Zabezpečenie kvality softvérových produktov

VIKTOR BACHRATÝ

Slovenská technická univerzita
Fakulta informatiky a informačných technológií
Ilkovičova 3, 842 16 Bratislava
viktor.bachraty@gmail.com

Abstrakt. Hlavným cieľom eseje je porovnať rôzne metódy vývoja softvéru z hľadiska zabezpečenia kvality. V prvej časti približuje základné vlastnosti procesov ako Rational Unified Process (RUP), Microsoft Solution Framework (MSF) a Extreme Programming (XP). Ďalej poskytuje stručný prehľad odporúčaní štandardu zabezpečenia kvality softvéru v jadrových aplikáciách - ASME NQA-2a-1990. Podrobnejšie sa analyzuje dôraz na kvalitu ako aj zabezpečenie kvality od počiatočnej fázy životného cyklu softvéru až po umiestnenie softvéru u zákazníka, technickú podporu a údržbu. Hľadá spoločné vlastnosti týchto procesov a snahou je odhalenie výhod jednotlivých prístupov. Z toho vyplýva diskusia o možnostiach vylepšenia jednotlivých modelov ako aj posúdenia, za akých kritérií možno považovať konkrétny model za vhodnejší.

Úvod

V súčasnej dobe rýchleho rozvoja IT sa dostáva vývoj trhu na takú úroveň, že už nie je postačujúce ak firma poskytuje iba produkt, ktorý je funkčný. Na trhu sa vytvára konkurencia a zákazník si vyberá dodávateľa z viacerých možností. Preto sa dodávatelia snažia rôznymi spôsobmi získať si zákazníkov. Už nestačí, ak zákazníkovi povieme, že náš produkt je kvalitný, ale potrebujeme to aj nejako demonštrovať. Jednou z ciest je získanie certifikátov, ktoré potvrdzujú, že pracovné metódy vo firme zabezpečujú dosiahnutie najvyššej možnej kvality. K tomu je potrebné aby sa kvalita dala nejako ohodnotiť alebo aspoň porovnať s inými produktmi podobného určenia.

V prvom rade, aby bolo možné kvalitu ohodnotiť, je potrebné si zadefinovať, čo pod pojmom kvalita rozumieme. Závisí to okrem iného aj od toho, z koho pohľadu kvalitu hodnotíme. Iný je pohľad vývojára a iný je pohľad používateľa. Kvalita sa všeobecnosti definuje ako miera uspokojenia potrieb zákazníka. Podľa normy ISO to

je súhrn vlastností a charakteristík výrobku procesu alebo služby, ktoré preukazujú jeho schopnosť splniť určené potreby [1]. Keďže vo väčšine prípadov je zákazník používateľom, bude pre nás tento pohľad dôležitý.

V softvérovom inžinierstve je v posledných rokoch veľa snáh o vývoj takých metodológií a techník ktoré umožňujú zabezpečiť, aby výsledný produkt bol najvyššej možnej kvality. Ďalšou z oblastí výskumu je snaha nájsť metódy, ako kvalitu výsledného produktu zmerať. V prípade softvéru to nie je také jednoznačné ako napríklad vo výrobe hmotných produktov. Pri nich je pomerne jednoduché zmerať fyzikálne vlastnosti ako sú rozmery, teplotná odolnosť a mnohé ďalšie podľa typu a použitia produktu. V prípade softvéru je to omnoho zložitejšie. Môžeme napríklad spočítať, koľko chýb bolo nájdených v danom programe, alebo koľko ich bolo opravených. Avšak toto nám neumožňuje odhadnúť množstvo skrytých chýb, ktoré vo význačnej miere môžu ovplyvniť spokojnosť zákazníka.

Pre najlepší možný výsledok je preto vhodné využívať oba prístupy. Dôležité je aby proces vývoja softvéru bol taký, ktorý sám o sebe zabezpečí že vznikne kvalitný produkt a okrem toho musíme zabezpečiť používanie techník, ktoré nám umožnia zmerať kvalitu produktu a tak odhaliť možné nedostatky.

Najrozšírenejšími modelmi vývoja softvéru sú Rational Unified Process (RUP), Microsoft Solution Framework (MSF) a Extreme programming (XP). Cieľom tejto eseje je porovnať ich možnosti, čo sa týka zabezpečenia kvality. Snahou je nájsť spoločné črty a odhaliť výhody a prípadné nedostatky. V ďalšej časti si priblížime aké sú požiadavky normy pre vývoj jadrových aplikácií - ASME NQA-2a-1990 (ASME). Na základe tejto analýzy potom je možné dať všeobecné odporúčania, ktoré je vhodné dodržať, aby výsledný produkt spĺňal aj požiadavky najnáročnejších zákazníkov.

Zabezpečenie kvality v softvérových projektoch

Na zabezpečenie kvality v softvérovom projekte je potrebné, aby boli splnené viaceré podmienky. Okrem kvalifikovaných a šikovných ľudí pracujúcich na vývoji je potrebné, aby vývoj prebiehal podľa dobre fungujúceho procesu vývoja softvéru. Každý proces vývoja softvéru obsahuje kroky ako analýza, návrh, implementácia a testovanie. Tieto kroky zabezpečia, aby bola vykonaná správna špecifikácia požiadaviek a aby implementovaný produkt spĺňal požiadavky. Niektoré modely zahrňujú aj umiestnenie u zákazníka a údržbu. Mnohé dokonca zahrňujú aktivity, ktoré zabezpečia, aby projekt bol ukončený v rámci stanoveného rozpočtu a časového limitu, čo tiež zvyšuje mieru spokojnosti zákazníka teda aj kvalitu. Avšak väčšinou explicitne nedefinujú zabezpečenie kvality ako súčasť modelu. Typické modely zabezpečenia kvality sú definované ako vedľajšie procesy k procesu vývoja softvéru čo spôsobuje niekoľko nevýhod.

1. Zabezpečenie kvality je definované ako úloha oddelenia pre zabezpečenie kvality. Kvôli rozpočtovým aj personálnym nedostatkom, hlavne v malých firmách toto nie je realizovateľné

2. Používajú sa metódy na zisťovanie chýb, ktoré už v produkte sú. Táto forma zabezpečenia kvality je príliš nákladná v porovnaní s prevenciou.

Kvalita môže byť zabezpečená pomocou troch techník – testovanie, statická analýza a metóda vývoja. Najlepší prístup je práve v kombinácii týchto troch techník. Žiaľ zatiaľ ani odborníci nedospeli ku konsenzu, akým spôsobom tieto tri techniky integrovať [4]. Na úrovni procesov sa odporúčajú štandardy ako Total Quality Management (TQM), ISO 9000 alebo Capability Maturity Model (CMM).

Rational Unified Process

Tento model bol vyvinutý na základe skúseností zo stoviek softvérových projektov a postrehov najvplyvnejších ľudí v softvérovom inžinierstve. RUP je jediný proces, ktorý je dostupný aj ako produkt a stal sa štandardom vo svojej oblasti. Je najpoužívanejším z porovnávanej trojice najrozšírenejších modelov. Jednou z jeho kľúčových výhod je stála podpora od firmy Rational Software, ktorá neustále tento proces vylepšuje a prispôsobuje meniacim sa požiadavkám. Okrem toho neustále prispôsobuje aj nástroje, ktoré sa používajú. RUP má dobre štruktúrovanú kostru (framework), ktorá je rozdelená do viacerých fáz a pracovných postupov. Okrem toho je ľahko pochopiteľná a umožňuje jednoduché plánovanie zdrojov a štruktúrovanie práce. Do procesu sú začlenené minimálne štandardy ako vyhodnocovanie požiadaviek, iteratívny vývoj softvéru, ako aj testovanie a spolupráca so zákazníkom počas celého procesu.

Microsoft Solution Framework

MSF je vlastne dôkladným popisom procesov a praktík, ktoré prebiehajú počas vývoja softvéru vo firme Microsoft. Reprezentuje pokus firmy Microsoft rozšíriť svoje know-how. MSF je jedným z dvoch komplementárnych frameworkov, druhý je Microsoft Operations Framework, ktoré spolu tvoria kompletné riešenie pre veľké softvérové spoločnosti, ktoré pracujú na stredných až veľkých projektoch. MSF nie je závislé na MOF, takže ho možno aplikovať samostatne. MSF kladie dôraz na vytváraní vysokokvalitného softvéru, pričom jedným zo základných princípov je iteratívne pridávanie funkcionality do produktu a systém práce „tím rovnocenných“ t.j. prístup bez dominantného vedúceho tímu. Oba princípy vylepšujú nie len kvalitu výsledného produktu ale aj proces vývoja. Microsoft poskytuje aj návrhové vzory, ktorých používanie MSF odporúča. Keďže MSF neobsahuje veľké množstvo nástrojov, je ľahko integrovateľné aj do existujúcich prostredí.

Extreme programming

Extreme programming je úplne nový prístup k vývoju softvéru. Je najrozšírenejšou agilnou metódou. Pre menšie projekty XP umožňuje veľmi dobrý prístup k dosiahnutiu

vysokej kvality. Tesná spolupráca so zákazníkom, veľký dôraz na testovanie a snaha znížiť námahu vynaloženú na návrh sú dobrým základom k dosiahnutiu cieľa. Nevýhodou je, že je problematické riadenie v prípade väčších projektov ako aj tesná spolupráca so zákazníkom nie je vždy možná. XP odporúča časté vydávanie malých upravených verzií. Výhodou tohto prístupu je, že pridaná funkcionalita sa veľmi rýchlo zavedie a tak je možné aj skoršie otestovanie. Okrem toho je spätná väzba od zákazníka prichádza skôr, teda je menšia pravdepodobnosť márne vynaloženého úsilia. Rýchla reakcia je aj z opačnej strany, ak sa zmenia požiadavky zákazníka, produkt sa dá prispôsobiť bez zbytočne veľkých zmien. Celý systém sa navrhuje tak, aby bol čo najjednoduchší, teda aby neobsahoval nepotrebnú funkcionalitu. Táto technika je známa pod menom KISS – (Keep It Simple, Stupid).

Softvér pre jadrové aplikácie

Norma ASME NQA-2a-1990 (ASME-NQA2) nie je vývojovým procesným modelom ako predchádzajúce. Je to len zoznam odporúčaní od United States Nuclear Regulatory Commission, ktoré sa majú dodržať pri vývoji softvéru pre jadrové elektrárne. Sú zaujímavé práve z toho dôvodu že v týchto aplikáciách nesmú vznikať chyby, pretože následky by boli katastrofálne. Preto možno považovať takýto druh aplikácií za vrchol možností, čo sa týka zabezpečenia kvality. Vychádza z normy IEEE 7300, ktorá sa zameriava na verifikáciu a validáciu softvéru. V samotnej norme sa píše, že mnohé jej odporúčania sú generické, a preto je ich možné použiť aj v inej oblasti, kde je spoľahlivosť softvérových systémov kritická.

ASME-NQA2 kladie dôraz na dokonalé zdokumentovanie všetkých funkcií a testov, ktoré boli na softvéri vykonané. Dokumentácia slúži ako nástroj na komunikáciu medzi členmi tímu, bližšie špecifikuje zodpovednosti za jednotlivé úlohy[3]. Umožňuje komunikáciu medzi vývojovým tímom aj revíznymi technikmi, ktorí môžu posúdiť podľa nej ako sa bude projekt vyvíjať, a či je kladený dostatočný dôraz na jednotlivé úlohy. Revízny technik môže dodatočne overiť, či vykonané testy spĺňajú požadované kritériá, aké chyby boli nájdené a ako boli odstránené, či produkt spĺňa špecifikácie.

Podľa ACME-NQA2, môže byť použitý ľubovoľný model životného cyklu softvéru, avšak musí obsahovať aktivity ktoré odporúča model IEEE1012. Určuje, ktoré revízie sa majú vykonať a ako sa majú zdokumentovať. Identifikuje sa, ktoré merania sa majú vykonať, kedy a kto ich vykonáva, prečo sa vykonávajú. Takáto dokumentácia napríklad môže odhaliť zvýšenie chybovosti v čase keď je projekt v časovej tiesni. Na základe toho sa vykonajú ďalšie kontrolné opatrenia. K softvéru sa musí dodávať manuál zdrojového kódu, ktorý obsahuje aké praktiky a štandardy boli dodržiavané pri písaní kódu.

Návrhári softvérových systémov pre nukleárne aplikácie musia včleniť do systému procesy, ktoré sú schopné identifikovať zlyhania softvéru a situáciu napraviť[2]. Zváženie rizík v takýchto typoch aplikácií nie je postačujúce, bezpečnosť systému treba systematicky a neustále ohodnocovať a nezávisle verifikovať. Aj

po inštalácii je softvér periodicky testovaný, či už ako celok alebo jednotlivé jeho komponenty. Frekvencia testovaní závisí od operačnej histórie a záznamov o zlyhaniach, ako aj relatívnej dôležitosti danej komponenty vzhľadom k spoľahlivosti celkového systému.

Vlastnosti softvérových procesných modelov

V nasledujúcom odstavci sú opísané jednotlivé spoločné znaky horeuvedených procesných modelov vývoja softvéru, ktoré sa týkajú zabezpečenia kvality. Tieto znaky boli získané na základe analýzy modelov MSF, RUP, XP smerom zdola nahor. Aby boli vymenované vlastnosti prakticky použiteľné, uprednostnené sú empirické prístupy analýzy. Aby bolo možné tieto procesné modely porovnať zo skúseností vývojárov, je potrebné získať dostatočný počet vývojárov, ktorý majú skúsenosti so všetkými tromi modelmi, čo je prakticky nemožné. Inou možnosťou je previesť kontrolované experimenty, kde tá istá skupina vývojárov bude vyvíjať podobné projekty pomocou rôznych procesných modelov. Avšak toto tiež nie je jednoducho uskutočniteľné pretože jednak by experiment trval príliš dlho na projektoch reálneho rozsahu, a navyše by to bolo príliš nákladné.

Iteratívny vývoj softvéru

Aby bol výsledný projekt kvalitný, musí postupovať pomocou iteratívneho a inkrementálneho prístupu. Iteratívny vývoj softvéru je charakterizovaný ako neustále zlepšovanie vlastností a opätovné definovanie požiadaviek vo viacerých iteračných cykloch. Základom je prototyp, ktorý sa vytvorí počas prvej iterácie a potom sa neustále modifikuje a vylepšuje. Každý iteračný cyklus obsahuje všetky fázy procesného modelu – t.j. analýzu, návrh, implementáciu, testovanie a aj uvedenie do prevádzky. V takomto procese je jednoduchšie zabezpečiť kvalitu. Vďaka iteratívnemu prístupu je projekt flexibilnejší, je jednoduchšie prispôsobiť produkt meniacim sa požiadavkám. Vývojári dostávajú skorú spätnú väzbu od zákazníka, vďaka čomu môžu veľmi rýchlo sa prispôsobiť jeho požiadavkám. Avšak tento prístup musí byť podporovaný manažmentom rizík a musí byť do neho zapojený aj zákazník. Na veľmi prísnych iteráciách je založené XP, ktoré vyžaduje každodenné vytvorenie aktuálnej verzie všetkých komponentov. Vďaka tomu sa skráti čas, kým sa skryté chyby môžu objaviť a núti vývojárov ich čím skôr vyriešiť. Jedinou nevýhodou je potreba dôkladne projekt naplánovať a rozložiť ho na malé komponenty, aby boli každým dňom integrovateľné nové funkcie.

Kvalita ako cieľ

Aby sa kládol na kvalitu dostatočný dôraz, treba ju zadefinovať ako jeden z cieľov. Tento cieľ musí byť zadefinovaný a zdokumentovaný zákazníkom spoločne s projektovým tímom. Týmto spôsobom sa zaručí, že ciele kvality sa budú dať dosiahnuť a aj merať. MSF definuje tieto ciele hneď na začiatku a ich naplnenie je jedným z hlavných bodov projektu.

Priebežná verifikácia kvality

Je dôležité aby sa každá zmena v projekte náležite zdokumentovala. Priebežné verifikovanie kvality obnáša extenzívne testovanie. Okrem interného testovania sú dôležité aj externé akceptačné testy u zákazníka, aby sa overilo splnenie jeho požiadaviek. Kvôli tomu musí každý softvérový proces obsahovať workflow testovania počas celého procesu, vrátane testovania u koncových zákazníkov, aby sa zabezpečila vysoká kvalita.

Požiadavky zákazníka

Potreby a želania zákazníka, ktorý nemá hlboké technické znalosti, musia byť zistené a dobre zdokumentované, aby vývojári mohli podľa nich spraviť aplikáciu. Preto je potrebné správne pochopiť zákazníka a jeho biznis. Inak nie je možné správne implementovať jeho požiadavky. Počas celého vývoja projektu je potrebné sa zamerať na požiadavky zákazníka, nie je to postačujúce len v začiatočných fázach. Ďalej je dôležité, aby do procesu boli zapojený aj používatelia softvéru, pretože úspech produktu závisí nielen od jeho predaja, ale aj následnej použiteľnosti. Z toho dôvodu musí obsahovať aj postupy ako zaškoliť budúcich používateľov systému.

Návrh závislý od architektúry

V súčasnosti sú stále väčšie tlaky na kvalitu a na cenu produktu. Preto je veľmi dôležité navrhnúť takú architektúru, aby boli komponenty znova použiteľné. Ďalším dôvodom je integrácia do už existujúceho prostredia. Vhodne navrhnutá architektúra umožňuje ľahkú integráciu ako aj znovupoužiteľnosť kódu.

Dôraz na tím

Tím je skupina rovnocenných ľudí, ktorí nesú kolektívnu zodpovednosť za kvalitu a úspech projektu. Ak sa dá zodpovednosť určiť na jednu osobu, nie je zaručené, že projekt bude úspešný. Zameranie sa na tímovú prácu tiež zvyšuje ich motiváciu členov, pretože každý sa cíti ako rovnako dôležitá súčasť projektu. V konečnom dôsledku to vedie k stotožneniu sa členov tímu s produktom. Je zrejmé, že dobre motivovaný člen tímu pracuje oveľa dôslednejšie a koncentrovanejšie. Musia byť dobre zadefinované úlohy v tíme, ako aj spôsoby komunikácie medzi členmi. Tím rovnocenných v MSF priraďuje každej roli rovnakú dôležitosť. Tím má niekoľko cieľov ako celok a možno to považovať za vynikajúcu implementáciu tímovej spolupráce.

Programovanie vo dvojiciach

Je to tiež spôsob tímovej spolupráce, v minulosti bol dosť podceňovaný. Je to demonštrácia toho, ako sa dvaja vývojári navzájom dopĺňajú namiesto toho aby sa spomaľovali. Jeden aktuálne implementuje nejakú funkciu, kým ďalší rozmýšľa nad problémom z globálneho hľadiska. Tento prístup šetrí čas aj znižuje množstvo chýb.

Je veľmi pravdepodobné, že vzniknú lepšie riešenia, pretože každý z vývojárov má pravdepodobne iný pohľad na vec a vždy vyberú tú lepšiu z možnosti, ktoré ich napadnú.

Spoločné vlastnosti modelov

Podpora kvality v MSF

Ako ukazuje tabuľka, MSF poskytuje najviac nástrojov podporujúcich zabezpečenie kvality. Kvalita je formálne definovaná ako cieľ, čo odlišuje tento proces od ostatných. Ďalšou výhodou tohto modelu je spomínaný dôraz na tím – tím rovnocenných (team of peers). Podobne ako RUP, aj tento proces na začiatku vo fáze „envisioning" definuje požiadavky zákazníka a podľa nich sa načrtne kandidát na architektúru a prototyp. MSF môže byť prispôsobované špecifickým požiadavkám, avšak jadro ostáva nezmenené čo zabezpečuje dodržanie úrovne kvality. Ako v jedinom softvérovo procesnom modeli je definovaný manažment rizík.

Kritérium	MSF	RUP	XP
Iteratívny vývoj	X	X	X
Kvalita ako cieľ	X		
Priebežná verifikácia kvality	X	X	X
Požiadavky zákazníka	X	X	X
Návrh závislý na architektúre	X	X	X
Dôraz na tím	X	X	X
Programovanie vo dvojiciach			X
Prispôsobenie obmedzeniam	X	X	
Proces manažmentu softvéru		X	
Manažment rizík	X		

Tabuľka 1. techniky podporujúce kvalitu

Podpora kvality v RUP

Kvalita nie je explicitne definovaná ako v modeli MSF, napriek tomu však nemožno povedať, že by tým stratila na význame. V každom cykle sa kvalita priebežne kontroluje a model obsahuje dobre definovanú revíziu na konci iterácie. Táto revízia overuje, či boli splnené ciele, a ak nie, tak odhaľuje príčiny, ktoré potom pomôžu pri plánovaní ďalšieho cyklu. RUP má veľmi sofistikovaný prístup k prispôsobovaniu. Samotný proces obsahuje nástroj ktorý to umožňuje. V rámci siete Rational developer

network sa nachádzajú rôzne prispôsobené implementácie RUP, prispôsobovanie nie je obmedzené. Manažment rizík je viacej prepracovaný ako v XP, ale nie natoľko ako v MSF. Manažment softvéru (angl. Configuration Management) je jedna z kľúčových predností RUP, zahŕňa aj spoluprácu investorov do projektu.

Podpora kvality v XP

Extrémne programovanie nemá veľmi podrobne implementované zabezpečenie kvality. Je to spôsobené tým, že XP nie je natoľko formalizované ako ostatné modely. Navyše XP sa zameriava na menšie projekty, kde nie je natoľko potrebné implementovať manažment rizík a manažment softvéru. Tento prístup umožňuje rýchly a jednoduchý vývoj softvéru. Kvalita vzniká ako výsledok dôsledného prístupu k vývoju. Aj vo veľkých projektoch by bolo však rozumné implementovať niektoré črty z XP. Vďaka tesnej spolupráci zákazníka, ktorý je začlenený aj do plánovania iterácii, má zákazník väčšiu možnosť zasahovať do zabezpečenia kvality. Krátke iterácie nútia vývojárov aby robili funkčné vydania, ktoré prejdú akceptačnými testami. XP používa takzvané „používateľské príhody“ na zachytenie požiadaviek, ale nedefinuje manažment požiadaviek ak proces. XP nemožno veľmi prispôsobovať nakoľko obsahuje len minimálne potrebné množstvo riadenia. Tímová spolupráca a programovanie vo dvojiciach je veľmi efektívny prístup a je to jedna z kľúčových výhod extrémneho programovania.

Záver

Porovnanie všetkých troch modelov procesu vývoja softvéru ukázalo, že všetky obsahujú nejaké metódy na zabezpečenie kvality. Tieto techniky zabezpečujú dostatočnú kvalitu v bežných prípadoch, takže nie je nutná ďalšia kontrola zo strany nejakého nezávislého oddelenia pre zabezpečenie kvality.

Výnimkou sú aplikácie pre jadrové elektrárne. Tam musí byť zabezpečená maximálna kvalita, preto sa takéto systémy pravidelne testujú nezávislými revíznymi technikmi aj po zavedení do prevádzky. Tieto systémy musia navyše mať prepracované scenáre možného zlyhania systému a analýzu možných dopadov.. Kladie sa aj zvýšený dôraz na dokumentáciu. Takéto zabezpečenie kvality je však príliš nákladné pre bežné systémy, preto sa používa len tam, kde je to nutné.

MSF a RUP sú najprepracovanejšie softvérové procesy, čo sa týka podpory zabezpečenia kvality. XP definuje menej techník na zabezpečenie kvality, avšak v tomto prípade kvalita vyplýva implicitne z princípov, ktoré XP využíva. XP je špecializované pre menšie až stredné projekty, kde svojou kvalitou nezaostáva za ostatnými modelmi.

Každý z týchto procesov má svoje výhody, ktoré by bolo možné integrovať aj do ostatných procesov. XP používa programovanie vo dvojiciach, MSF definuje kvalitu ako jeden z cieľov. Existujú aj snahy o vytvorenie nejakého generického modelu, ktorý by spájal výhody všetkých troch prístupov [4].

Niektoré techniky sú použité vo všetkých troch modeloch, preto ich možno považovať za de-facto štandard. Sú to nasledovné:

- Iteratívny vývoj
- Priebežná verifikácia kvality
- Dôraz na požiadavky zákazníka
- Dôraz na tímovú spoluprácu

Odporúča sa, aby bolo všetkých päť techník zahrnutých v každom softvérovom procese v každej firme. Vhodné by bolo aj keby sa aj ďalej vyvíjali a nakoniec by mohli z nich vniknúť oficiálne štandardy, napr. ISO.

Použitá literatúra

1. Bieliková, M.: Manažment v Softvérovom Inžinierstve, 1999
2. Sparkman, D.R., Lagdon, R.: *Software Quality Assurance for Nuclear Safety Systems,* International System Safety Conference , August 2 – 4, 2004
3. Wallace, D.R., Peng, W.W., Ippolito, L.M.: *Software Quality Assurance: Documentation and Reviews*, In Publication NISTIR 4909, U.S. Department of Commerce, National Institute of Standards and Technology, http://hissa.nist.gov/publications/nistir4909/
4. Zuser W., Heil, S., Grechenig T.: *Software Quality Development and Assurance in RUP, MSF and XP - A Comparative Study.*, ACM, 2005

Annotation

Quality Assurance in Software Products

The main purpose of this paper is to compare different methods of software development processes regarding to software quality assurance. In the first part the main features of processes like Rational Unified Process (RUP), Microsoft Solution Framework (MSF) , and Extreme Programming (XP) are described. Furthermore, it offers a short overview of recommendations of the ASME-NQA-2a-1990 software quality assurance standard for nuclear applications. It provides a deeper insight to the focus on quality and quality assurance beginning form the early phases of the software lifecycle trough the deployment, technical support and maintenance. It reveals the common features in all these software development processes and it is trying to reveal the advantages of different approaches. This results in a discussion about the possibilities of improving the individual models and also a judgment which criterions lead to consideration, that a particular model is more suitable.