Engineering*. Indianopolis, Indiana: Wiley, 2022. 720 s. ISBN 9781119901709.

THEISENS, H. *Lean six sigma black belt*. 's-Hertogenbosch.: Van Haren Publishing, 2020. 705 s. ISBN 9789401809788.

TOCKEY, S. *How to engineer software a model-based approach*. Hoboken, NJ: Wiley, 2019. 1168 s. ISBN 9781119546627.

TSIGKAS, A. *The Performative Enterprise: Ideas and Case Studies on Moving beyond the Quality Paradigm*. Cham: Springer, 2021. 248 s. ISBN 9783030814915.

VALLABHANENI, S. *Wiley CIA Exam Review focus notes 2021*. Hoboken: John Wiley & Sons, Inc., 2021. 1104 s. ISBN 9781119753469.

WALSH, K., SATHIADEV, M. a TRUMBACH, C. *Agile scrum implementation and its long-term impact on organizations*. Hershey, PA: IGI Global, 2021. 268 s. ISBN 9781799848868.

WIEGERS, K. *Software Development Pearls: Lessons from Fifty Years of Software Experience*. [*s.l.*] : Addison-Wesley Professional, 2021. 336 s. ISBN 9780137487806.

WITTE, F. *Strategy, planning and organization of test processes basis for successful project execution in software testing*. Wiesbaden: Springer Fachmedien, Imprint: Springer, 2022. 256 s. ISBN 9783658369811.

Sborníky

KOHEN, A., KAPOOR, S. a BHATIA, R. *Proceedings of the future technologies conference (FTC) 2020, volume 3*. Cham: Springer International Publishing, 2021, s. 361-365. ISBN 9783030630928.

Technické normy

IEEE 730. *Standard for Software Quality Assurance Processes*. New York: IEEE, 2014. ISBN 978-0-7381-9168-3.

Přílohy

Řešená problematika

VŠEM

úvod

Dnešní moderní společnost je úzce provázána s technologiemi, jejichž kvalita je tak stále důležitější. V globálním konkurenčním prostředí přináší investice do kvality výrobcům zlepšení reputace, zvýšení spokojenosti a loajality zákazníků a tedy konkurenční výhodu.

problém

Vybraná společnost používá při dodávkách softwarových řešení vlastní model životního cyklu vývoje softwaru, přičemž se potýká s vyššími počty chyb a tedy dodatečnými náklady na jejich odstranění.

přístup

Revizí stávajícího modelu a rozborem nejčastějších příčin chyb bylo cílem navrhnout taková zlepšení, jejichž zavedení by za přijatelných nákladů zvýšilo kvalitu výsledného softwarového produktu.

Postup řešení

zdroj

Pro teoretickou část byly využita dostupná literatura a technické normy. V analyticko-praktické části byly zkoumány firemní systémy (SAP PPM, Jira, Confluence), metodické dokumenty a dokumenty referenčních projektů.

získávání

Srovnáváním literatury byla zjištěna doporučení v různých aspektech kvality softwaru. Z firemního modelu a dat referenčních projektů byla získána data o současném stavu – aktivitách souvisejících s kvalitou, počtech chyb, nákladech na jejich odstranění.

zpracování

Byly identifikovány nedostatky současného modelu, časté kořenové příčiny chyb a jejich pozdního odhalení, což sloužilo jako výchozí bod pro návrh vhodných opatření/zlepšení. Dále byla vybraná množina takových zlepšení, aby jejich zavedení splnilo rozpočtové omezení a zároveň vedlo ke snížení chyb.

Výsledky práce

→Z výsledků práce vyplynulo, že podnik využívá primárně reaktivní přístup ke kvalitě, pro než byly navrženy změny tak, aby byl více proaktivní a preventivní.

→ Cíle bylo dosaženo. Navržená zlepšení reflektují potřebu včasné kontroly vstupů, poskytování zpětné vazby a nastavení procesů pro vyšší účinnost.

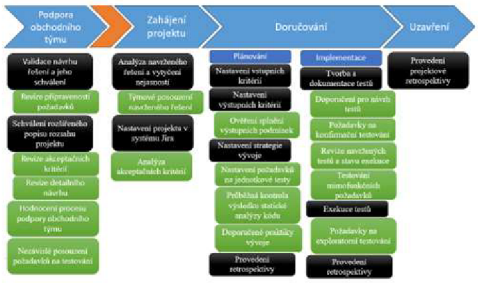
→**Z dat lze vyčíst fakta:**

→ konečná množina navržených zlepšení obsahuje 16 prvků s náklady na zavedení ve výši 22 000 EUR a tedy splňující hranici přijatelnosti.

→U dvou ze tří referenčních projektů vedla jejich aplikace (na základě odhadu účinnosti poskytnuté zástupci podniku) k redukci nákladů, a to o 11,7 % a 8,2 %.

→Investice bude výhodná u projektů, kde je na opravy chyb plánováno nejméně 33 000 EUR.

Výsledky práce – grafické znázornění



Zdroj: vlastní na základě podnikového modelu

Doporučení

Na základě výsledků lze doporučit.....

1. Firma bude profitovat ze zavedení navržených zlepšení na těch projektech, kde je plánováno alokovat nejméně 33 000 EUR na opravy chyb.

2. Zaměření na aktivity proaktivního a preventivního přístupu ke kvalitě umožňuje její zvyšování a následné snižování nákladů.

3. Z ekonomického hlediska znamená zavedení navržených zlepšení úsporu zejména na velkých projektech, kde jsou náklady na opravy chyb násobně vyšší než výše potřebné investice.

Závěr

Práce přinesla posouzení přístupu ke kvalitě softwaru u vybrané společnosti a navržení zlepšení zjištěných nedostatků za účelem snížení počtu chyb a souvisejících nákladů.

Novým řešením je úprava podnikového modelu životního cyklu vývoje softwaru tak, aby bylo ke kvalitě přistupováno více proaktivně a preventivně.

Problematika byla posunuta díky aplikaci doporučených principů zajišťování a řízení kvality do prostředí vybrané organizace.