Citace článku

BUCHALCEVOVÁ, Alena, KUČERA, Jan. Hodnocení metodik vývoje informačních systémů z pohledu testování. Systémová integrace, 2008, roč. 15, č. 2, s. 42–54. ISSN 1210-9479

Hodnocení metodik vývoje informačních systémů z pohledu testování

Alena Buchalcevová, Jan Kučera

Katedra informačních technologií VŠE Praha
nám. W.Churchilla 4, Praha 3

Abstrakt

Kvalita informačních systémů závisí na kvalitě softwarových produktů, které jsou jejich součástí. Kvalita softwaru je ovlivněna řadou faktorů, mezi které patří proces jejich vývoje. Tento článek se zabývá porovnáním procesů testování softwaru ve vybraných metodikách - Extrémní programování, Rational Unified Process a MSF for Agile Software Development.

Úvod

Jednou z charakteristik dnešní doby je rychlý vývoj v oblasti technologií, přičemž vývoj informačních a komunikačních technologií (ICT) patří k nejdynamičtějším. Informační systémy pomáhají podnikům uspět v současných tržních podmínkách, využít nabízených možností a čelit potencionálním hrozbám. Základem informačních systémů je software a kvalita softwaru silně ovlivňuje celkovou kvalitu systému. Vady softwaru mohou vést k selhání informačních systémů, což může mít za následek i značné škody. Z tohoto důvodu by softwarové produkty měly být testovány. Testování softwaru je proces, při kterém dochází k ověření, zda softwarový produkt odpovídá definovaným požadavkům. Za tímto účelem je software podroben sérii testů, kterými je softwarový produkt prověřen na několika úrovních, od jednotlivých programových komponent až po programový systém jako celek. Testování softwaru není samostatný proces, ale je součástí procesu vývoje softwaru. Proces vývoje softwaru popisují metodiky vývoje softwaru, které definují principy, procesy, praktiky, role, techniky, nástroje a produkty používané při vývoji, a to jak z hlediska softwarově inženýrského, tak z hlediska řízení [BUCH]. Metodik existuje velké množství a liší se zaměřením, rozsahem, oblastí, pro kterou jsou určeny, přístupem k řešení a v neposlední řadě také úrovní podrobnosti, na které jsou popsané jednotlivé postupy, procesy, doporučení, techniky a další složky metodiky. Rozdíly mezi metodikami nalezneme i v přístupu k testování softwaru, který při jeho vývoji uplatňují. Proces testování by sám měl být kvalitní, aby podával relevantní údaje o dosažené kvalitě produktu. Cílem tohoto článku je porovnat přístup k testování softwaru v několika vybraných metodikách vývoje softwaru a zhodnotit přínosy a případné slabé stránky těchto přístupů pro vývoj softwaru.

Kritéria hodnocení metodik z pohledu testování

Abychom mohli porovnat, jak jednotlivé metodiky přistupují k testování softwaru, definujeme následující sadu kritérií, která jsou dále vysvětlena:

- zaměření testování,
- proces testování,
- role,
- úrovně testování,
- trasovatelnost,
- životní cyklus vady,
- typy testů a testovací techniky,
- artefakty testování,

- užití nástrojů a automatizace.

Zaměření testování

Činnosti spojené s testováním provádějí většinou specializované testovací role, ale mohou se na nich podílet i další role. Z tohoto pohledu rozlišujeme testování závislé a nezávislé. Závislé testování realizují role, které mají jinou specializaci než testování. Příkladem může být vývojářské testování, kdy například programátoři provádějí jednotkové testování. Nezávislé testování provádějí role, které testují výsledek práce jiných rolí (proto nezávislé). Jedná se zejména o role, jejichž specializací je testování, ale může jít i o zástupce zákazníka provádějící akceptační nebo beta testování.

Proces testování

Procesem testování rozumíme činnosti související s testováním, jejich sled a vzájemné vztahy. Toto kritérium hodnotí fáze životního cyklu projektu, do kterých je testování zapojeno, a přístup k tvorbě testů (tvorba testů souběžně s implementací či Test-first Design).

Role

V rámci tohoto kritéria se hodnotí role, které se na testování podílejí. Jde jednak o role specializované na testování a jednak role, které se na testování podílejí.

Úrovně testování

Úroveň testování je definována jako sada testů, kterými je softwarový produkt testován na definované úrovni podrobnosti. Rozeznáváme jednotkové testování, které testuje nejmenší části systému, integrační testování, které se zaměřuje na testování vazeb a propojení částí systému, systémové testování, tedy testování systému jako celku, a akceptační testování, jehož cílem je prokázat splnění definovaných akceptačních kritérií. V rámci tohoto kritéria se zjišťuje, které úrovně testování metodika pokrývá, jaké role se podílejí na testování na dané úrovni a jaká je intenzita testování.

Trasovatelnost

Trasovatelnost umožňuje identifikovat vzájemné vztahy mezi položkami dokumentace a softwaru. Toto kritérium omezuje trasovatelnost na vztah mezi požadavky a testy. Pokud testy vznikají v návaznosti na definované požadavky, potom je metodika hodnocena tak, že trasovatelnost zajišťuje.

Životní cyklus vady

Životní cyklus vady začíná jejím odhalením a končí jejím vyřešením (odstraněním či zdokumentováním postupu, jak se vyhnout selhání). Toto kritérium hodnotí, zda v metodice existuje definovaný životní cyklus vady a jakým způsobem se rozhoduje o řešení nalezených vad.

Typy testů a testovací techniky

Testy kategorizujeme zpravidla podle oblastí kvalitativních ukazatelů, které testují. Rozeznáváme testy funkčnosti, použitelnosti, výkonu, spolehlivosti a podpory. Mezi testy funkčnosti patří testy ověřující správnou implementaci požadovaných funkcí systému nebo bezpečnost systému. Testy použitelnosti se zaměřují na uživatelské rozhraní a jeho přívětivost, dále na oblast nápovědy, dokumentace a kvalitu výukových materiálů. Výkonnostní testy ověřují správný a stabilní průběh funkcí systému v situacích, kdy systém musí čelit velké zátěži. Testy spolehlivosti testují schopnosti systému zvládat nestandardní chování, chybné vstupy nebo možnosti obnovy po havárii systému. Testy podpory pokrývají oblasti jako je možnost nasazení na různých platformách, možnosti údržby a rozšíření systém nebo možnosti konfigurace systému. Zvláštním typem testu je smoke test. Dle [ISTQB] se jedná o realizaci vybrané podmnožiny vytvořených nebo plánovaných testů s cílem ověřit správnou funkčnost všech kritických částí systému. Mezi testovací techniky patří například průzkumné testování nebo beta testování.

Artefakty testování

Součástí tohoto kritéria je zhodnocení, jaké dokumenty ve srovnání se standardem IEEE 829 [IEEE] metodika pokrývá. Standard IEEE 829 definuje následují dokumenty:

- Test Plan – dokument popisující, co bude testováno, jaké k tomu budou použity testy, techniky a nástroje, jakým způsobem bude testování probíhat, jaké jsou požadavky na kvalitu a jaký je harmonogram testování.
- Test Design Specification – dokument, který na základě dalších podkladů jako je například návrh systému nebo specifikace produktu shrnuje a blíže vymezuje, co má být otestováno a jak bude posuzována úspěšnost jednotlivých testů.
- Test Case Specification – popis testovacího případu zahrnující vstupní a výstupní data, vstupní a výstupní hodnoty a jednotlivé kroky testovacího případu.
- Test Procedure Specification – dokument popisující jednotlivé kroky při provádění testovacího případu.
- Test Item Transmittal Report – dokument, který doprovází předávání nové verze produktu k testování. Obsahuje popis změn oproti minulé verzi a je zde vymezeno, co má být otestováno v současné verzi.
- Test Log – dokument, ve kterém je zachycen průběh testovacích případů a jejich výsledků.
- Test Incident Report – dokument, který obsahuje popsané pozorované neshody, kroky a okolnosti, za kterých k nim došlo a další doplňující informace.
- Test Summary Report – souhrnná zpráva o průběhu celého testování a zjištěné úrovni kvality testovaného systému.

Užití nástrojů a automatizace

Automatizací testování rozumíme použití softwaru k podpoře nebo provádění činností spojených s testováním, jako je řízení testování, provádění a vyhodnocování testovacích případů a scénářů. Z tohoto hlediska se u metodiky hodnotí, zda existuje definovaný přístup k automatizaci testování a jaké oblasti testování metodika doporučuje automatizovat. Praktiky některých metodik přímo vyžadují, aby testování bylo automatizované, jinak je jejich realizace obtížná.

Extrémní programování

Extrémní programování (XP) je metodika určená zejména pro malé až středně velké týmy (2 – 10 programátorů), které vyvíjejí software, jehož zadání není jasné a nebo se mění. Autory metodiky jsou Kent Beck, Ward Cunningham a Ron Jeffries [BECK].

Zaměření testování

XP se zabývá zejména závislým testováním, které je reprezentováno testováním programátorů. Do kategorie nezávislého testování spadá akceptační testování, které je v této metodice hlavní zodpovědností samotných uživatelů, kteří by sami měli navrhovat akceptační testy.

Proces testování

Testování je součástí všech fází životního cyklu projektu. Testeři se podílejí na odhadu pracnosti definovaných uživatelských příběhů. V rámci fází analýzy, návrhu a implementace probíhá testování průběžně a velmi často. Ve fázi zavádění jsou provedeny akceptační testy. Metodika vyžaduje přístup Test-first Design, testy pro každý požadavek musí být vytvořeny před tím, než je implementován. Tímto způsobem jsou tvořeny jak unit testy, tak i testy funkčnosti. Pro testování programátorů je přístup TFD podrobně popsán. Každá možná cesta programem by měla mít v ideální případě svůj test. Programátor by měl postupovat tak, že implementuje jednoduchý test, který musí selhat, protože ještě neexistuje kód programu, který je testován. Poté je implementován co nejjednodušší kód programu, který úspěšně projde testem. Následujícím krokem je refaktorizace, jejímž cílem je čitelný a znovupoužitelný program. Po každém kroku refaktorizace je test znovu spuštěn.

Role

Testování a kvalita je podle XP odpovědností každého člena týmu, tudíž i všech rolí. Nicméně hlavními rolemi, kterých se testování přímo týká, jsou role zákazník, vývojář a tester. Zákazníci by měli tvořit testy funkčnosti k ověření jimi definovaných uživatelských příběhů. Vývojáři tvoří, provádějí a vyhodnocují unit testy. XP definuje pouze roli tester, nerozvádí ji do dalších

specializovaných rolí. Testeři plní roli poradců, kteří oběma výše zmíněným rolím pomáhají se sestavováním testů, jejich automatizací a tvorbou testovacího prostředí. Pomáhají s odhadem rizik, prověřují uživatelské příběhy, napomáhají definovat požadavky na kvalitu. Dohlížejí také na to, že testy funkčnosti skutečně ověřují definované uživatelské příběhy.

Úrovně testování

Metodika XP specifikuje dvě skupiny testů: unit testy a testy funkčnosti. Unit testy slouží k testování jednotek. Při jejich vytváření je třeba brát v úvahu i interakci jednotky s okolím. Unit testy definované podle metodiky XP plní roli jednotkových testů a integračních testů dle úrovní testování. Tyto testy jsou převážně tvořeny a spouštěny vývojáři. Metodika doporučuje provádět toto testování velmi často, pokud možno neustále. Testy funkčnosti slouží k ověření splnění uživatelského příběhu. Testován je systém od vstupu po výstup. V rámci iterace by sada testů funkčnosti nebo její podmnožina měla být spouštěna alespoň jednou denně, zde tyto testy plní úlohu systémových testů. Na konci iterace při předávání produktu zákazníkovi slouží tyto testy k prokázání, že veškerý dohodnutý rozsah byl implementován. Plní tedy úlohu akceptačních testů. Tvorba testů funkčnosti je společnou zodpovědností zákazníků a testera. Tester pak provádí vytvořené testy a vyhodnocuje jejich průběh.

Trasovatelnost

Ke každému uživatelskému příběhu musí existovat jeden či více testů funkčnosti k ověření jeho splnění. Testy vznikají ještě před napsáním programového kódu. Vývojář při rozhovoru se zákazníkem získá podrobnosti k jednotlivým uživatelským příběhům a rozhodne o způsobu jejich realizace. Poté následuje tvorba příslušných unit testů. Vazby mezi požadavky a testy vznikají v průběhu vývoje a jejich vznik je dán navrženým procesem. Tvorba dokumentu, který by trasovatelnost zachytil ale není zmiňována.

Životní cyklus vady

Životní cyklus vady není popsán, nakládání s odhalenou vadou je popsáno pouze v souvislosti s tvorbou testů. Po objevení vady, by měl být vytvořen test, který ověří její odstranění. Poté by vada měla být odstraněna. Způsob rozhodování o odstraňování vad je pouze nastíněn v souvislosti s výhodami častého testování. Jednou z výhod častého testování je, že vady jsou objeveny záhy poté, co je dokončena nová část aplikace nebo je provedena změna. Pro programátory se jedná o čerstvou záležitost, lokalizovat a odstranit vadu by tedy mělo být snazší.

Typy testů a testovací techniky

Metodika neuvádí žádné konkrétní typy testů nebo testovací techniky. Můžeme nalézt články, které se zabývají užitím testovacích technik nebo typů testů v rámci XP. Příkladem může být [JEFF], který se věnuje beta testování.

Artefakty testování

XP jako agilní metodika se příliš nezaměřuje na dokumentaci. Důsledkem toho je, že nedefinuje žádné povinné artefakty testování kromě samotných testů a jejich výsledků. Dokumentace testování neodpovídá standardu IEEE 829.

Užití nástrojů a automatizace

Velmi krátké iterace, neustálé spouštění jednotkových testů a integrace, časté spouštění sady testů funkcionality jsou faktory, které vedou k nutnosti testování automatizovat. Extrémní programování není přímo spojeno s žádným robustním komerčním testovacím či vývojovým nástrojem. často se ale používají nástroje volně distribuované, mezi nimi lze nalézt dva, na jejichž tvorbě se podíleli autoři této metodiky. Jedná se o nástroje xUnit a Fit. xUnit je rodina testovacích nástrojů pro provádění unit testů tak, jak je definuje Extrémní programování. X v názvu nástroje značí programovací jazyk nebo platformu, pro kterou je určena daná implementace tohoto přístupu k testování - SUnit pro Smalltalk, JUnit pro jazyk Java nebo CUnit pro jazyk C. Fit je jednoduchý nástroj na podporu systémových nebo akceptačních testů jehož autorem je Ward Cunningham. Fit je volně šířen a je možné ho získat na internetových stránkách tohoto nástroje (viz. [FIT]). Existuje v několika verzích pro různé vývojové platformy. K tomuto nástroji existují také nadstavby, které rozšiřují jeho možnosti a usnadňují jeho

používání. Jedním z těchto rozšíření je FitNesse. Jedná se implementaci webového serveru, kam mohou členové týmu přidávat nové stránky s testy a vytvářet tak sady testů a testovací scénáře.

Rational Unified Process

Metodika Rational Unified Process (RUP) je rámec pro definování metodiky pro celé spektrum projektů od malých až po velmi velké. Dříve byla řazena mezi rigorózní metodiky, ale v současné době zahrnuje agilní principy a praktiky a je možné ji přizpůsobit i agilnímu vývoji [RUP]. Metodika RUP je šířena na komerční bázi a je podporována nástroji IBM Rational.

Zaměření testování

Tato metodika se zabývá závislým i nezávislým testováním. Závislé testování provádějí vývojáři. Nezávislé testování má na starosti testovací tým nebo v případě beta testování samotní uživatelé. Pro závislé i nezávislé testování jsou zpracovány koncepty, které popisují obecná doporučení a upozorňují na možné problémy.

Proces testování

Testování prochází všemi fázemi životního cyklu projektu, přičemž množství práce a prostředků věnovaných testování zpravidla postupně narůstá, maxima je dosaženo ve fázi konstrukce. Celý proces testování je rozdělen do několika kroků, z nichž některé probíhají paralelně a opakují se v každé iteraci.

Prvním krokem testovacího cyklu je určení cílů testování. Následuje ověření, že aktuální sestava je stabilní. Pokud není, o čemž svědčí nepřiměřený výskyt vad, hlavně těch velmi závažných, je tato sestava vrácena zpět do vývoje. Pokud je sestava stabilní, je podrobena testování a ohodnocení. Paralelně s touto činností probíhá příprava zprávy o průběhu a výsledcích testování určená manažerům projektu a dalším zainteresovaným osobám. Po dokončení testování a vytvoření výkazů následuje proces údržby a zkvalitnění testovací dokumentace, testovacích dat, vytvořených testů a dalších náležitostí testování (v metodice souhrnně označených jako Test Assets). Cílem je udržovat náležitosti aktuální a tvořit z nich základnu, jejíž prvky lze znovupoužít. Činnosti od ověření stability sestavy po zkvalitnění náležitostí testování mohou v rámci iterace proběhnout vícekrát, jedná se o tzv. testovací cyklus. Paralelně s testovacím cyklem běží proces, jehož úkolem je ověřovat, že zvolený přístup k testování je vhodný, testy jsou kvalitní a podávají relevantní výsledky.

Metodika doporučuje užít přístup TFD, je ale třeba zvážit, kdy je vhodný čas testy navrhnout. Není účelem navrhnout všechny testy co nejdříve, je třeba postupovat v souladu s cíli iterace a fáze projektu.

Role

V disciplíně testování vymezuje metodika RUP čtyři role: manažer testování, test analytik, návrhář testů a tester. Pro každou roli metodika vymezuje činnosti, které vykonává a artefakty, se kterými pracuje. Dále také uvádí doporučení pro personální obsazení jednotlivých rolí. Manažer testování zastřešuje práci celého testovacího týmu. Jeho úkolem je řídit proces testování a řešit problémy s ním spojené. Je zodpovědný za plánování zdrojů a měl by v celém řešitelském týmu vystupovat jako advokát kvality. Úkolem test analytika je identifikovat a popsat potřebné testy, sledovat průběh procesu testování a hodnotit dosaženou úroveň kvality zjištěnou během testování. Je také zodpovědný za vhodnou volbu reprezentace výsledků testování. Návrhář testů je zodpovědný za definici přístupu k testování a jeho úspěšnou realizaci, což zahrnuje identifikaci vhodných postupů, nástrojů a doporučení pro provádění testů a pro odhad zdrojové náročnosti testování. Tester je zodpovědný za tvorbu testů, jejich provádění, zaznamenání průběhu a jejich vyhodnocení. Testování není zodpovědností pouze těchto rolí, ale probíhá také v rámci vývojářského týmu. Vývojářské testy jsou zodpovědností role vývojář (implementer). Za integraci systému a provádění integračních testů je zodpovědná role integrátor. Ve fázi předávání řídí realizaci akceptačního a beta testování manažer nasazení (Deployment Manager). V této fázi jsou do testování zapojeni také zástupci zákazníka. U akceptačních testů rozhodují o přijetí či nepřijetí vyvinutého produktu. Během beta testování systém přímo testují.

Úrovně testování

Metodika RUP pokrývá všechny úrovně testování. Jednotkové a integrační testy by měli provádět vývojáři, kteří se také mohou podílet na systémových testech. Testovací tým by měl provádět hlavně testy systémové a akceptační. Jednotkové testy se provádějí podle potřeby, typicky několikrát během iterace. Integrační testy jsou rozděleny na testy podsystémů a testy systému jako celku. Testy podsystému by měly proběhnout alespoň jednou během iterace, testy celého systému jsou vázány na tvorbu sestavy. Pro tvorbu sestav metodika neurčuje žádné specifické načasování. Systémové testy by měly proběhnout několikrát během iterace, akceptační testy jsou pak prováděny v závěrečné fázi projektu, kdy dochází k předávání hotového řešení zákazníkovi.

Trasovatelnost

Testovací případy a scénáře jsou v metodice RUP odvozené od případů užití systému. Model případů užití slouží k zachycení požadavků na systém a je odsouhlasen zákazníkem. Definice konkrétního přístupu k zajištění trasovatelnosti je součástí testovacího plánu. Popis testovacích případů a scénářů je doplněn o odkazy na požadavky, které jsou jimi testovány. Trasovatelnost je v této metodice zajištěna a dokumentována.

Životní cyklus vady

Řešení nalezené vady je předmětem změnového řízení, kdy je požadavek na odstranění vady jedním z typů požadavků na změnu. Životní cyklus změnového požadavku je popsán. Vada je označena prioritou a jsou odhadnuty její dopady. V rámci změnového řízení je rozhodnuto o způsobu jejího řešení, přiřazení příslušných kapacit a způsobu, jakým bude odstranění vady ověřeno.

Typy testů a testovací techniky

Metodika RUP se zmiňuje o použití smoke testu a beta testování. Smoke test slouží k ověření stability aktuální sestavy produktu. Beta testování je doporučováno provést ve fázi zavádění, kdy je produkt nasazen do zkušebního provozu. Zjištěné problémy tak mohou být odstraněny nebo alespoň zdokumentovány před tím, než je produkt nasazen do běžného provozu. Typy testů jsou popsány a u některých z nich jsou zpracována doporučení, jak je provádět, kdy přistoupit k jejich automatizaci a které nástroje je k tomu vhodné využít. Jejich volba závisí na konkrétním projektu.

Artefakty testování

Metodika RUP definuje celou řadu artefaktů testování. U každého vymezuje, jeho účel, náplň a zodpovědnou roli. Nejvýznamnější artefakty testování jsou Test Plan, Test Evaluation Summary (ekvivalentní dokumentu Test Summary Report), Test Script (odpovídá dokumentu Test Procedure Specification z IEEE 829), Test Log, Test Case (odpovídá dokumentu Test Case Specification z IEEE 829), Test Results, Test Strategy (ekvivalentní k dokumentu Test Design Specification). Nejsou pokryty dokumenty Test Item Transmittal Report a Test Incident Report. Metodika uchovává a spravuje výsledky testů, samotnou zprávu o neshodách nevyhotovuje.

Užití nástrojů a automatizace

Pro automatizaci testování je v metodice RUP obsažen obecný koncept, který popisuje výhody a nevýhody automatizace a možné problémy a rizika jeho použití. Metodika zdůrazňuje nutnost lidského úsudku při testování. Hlavní oblastí pro využití automatizace testování jsou vývojářské testy, hlavně testy jednotek. Další doporučení pro využití automatizace lze nalézt u popisu typů testů. Automatizace je tedy hodnocena jako doporučená.

Nástroje IBM Rational tvoří široké portfolio, které svým rozsahem pokrývá celý cyklus vývoje softwaru. Metodika RUP obsahuje rádce pro práci s nástroji (Tool Mentors), které popisují užití jednotlivých nástrojů při konkrétních činnostech popsaných v RUP. Nástroj IBM Rational PurifyPlus pro analýzu chodu aplikací spíše než testeři využijí vývojáři při řešení problémů spojených se spolehlivostí a výkonem aplikací. IBM Rational ClearQuest je komplexní nástroj pro podporu, kontrolu a automatizaci vývojového cyklu. Umožňuje definici a automatizaci pracovního toku (workflow), tvorbu hlášení, kalkulaci metrik, automatizaci tvorby sestavy aplikace a jejího nasazení. Tento nástroj slouží také jako nástroj pro řízení testování – jeho plánování, přípravu, provádění a oznamování výsledků. IBM Rational Manual Tester je nástroj na podporu manuálního testování.

Umožňuje vytvářet a spravovat testovací případy a scénáře pro manuální testy. Nástroj IBM Rational Functional Tester je určen pro automatizaci testování funkčnosti. Testovací scripty pro automatizované provádění testů je možné vytvořit záznamem interakce s testovaným systémem nebo je vytvářet a upravovat v jednom z podporovaných programovacích jazyků. IBM Rational Performance Tester je specializovaným nástrojem pro provádění zátěžových a výkonnostních testů. Nástroj IBM Rational AppScan je určen pro testování zabezpečení webových aplikací. IBM Rational Policy Tester je nástroj pro testování webových aplikací a existuje ve třech verzích. Verze Accessibility Edition se specializuje na testování dostupnosti, především na plnění standardů, zákonných nařízení a doporučení pro tvorbu webových aplikací. Privacy Edition je specializovaná na ochranu osobních údajů uživatelů. Quality Edition se zaměřuje na problémy jako jsou nefunkční odkazy mezi stránkami, gramatické chyby v obsahu stránek nebo jejich pomalé načítání a další. Všechny tyto nástroje fungují tak, že procházejí stránky testované webové aplikace a provádějí definované testy. Ze získaných výsledků je pak sestavena zpráva. IBM Rational Robot je nástroj pro automatizaci testů pro klient-server aplikace. Tento nástroj umožňuje tvorbu testovacích scriptů záznamem interakce s grafickým uživatelským rozhraním aplikace.

MSF for Agile Software Development

Microsoft Solutions Framework (MSF) je obecný rámec, na jehož základě lze vytvořit konkrétní metodiku, která odpovídá potřebám určitého projektu nebo skupiny projektů. Zaměřuje se jak na oblast softwarově inženýrskou, tak na řízení projektu a je podporován vývojovou platformou Microsoft Visual Studio Team System. MSF nabízí dvě předefinované metodiky - MSF for Agile Software Development a MSF for CMMI Process Improvement [MAGP].

Zaměření testování

MSF for Agile Software Development se zabývá oblastí závislého i nezávislého testování. U každé role jsou definovány činnosti spojené s testováním, které jsou detailněji rozpracovány do jednotlivých kroků, u nichž jsou k dispozici doporučení a rady pro jejich provádění. Testovací činnosti a kroky jsou u obou kategorií testování popsány stejně podrobně.

Proces testování

Testování se zapojuje do životního cyklu ve fázi analýzy a návrhu. Testy jsou tvořeny před kódováním nebo souběžně s ním. Metodika doporučuje vyčlenit samostatné iterace na řešení a odstraňování chyb, pokud přibývají výrazně vyšším tempem, než se je daří odstraňovat v rámci běžného běhu iterace.

Role

Ze skupiny testovacích rolí je v této metodice definována pouze role tester. Úkolem testera je objevit vady v produktu a další problémy, které mohou ovlivnit výslednou kvalitu a hodnotu pro zákazníka, určit jejich dopady a navrhnout postupy, jak tyto dopady zmírnit. Podílí se také na plánování iterace, kde plní konzultační úlohu. Na testování se podílí i další role. Vývojáři tvoří, provádějí a vyhodnocují testy jednotek a integrační testy. Databázoví vývojáři také testují pomocí testů jednotek. Ve fázi nasazení přebírá odpovědnost za testování manažer nasazení (release manager), tester přechází do role konzultanta. Vývojáři jsou také zodpovědní za provádění testů při ověřování, zda-li aktuální sestava produktu, splňuje minimální stanovené požadavky na kvalitu.

Úrovně testování

Jednotkové testy metodika dělí do dvou skupin. První skupinou jsou jednotkové testy aplikace, za jejichž tvorbu, provádění a vyhodnocení jsou zodpovědní vývojáři. Druhou skupinu tvoří testy jednotek databáze, za které jsou zodpovědní databázoví vývojáři. Oba druhy jednotkových testů se provádějí průběžně vždy, když vývojáři dokončí jim přidělené úkoly nebo dojde ke změně ve vyvíjeném systému. Po dokončení úkolů vývojářů následuje integrace a integrační testování. Jako systémové testy můžeme chápat testy scénářů a testy kvality služby, obě skupiny testují celý systém od vstupů až po výstupy. Testy scénářů ověřují splnění funkčních požadavků, testy kvality služby testují nefunkční požadavky, respektive výkonnost, bezpečnost, zátěž a výkon při omezených zdrojích. Akceptační testy jako takové nejsou metodikou definovány. Před nasazením produktu u zákazníka je vytvořený produkt otestován, aby bylo ověřeno, že jsou splněny všechny specifikované požadavky a

všechny známé problémy jsou řešeny (buď byly odstraněny, nebo alespoň existuje popsaný návod, jak se jim vyhnout). Zákazník ale není přítomen, po provedení výše popsaného testování je produkt nasazen.

Trasovatelnost

Vztah mezi požadavky a testy není jasný a není zachycen ani v rámci popisů jednotlivých činností ani v dokumentaci testů nebo požadavků není vyznačena jejich souvislost.

Životní cyklus vady

Životní cyklus vady je popsán. Nové vady ohlašují testeři a vývojáři, chyba se tak dostává do stavu aktivní. Manažer projektu přiřazuje chybám prioritu z hlediska jejich řešení, o které se starají vývojáři. Chyba přechází do stavu vyřešena. Odstranění chyby je ověřeno testery a životní cyklus chyby je uzavřen.

Typy testů a testovací techniky

V rámci testování požadavků na kvalitu služby jsou prováděny následující typy testů: výkonnostní testy, bezpečnostní testy, zátěžové testy a stress testy (testy chování aplikace s minimem nebo nedostatkem výpočetních zdrojů). Zvlášť jsou vymezeny webové testy, které ověřují webové stránky a komunikaci prostřednictvím internetu. Průzkumné testování je doporučená technika testování jak pro testování funkčních požadavků tak i požadavků nefunkčních. Základem pro provádění tohoto testování je definice vzorových uživatelů, tzv. personas. Jejich chování je během průzkumného testování simulováno. Po integraci produktu je doporučeno provést smoke test.

Artefakty testování

Metodika nepodporuje vytváření rozsáhlé testovací dokumentace. Ze standardu IEEE 829 pokrývá pouze dokument Test Plan, kterému odpovídá dokument nazývaný Test Approach (přístup k testování).

Užití nástrojů a automatizace testování

Metodika doporučuje, aby stanovený přístup k testování obsahoval jak testy prováděné manuálně tak testy automatizované. Automatizovány by měly být ty testy, u kterých je to podle situace vhodné a možné. Bližší určení přístupu k automatizaci metodika neobsahuje. Automatizace je tedy hodnocena jako doporučená.

Propojení mezi metodikami založenými na Microsoft Solutions Framework a sadou nástrojů Microsoft Visual Studio Team System (VSTS) je velmi silné. Aplikovat principy MSF ve vývoji je možné i bez VSTS, nicméně již způsob distribuce těchto metodik napovídá, že největší efekt přináší jejich užití ve spojení právě s těmito nástroji. Metodiky MSF Agile a MSF CMMI jsou distribuovány jako procesní šablony pro VSTS. To znamená, že definují, jaké činnosti jsou jednotlivými členy týmu prováděny, jakou podobu mají vstupy a výstupy těchto činností, jaké pracovní položky budou používány a jaké zprávy a reporty bude mít vedení projektu k dispozici. Procesní šablony lze podle potřeby upravovat a definovat tak vlastní proces. Kromě procesních šablon pro MSF jsou k dispozici i šablony pro procesy a metodiky třetích stran, například pro agilní metodiku SCRUM. Testerům je určena samostatná verze Visual Studia, která obsahuje nástroje pro tvorbu, spouštění a správu testů. Testování není v MSF záležitostí pouze testerů, proto nástroje pro jednotkové testování jsou obsaženy také ve verzích pro vývojáře a databázové specialisty. Nástroje pro zjišťování pokrytí kódu testy jsou společné verzím pro vývojáře a pro testery. Verze pro testery podporuje jednotkové testování, zátěžové testování a testování webových aplikací. Popsané testové případy, které mají být prováděné manuálně je možno uložit jako manuální test. Manuální testy nemají žádnou zvláštní podporu, jedná se pouze o textové dokumenty s popsanými kroky testovacího scénáře, které jsou uloženy ve společném úložišti a k jejich obsahu lze přistupovat z prostředí Visual Studia.

Závěr

Tabulka 1 obsahuje v přehledné formě porovnání jednotlivých metodik. Obecně pro všechny hodnocené metodiky platí, že zapojují testování již do počátečních fází projektu. Každá z porovnávaných metodik definuje specializované testovací role a kromě nich do procesu testování

zapojuje i role další. Společný je i přístup ke kvalitě jako zodpovědnosti všech členů týmu. Ze všech porovnávaných metodik se svým přístupem k testování nejvíce odlišuje metodika Extrémní programování. U ostatních metodik tvoří testování v rámci celého procesu vývoje samostatnou oblast. V metodice XP je testování velice úzce provázáno s ostatními praktikami a zajišťuje jejich plynulý a efektivní průběh. XP se snaží o co největší automatizaci testování, za jakýsi ideální stav je pak považována kompletní automatizace. Ostatní metodiky v této oblasti zdůrazňují, že ne vždy lze automatizovat vše a také, že lidský úsudek nelze ničím nahradit. Proto doporučují vhodně kombinovat manuální a automatizované testování.

	XP	**RUP**	**MSF Agile**
Zaměření testování	Závislé testování	Závislé i nezávislé testování	Závislé i nezávislé testování
Proces testování			
fáze	ÚS, AN, IMP, ZAV	ÚS, AN, IMP, ZAV	AN, IMP, ZAV
tvorba testů	Vyžadovaný TFD	Doporučený TFD	Souběžně s implementací nebo TFD
Role			
specializace testování	Tester	Manažer testování Test analytik Návrhář testů Tester	Tester – řídící, prováděcí, konzultační
další zúčastněné role	Vývojář, zákazník	Vývojář, integrátor, manager nasazení	Vývojář, databázový vývojář, manažer nasazení
Úrovně testování			
jednotkové	Vývojář; probíhá kontinuálně	Vývojář; dle potřeby	Vývojář, databázový vývojář; dokončený úkol
integrační	Vývojář; probíhá kontinuálně	Integrátor; alespoň 1x během iterace	Vývojář; dokončený úkol
systémové	Tester; alespoň 1x denně	Testovací tým, (vývojáři); 1/sestava	Tester; 1/sestava
akceptační	Tester, uživatel; předávání na konci iterace	Testovací tým, manager nasazení; předávání	Velmi nejasné
Trasovatelnost	Zajištěna, nedokumentována	Zajištěna, dokumentována	Nezajištěna
Životní cyklus vady	Životní cyklu explicitně nedefinován, vady odstraňovat co nejdříve	Životní cyklus definován, vady předmětem změnového řízení	Životní cyklus definován, vady řešeny dle priority stanovené vedoucím projektu
Typy testů, praktiky	Přístup nedefinován	Smoke test, beta testování, typy testů vymezeny – užití dle potřeby	Průzkumné testování, smoke test. Doporučené typy testů
Nástroje a automatizace	Automatizace nezbytná, nástroje pouze obecné	Automatizace doporučená, nástroje přímo určené	Automatizace doporučená, nástroje přímo určené
Artefakty testování	Nedefinuje, IEEE 829 neodpovídá	Vyhovují IEEE 829, nezahrnuje Item Trasmittal Report a Test Incident Report	IEEE 829 – pouze Test Plan

Tabulka 1 Porovnání metodik z hlediska testování

XP vyžaduje přístup Test-first Design, metodika RUP tento přístup doporučuje užít po zvážení, kdy a pro jaké testy je přístup TFD vhodný. Metodiky založené na MSF uvádějí užití tohoto přístupu jako

možnost. Nejobsáhlejší testovací dokumentace je definována v metodice RUP, svým rozsahem pokrývá většinu standardu pro testovací dokumentaci IEEE 829. Na rozdíl od ostatních tato metodika také ověřuje, že zvolený přístup k testování odpovídá potřebám projektu a že je možné ho s dostupnými zdroji realizovat. Celkově lze říci, že v hodnocených metodikách je testování důležitou součástí procesu vývoje. Jednotlivé metodiky se ani tak neliší obecnými principy jako spíše v jejich konkrétní aplikaci.

Použité zdroje

[ASQ1] American Society for Quality: Quality Assurance and Quality Control. Dostupný na WWW: http://www.asq.org/learn-about-quality/quality-assurance-quality-control/overview/overview.html [27.1.2008]

[ASQ2] American Society for Quality: Glossary. Dostupný na WWW: http://www.asq.org/glossary/q.html [1.2.2008]

[BECK] Beck, K.: Extrémní programování, Grada, 2002, ISBN 80-247-0300-9

[BUCH] BUCHALCEVOVÁ, A.: Metodiky vývoje a údržby informačních systémů. 1. vyd. Praha, GRADA 2005. 164 s. ISBN: 80-247-1075-7

[FIT] Fit: Framework for Integated Test, stránky testovacího nástroje. Dostupné na WWW: http://fit.c2.com/ [14.3.2008]

[FNES] FitNesse: stránky testovacího nástroje. Dostupné na WWW: http://www.fitnesse.org/ [14.3.200]

[IEEE] Institute of Electrical and Electronics Engineers: ANSI/IEEE 829-1983 IEEE Standard for Software Test Documentation –Description, nedatováno. Dostupný na WWW: http://standards.ieee.org/reading/ieee/std_public/description/se/829-1983_desc.html [16.3.2008]

[ISTQB] International Software Testing Qualification Board – Glossary working Party, ed. Veenendaal, E. van: Standard glossary of terms used in Software Testing. Version 1.3, 31.5.2007, 39 s.

[JEFF] JEFFRIES, R.: Beta Testing in XP, XP Magazine, 4.9.2003. Dostupný na WWW: http://www.xprogramming.com/xpmag/jatBetaTesting.htm [13.3.2008]

[KADL] KADLEC, V.: Agilní programování. 1. vyd. Brno, Computer Press 2004. 280 s. ISBN: 80-251-0342-0

[MAGP] Microsoft Corporation: MSF for Agile Software Development Process Giudance, verze 4.1.0. Dostupný na WWW: http://msdn2.microsoft.com/en-us/teamsystem/aa718795.aspx [13.3.2008]

[RUP] IBM Corporation: Rational Unified Process version 2003.06.15

[SEPT] Software Engineering Process Technology: Software Engineering Standards, nedatováno. Dostupný na WWW: http://www.12207.com/test.htm [14.3.2008]

[PATT] PATTON, R.: Testování softwaru. 1. vyd. Praha, Computer Press 2002. 328 s. ISBN: 80-7226-636-5