PODPORA POŽIADAVIEK NA ZMENU – AUTOMATICKY ALEBO MANUÁLNE?

Rozdeľuj a panuj... Alebo si nájdi niekoho, kto bude rozdeľovať za teba.

Radoslav Kontúr

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

Abstrakt. *Jednou z najhlavnejších častí v životnom cykle vývoja softvéru je proces požiadaviek na zmenu. Tento proces je súčasťou fázy, na ktorú je vynakladaných najviac prostriedkov z celého životného cyklu a to fázy údržby. Kľúčovou úlohou v tomto procese je vedieť čo najrýchlejšie a čo najefektívnejšie prideliť danú požiadavku zodpovednému programátorovi, ktorý zabezpečí jej realizáciu. V súčasnosti práve táto úloha predstavuje najväčší problém vo fáze údržby softvéru, ktorý spôsobuje v prvom rade predĺženie procesu spracovania požiadaviek na zmenu a v druhom rade predstavuje veľký problém pre vedúceho tímu, ktorý musí dané pridelenie vykonať. Existuje viacero nástrojov, ktoré slúžia práve na podporu tejto úlohy. Takisto existujú aj metódy, ktoré podporujú tento proces bez použitia podporných nástrojov. V tejto eseji sa zameriam na porovnanie automatického prístupu s manuálnym prístupom k podpore procesu požiadaviek na zmenu a pokúsim sa navrhnúť vylepšenie jednej z týchto metód.*

Kľúčové slová: *požiadavka na zmenu, podporné nástroje, vedúci tímu, analytik, programátor*

Úvod

Fáza v životnom cykle vývoja softvéru, na ktorú sú vynaložené najväčšie náklady je podľa [1] fáza údržby, čo je znázornené na tabuľke 1, ktorá zobrazuje jednotlivé fázy životného cyklu vývoja softvéru aj s relatívnymi nákladmi na každú z týchto fáz.

Najvýznamnejšou časťou tejto fázy je proces požiadavky na zmenu. Je to proces, v ktorom je nutné vedieť rýchlo a efektívne prideliť úlohy správnym členom tímu. Práve toto prideľovanie predstavuje veľký problém a teda je zrejmé, že je nutné nejakým spôsobom podporiť tento proces. Otázkou teda nie je, či má zmysel podporovať tento proces, ale akým spôsobom ho čo najlepšie podporiť, podpora ktorej fázy procesu požiadaviek na zmenu má najväčší zmysel a hlavne aká forma podpory je najefektívnejšia? V tejto práci rozoberiem rôzne možnosti podpory tohto procesu, porovnám ich a uvediem aj praktickú skúsenosť s podporou tohto procesu. Ďalej uvediem, ktorými metódami a za akých okolností sa má podľa môjho názoru zmysel zaoberať.

Tab. 1. Relatívne náklady na jednotlivé fázy vývoja softvéru [1].

	Relatívne náklady
Analýza požiadaviek	3%
Špecifikácia požiadaviek	3%
Návrh	5%
Implementácia	7%
Testovanie	15%
Údržba	67%

Aké sú možnosti podpory procesu požiadavky na zmenu?

Počas fázy údržby v životnom cykle vývoja softvéru je najviac času stráveného štúdovaním doručených požiadaviek na zmenu. Ostatné procesy vo fáze údržby ako je napríklad návrh zmeny, implementácia zmeny alebo testovanie nevyžadujú až toľko veľa času [2].

Proces požiadavky na zmenu vždy začína u zákazníka. Zákazník, teda žiadateľ o zmenu, musí odoslať svoju požiadavku firme, ktorá vyvíja daný systém. Tieto požiadavky

sú bežne zasielané prostredníctvom formulárov na stránkach vývojárov alebo vyplnením dotazníka v papierovej podobe [3]. V oboch prípadoch sú formuláre vyplnené zákazníkom. Veľmi veľa nedorozumení, v procese požiadavky na zmenu, nastáva práve z toho dôvodu, že zákazník používa v požiadavke pojmy, ktoré nie sú úplne jasné programátorovi, ktorý má danú požiadavku zrealizovať. Povedzme že zákazník má vo svojom systéme okno, ktoré obsahuje viacero zoznamov a povedzme že chce do jedného z týchto zoznamov doplniť nový atribút. Zákazník v tomto prípade zadá požiadavku, v ktorej napríklad napíše že chce v oblasti XY doplniť do prehľadu nový atribút. Samozrejme v takomto prípade programátor nemôže vedieť o ktorý zoznam sa jedná a celý proces sa predĺži tým, že sa celá požiadavka dostáva do stavu čakania na doplňujúce informácie. Presne pre takéto prípady je možnosť podpory prostredníctvom nástrojov, ktoré sú schopné vygenerovať požiadavku. Systém, ktorý je nasadený u zákazníka musí mať možnosť prepnutia do režimu pre zasielanie požiadaviek na zmenu. Tieto nástroje na generovanie požiadaviek majú už prednastavené operácie ako je napríklad pridávanie atribútov, takže ak sa zákazník v tomto režime nastaví do spomínaného okna tak môže jednoducho odoslať požiadavku, ktorá už bude obsahovať aj presné informácie o tom aký atribút sa má pridať a hlavne kam sa má pridať. Programátor bude v tomto prípade môcť okamžite zareagovať a danú požiadavku zrealizovať.

Po doručení požiadavky na zmenu je potrebné túto požiadavku identifikovať a prideliť ju na riešenie konkrétnemu programátorovi. V bežnom prípade má túto úlohu na starosti vedúci tímu, ktorý vidí zoznam všetkých doručených požiadaviek a má za úlohu rozhodnúť o ich realizácií a posunúť ich na ďalšie spracovanie. Táto fáza môže byť takisto podporená a to nástrojom na automatické smerovanie prichádzajúcich požiadaviek na zmenu. Tento prístup je založený na myšlienke, že najideálnejší kandidát na riešenie tejto požiadavky je programátor, ktorý už vyriešil podobnú požiadavku v minulosti [4]. Jedná sa teda o nástroje, ktoré vychádzajú z histórie implementácie daného vyvíjaného systému. Je ale dôležité si uvedomiť, že nástroj na automatické smerovanie požiadaviek nemôže fungovať vždy bez obsluhy človeka. Tieto nástroje sú schopné aj automaticky odoslať prijatú požiadavku, ale v prípade, že nevedia jednoznačne nájsť adresáta, tak iba ponúknu zoznam vhodných vývojárov z ktorých musí byť jeden manuálne zvolený. Tu môžeme zaviesť hlavnú otázku a to: Je výhodnejšie využiť nástroj na automatické smerovanie požiadaviek, alebo manuálne smerovať každú požiadavku?

Metóda „obetného baránka“

V tejto časti opíšem moju praktickú skúsenosť so snahou o podporu procesu požiadaviek na zmenu vo firme, v ktorej momentálne pôsobím. Hlavnou myšlienkou tejto snahy bolo zvoliť na každý týždeň jedného programátora a jedného analytika, ktorý by boli primárne zodpovedný za spracovávanie všetkých požiadaviek zo strany zákazníka. Teda celý proces vyzeral nasledovne. Predpokladajme, že na aktuálny týždeň boli zvolený analytik A a programátor P. Všetky požiadavky zo strany zákazníka boli adresované priamo na analytika A, ktorý zhodnotil a prípadne aj naplánoval spracovanie požiadavky a posunul ju zvolenému programátorovi P. Programátor P mal za úlohu implementovať danú požiadavku. Cieľom tohto vylepšenia bolo odbremenenie vedúceho tímu, ktorý bol inak zodpovedný za spracovanie a naplánovanie všetkých požiadaviek. Po asi mesiaci

používania tejto metódy vo firme môžem povedať, že naozaj došlo k odbremeneniu vedúceho tímu, ktorý sa takto mohol plne sústrediť na vedenie zvyšku ľudí v tíme a plnenie ostatných úloh, takže naozaj došlo k určitému zefektívneniu procesu požiadaviek na zmenu.

Avšak negatívny dopad mal tento systém na dvoch zvolených pracovníkov. Za normálnych okolností fungovalo všetko dobre. Analytik A prijal požiadavku naplánoval zmenu a pridelil jej implementáciu zodpovednému programátorovi P. Ibaže často dochádzalo k neštandardným situáciám, kedy napríklad analytik prijal požiadavku, ktorá sa týka oblasti vyvíjaného systému, s ktorou nemá skúsenosti. V takomto prípade musel buď stráviť nejaký čas študovaním dokumentácie, alebo musel požiadať o pomoc analytika, ktorý ma na starosti danú oblasť vyvíjaného systému. Tento problém nebol až tak výrazný pri analytikoch ako pri programátoroch. Každý programátor v podniku býva väčšinou špecializovaný na určitú oblasť (či už oblasť v danom systéme, alebo oblasť jeho zamerania, alebo oboje). Nie je preto možné zabezpečiť, aby zvolený programátor P vedel sám vyriešiť všetky doručené požiadavky, takže ak dostal práve požiadavku z oblasti ktorá nespadá pod jeho zameranie, tak potreboval ďalšie konzultácie od svojich kolegov.

Ďalší veľký problém v tejto metóde nastával vtedy, ak analytik dostal požiadavku na zmenu, ktorá sa týkala väčších zmien v systém a vyžadovala väčší počet programátorov. V takomto prípade musela byť požiadavka riešená priamo vedúcim tímu, ktorý ju naplánoval a pridelil jednotlivé úlohy potrebným členom tímu. Na tomto príklade môžeme teda pozorovať, že je naozaj dôležité aby boli požiadavky smerované. Aj keď si myslím, že táto metóda môže bezchybne fungovať v prípade, že by sa jednalo o menší projekt alebo v prípade, že zvolený analytik a programátor by boli schopný vyriešiť každú požiadavku na zmenu bez akejkoľvek pomoci.

Možné vylepšenia („kruhová metóda“)

Keby sme išli znova zavádzať túto metódu, tak by sme ju mohli vylepšiť niekoľkými spôsobmi. Ako som už načrtol, tak prvý problém pri použití tejto metódy nastával vtedy, keď zvolený zodpovedný analytik mal riešiť požiadavky z oblastí v ktorých nemal prehľad.

Keď zoberieme do úvahy projekt, na ktorom bude dokopy pracovať napríklad iba 10 až 15 programátorov a analytikov, tak by sa celý proces dal zjednodušiť tým, že sa na každý týždeň sa nebude voliť jeden zodpovedný analytik, ale dopredu sa definuje kruhový zoznam všetkých týchto analytikov a dajme tomu, že každý týždeň bude ten kruh začínať od iného analytika. Analytik ktorý by bol prvý v tomto kruhovom zozname, by si najprv prečítal doručenú požiadavku na zmenu a ak by ju nevedel ďalej spracovať, tak by ju jednoducho iba posunul ďalšiemu analytikovi v poradí. Celý tento proces by sa dal automatizovať zavedením jednoduchého nástroja, ktorý by iba posúval požiadavku podľa prednastaveného poradia. Nevýhodou tejto metódy by bolo to, že zahrnie do procesu aj ľudí, ktorí by v tomto procese nemali byť zúčastnený. Musím ale povedať, že sa zúčastnia tohto procesu iba s minimálnym zbytočným časom. Analytik často krát už podľa názvu požiadavky môže vedieť že túto požiadavku bude musieť iba posunúť ďalej, takže zbytočný čas by bol iba ten čas pokiaľ by si prečítal požiadavku. Podstatná zmena by bola hlavne v tom, že ak by daný analytik nevedel vyriešiť danú požiadavku, tak by ju nešiel

riešiť s pomocou od ďalšieho analytika, ale iba by ju pridelil ďalšiemu v poradí. Takýto istý kruhový zoznam by bol zavedený aj pre programátorov. V oboch prípadoch, ak by sa mala požiadavka na zmenu vrátiť prvému pracovníkovi, tak by sa odoslala vedúcemu tímu, ktorý by ju musel vyriešiť. Pri tejto metóde by daní zamestnanci museli byť upovedomení o tom, že musia doručenú požiadavku bezodkladne posunúť ďalej ak ju nevedia sami vyriešiť, inač by dochádzalo k príliš veľkým zdržaniam.

Ťažšie by bol už vyriešiť problém, keď analytik dostane požiadavku na zmenu, ktorá sa týka väčšieho celku a je potrebné na jej riešenie zapojiť viacero pracovníkov. Takéto požiadavky musia byť riešené priamo vedúcim tímu, takže by bolo potrebné nejakým spôsobom rozlíšiť jednoduché požiadavky od tých zložitých a tie zložité hneď odoslať vedúcemu tímu.

Automatika alebo ručné spracovanie?

Myslím si, že jednoznačná odpoveď na otázku, či je výhodnejšie využiť nástroj na automatické smerovanie alebo zostať pri manuálnom prideľovaní jednotlivých požiadaviek neexistuje. V nasledujúcom texte som zvolil niekoľko kritérií, podľa ktorých som porovnal automatický prístup oproti manuálnemu.

V prvom rade zoberiem do úvahy veľkosť projektu. Aký projekt môžeme pokladať za veľký? Na veľkosť projektu sa môžeme pozrieť z dvoch hľadísk:

1. Počet programátorov
2. Počet modulov v systéme

Keď si predstavíme projekt, na ktorom pracuje iba jeden programátor, tak je jasné, že sa vôbec nemá zmysel zaoberať otázkou prideľovania požiadaviek na zmenu. V takomto prípade vedúci tímu automaticky prideľuje každú požiadavku v danom projekte práve tomuto programátorovi. S rastúcim počtom programátorov ale náročnosť tohto problému prudko stúpa. Ak má projekt viac programátorov, tak pri doručení požiadavky na zmenu musí vedúci tímu brať do úvahy to, že každý z týchto programátorov je špecializovaný na inú oblasť. Teda jeden programátor je databázový špecialista, ďalší je zameraný na webové služby ďalší napríklad na používateľské rozhranie. Ak má projekt ešte viac programátorov, tak aj každá z týchto oblastí môže mať viacero špecialistov. Pre vedúceho tímu sa tým pádom stáva ťažšou úlohou pridelenie požiadavky správnej osobe. Môžeme teda tvrdiť, že čím je väčší počet ľudí zúčastnených na projekte, tým náročnejšie sa stáva manuálne smerovanie požiadaviek. V takomto prípade má nástroj na automatické smerovanie požiadaviek veľké uplatnenie a pri jeho správnom použití môže urýchliť celý proces požiadaviek na zmenu a uľahčiť prácu viacerým pracovníkom.

Nie vždy počet zúčastnených programátorov určuje aj veľkosť projektu. Na jednej strane máme veľký projekt, na ktorom pracuje veľa vývojárov, lenže tento projekt je zložený z množstva modulov, ktoré sú na sebe nezávislé a jednotliví programátori majú prehľad iba o svojom kóde v rámci modulu na ktorom pracovali. Na druhej strane môžeme mať veľký projekt, na ktorom sa mohol podieľať rovnaký počet vývojárov, lenže tento projekt už nie je členený na nezávislé moduly ale pozostáva z „jednotného“ zdrojového kódu, takže všetci programátori by tu už mali mať prehľad o takmer celom zdrojovom kóde. Myslím si, že v takomto prípade by bolo efektívnejšie zvoliť metódu

„obetného baránka“ pre projekt, ktorý má „jednotný“ zdrojový kód, teda pre projekt, ktorý nie je zložený z viacerých na sebe nezávislých modulov. Síce je dosť pravdepodobné, že zvolený programátor nie vždy bude vedieť vyriešiť každú požiadavku, ale som si istý, že keď má k dispozícii celý zdrojový kód, tak bude aspoň vedieť ktorej oblasti sa daná požiadavka na zmenu týka a teda bude vedieť ktorého programátora má kontaktovať. Ak by bola metóda „obetného baránka“ použitá pre viac-modulový projekt, tak zvolený programátor či analytik by mal určite vo väčšine prípadoch problém pri hľadaní správneho pracovníka, ktorý by vedel vyriešiť daný problém. Môžeme teda tvrdiť, že pre projekt zložený z viacerých nezávislých modulov by bolo manuálne prideľovanie požiadaviek menej efektívne ako automatické smerovanie.

Pri malých projektoch je to jednoduchšie. Som si istý, že ak máme projekt, na ktorom robia napríklad iba traja programátori, tak jednotlivý programátori majú o sebe navzájom prehľad, a vedia si aj rozdeliť pridelené požiadavky, takže v tomto prípade by bolo použitie nástroja na automatické smerovanie zbytočné a myslím si, že vo väčšine prípadoch by malo dokonca negatívny dopad na celý proces požiadavky na zmenu. Myslím si to preto, lebo musíme zobrať do úvahy to, že zavedenie takéhoto nástroja vyžaduje určité náklady. Ďalej musíme rátať s tým, že aby takýto nástroj fungoval správne potrebuje mať nastavenú databázu s históriou zmien vo vyvíjanom systéme. Čas strávený pri migrácií týchto údajov by bol v malom projekte určite zbytočný a predstavoval by v konečnom dôsledku nemalé straty.

Keď sa pozrieme na porovnanie automatického a manuálneho prístupu z hľadiska počtu zainteresovaných ľudí, tak sa na prvý pohľad môže zdať, že sa menej ľudí bude podieľať na procese pri použití nástroja na automatické triedenie požiadaviek. Myslím si, ale že to nie je úplne pravda, pretože ako som už spomínal, tak nástroje na automatické smerovanie požiadaviek väčšinou nevedia jednoznačne prideliť danú požiadavku konkrétnemu programátorovi, ale iba ponúknu zoznam možných ľudí na pridelenie. Takže počet zainteresovaných ľudí je pri automatickom triedení väčšinou rovnaký ako pri tom manuálnom. Dokonca si myslím, že ak nastanú pri automatickom triedení komplikácie a tento nástroj odošle požiadavku nesprávnemu programátorovi, tak v konečnom dôsledku môže byť pri automatickom prístupe zainteresovaných ľudí omnoho viac ako pri tom manuálnom.

Zhodnotenie

V tejto eseji som sa snažil rozobrať možné prístupy k podpore procesu požiadaviek na zmenu nielen pomocou podporných nástrojov, ale aj pomocou zmeny v organizácií tímu pri tomto procesu. Dospel som k záveru, že nemožno jednoznačne tvrdiť že použitie jedného prístupu je vždy výhodnejšie ako použitie toho ďalšieho. Snažil som sa ukázať kedy sa má ktorý prístup použiť a naopak, kedy by sa zase daný prístup nemal použiť. Myslím si, že každý podporný nástroj sa vyvíja s cieľom uľahčiť nejakú prácu. Je ale dôležité aby každý, kto sa chystá použiť nejaký podporný nástroj, najprv uvážil, či takáto zmena prinesie zlepšenie alebo naopak bude celý proces zbytočne predĺžený. Celá moja práca je okorenená aj vlastnou praktickou skúsenosťou s podporovaním procesu požiadavky na zmenu. Po prístupoch, ktoré som si naštudoval a po skúsenostiach z praxe som dospel k riešeniu, ktoré by mohlo danú metódu vylepšiť.

Použitá literatúra

1. Vo, H. (2009, July 29). *Software Engineering*. Retrieved from the Connexions Web site: http://cnx.org/content/col10790/1.1/
2. D. C. C. Poo and M. K. Chung, "*CASE and software maintenance practices in Singapore*", Journal of Systems and Software, Vol. 44, Issue 2, December 1998.
3. B. Blum, *Software Engineering*, A Holistic View, Oxford University Press, New York, 1992.
4. Gerardo Canfora , Luigi Cerulo: *Supporting Change Request Assignment in Open Source Development*

Annotation

Support of change request process – automatic or manual?

Process of change request is one of the most important and most expensive processes in software development life cycle. This essay discusses the means of support to this process. This work critically compares the approach based on supporting this process by using a tool for automatic routing of change requests with approach based on organization changes in this process. The work is enriched by the author's practical experience with the support of this process.