Vzťah zákazník a vývojár v softvérových projektoch

ANDREJ NECZLI

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

Abstrakt. Súčasné trendy v oblasti vývoja softvéru naznačujú čoraz väčšiu potrebu existencie vzťahu medzi zákazníkom (zadávateľom projektu) a vývojárom daného softvéru. Potreba zdokonalenia a prehĺbenia komunikácie medzi nimi je len logickým vyústením neustáleho konkurenčného boja firiem v snahe vyvinúť softvérový produkt v čo najkratšom čase. Tento článok sa venuje problémom komunikácie medzi oboma subjektami, úrovniam komunikácie medzi nimi, ako aj samotnej úlohe zákazníka pri vývoji softvéru. Esej ďalej pojednáva o agilných metodikách ako novom trende vývoja softvéru a skúma vzťah zákazníka a vývojára pri extrémnom programovaní, hlavnom predstaviteľovi týchto metodík. Na záver upozorňuje na existenciu liniek medzi zákazníkom a vývojárom, ktoré môžu do značnej miery zefektívniť vývoj softvéru aj v budúcnosti.

Úvod

Medzi zákazníkom a vývojárom existuje krehký, dynamický vzťah, meniaci sa časom, požiadavkami na vyvíjaný softvér, ako aj odlišnými pohľadmi vývojára na zákazníka.

V takomto vzťahu zákazník nemôže mať len pasívnu úlohu na projekte, ale mal by byť aktívnou súčasťou v procese vývoja. Obaja, tak zákazník ako aj vývojár, si musia neustále overovať, či sa správne chápu a skúmať, či všetko podstatné bolo zodpovedané.

Proces komunikácie medzi týmito dvoma subjektami je dosť zložitý a skrýva v sebe mnoho úskalí, ktoré možno prekonať len výberom vhodných liniek medzi zákazníkom a vývojárom spomenutých v závere tejto eseje. Najprv sa však budeme venovať problémom komunikácie, úrovňami komunikácie a predovšetkým využitiu komunikácie pri použití nových vývojových agilných metodík s dôrazom na extrémne programovanie.

Manažment v softvérovom inžinierstve, december 2005, s. 1-11.

Problémy komunikácie so zákazníkom

Okolo každého projektu dochádza z času na čas k ostrejšej diskusii medzi zadávateľom projektu (zákazníkom) a vývojovým tímom. Obe strany by sa mali snažiť riešiť podobné situácie dohodou založenou na racionálnej analýze. Ale v niektorých prípadoch môže vývoj vyústiť do situácie, keď zákazník vyjadruje značnú mieru nespokojnosti a vyhráža sa podniknutím právnych krokov, ako sú penále alebo vypovedanie zmluvy. Niekedy sa môže dodávateľ iba domnievať, čo takúto reakciu zo strany zákazníka vyvolalo.

Hlavná prekážka, ktorú si musíme pri komunikácii so zadávateľom projektu (zákazníkom) a budúcimi užívateľmi vždy uvedomiť, spočíva v tom, že ide o špecialistov z iného odboru používajúcich odlišné termíny a de facto hovoriacich iným jazykom. Väčšinou nebudú informačným technológiám rozumieť do detailov, a preto je treba sa pri jednaní s nimi vyhnúť podrobnostiam. Užívateľa (zákazníka) zaujíma predovšetkým to, čo vidí na obrazovke a ako bude s aplikáciou pracovať. Aj keď má zadávajúca firma svojho správcu informačného systému, musíme počítať s tým, že ani on nebude orientovaný na spomínaný iný typ informácií, aký používa práve vývojový tím. Pre neho bude dôležité skôr to, aký hardvér a softvér aplikácia vyžaduje, ako má celý systém nastaviť a či aplikácia dodržuje všetky bezpečnostné pravidlá. Preto nemôže vývojový tím očakávať, že mu zákazník nejakým tvorivým spôsobom pomôže pri riešení technických problémov. [5]

Úrovne komunikácie so zákazníkom

Komunikácia so zákazníkom prebieha na niekoľkých úrovniach. Pred zahájením projektu je potrebné vyriešiť všetky obchodné a právne otázky. V rámci projektu sa vymenuje takzvaný projektový výbor, čo je grémium pre formálnu komunikáciu zúčastnených strán. Jeho členmi bývajú projektoví manažéri a niekto z ich nadriadených. Obvykle sa schádzajú pravidelne k posúdeniu priebehu projektu, na mimoriadnych stretnutiach sa preberajú prípadné problémy, ktoré sa na úrovni vedúcich projektov vyriešiť nepodarilo.

Najčastejšia komunikácia prebieha medzi vedúcimi projektu oboch strán – na strane zákazníka môže byť jeho skutočná funkcia iná, ale vždy musí existovať niekto, kto má projekt v zadávajúcej organizácii na starosti, napríklad správca alebo manažér informačných systémov. Komunikácia na tejto úrovni býva niekedy formálna, napríklad keď si obe strany vymieňajú zoznam požiadaviek alebo nájdených chýb, ale väčšinou ide o bežnú výmenu informácií o tom, ako sa projekt vyvíja, aké sú názory koncových užívateľov, kedy prebehnú konzultácie špecialistov a podobne. Je prirodzené, že občas dochádza ku sporom, ale obaja manažéri by sa mali snažiť nakoniec nejakým spôsobom dohodnúť. Pokiaľ sa to nepodarí, rieši problém projektový výbor, ale je veľmi nebezpečné, keď k tomu dochádza často a prirodzená komunikácia sa zmení na formálnu výmenu dokumentov.

A na tej poslednej úrovni si informácie vymieňajú odborní pracovníci vo väčšine prípadov úplne neformálne a nepredvídateľne. Niekedy sa projektoví manažéri snažia túto komunikáciu obmedziť, vynára sa však otázka, nakoľko je to účinné. Väčšinou ide o krátke telefonické rozhovory alebo výmenu e-mailov, keď obidvaja pracovníci diskutujú o nejakom detaile, ktorý nie je pre ďalší priebeh projektu významný.

Písomné zápisy komunikácie

Z každého jednania, ktoré môže mať na ďalší priebeh projektu nejaký vplyv, aj keď ide iba o krátky telefonický rozhovor, je potrebné vytvoriť zodpovedajúci písomný zápis a zaslať ho každému, koho sa týka. Pokiaľ sa dvaja technici dohodnú na tom, že na obrazovke číslo 34 bude nové textové pole, stačí poslať krátky e-mail vedúcim projektu a tomu členovi tímu, ktorý má na starosti užívateľské rozhranie. Z jednania projektového tímu bude naopak vytvorený formálny zápis obsahujúci závery všetkých preberaných bodov, ktoré obdržia všetci jeho členovia.

Pokiaľ dôsledky jednania vyžadujú zmenu v oblasti, ktorá je opísaná niektorým dokumentom, ako je napríklad zoznam požiadaviek, schéma databázy, návrh užívateľského rozhrania a podobne, musí zápis dostať taktiež pracovník, ktorý je za udržiavanie tohto dokumentu zodpovedný, aby mohol čo najskôr tieto úpravy do dokumentu zaznamenať. Mal by taktiež uviesť odkaz na jednanie, na ktorého základe sa zmena prevádza, aby v budúcnosti bolo možné vystopovať jej príčinu.

Úloha zákazníka pri vývoji softvéru

Klasické techniky softvérového inžinierstva sa týkajú situácie, ktorá bola platná pred desiatimi, dvadsiatimi rokmi – tejto situácii plne vyhovujú a jej požiadavky bezvýhradne napĺňajú. Softvér bol vyvíjaný skôr väčšími, organizovanými a stabilnými tímami. Dôraz, ktorý sa kládol na kvalitu, bol ešte stále spôsobený obavou z toho, aby sa neopakovalo čosi ako softvérová kríza, ktorá spomalila vývoj programov pred nástupom softvérového inžinierstva. Zákazník si vtedy radšej počkal trochu dlhšie (nič iné mu ani neostávalo, lebo konkurencie bolo málo), ale očakával zato kvalitný výsledok.

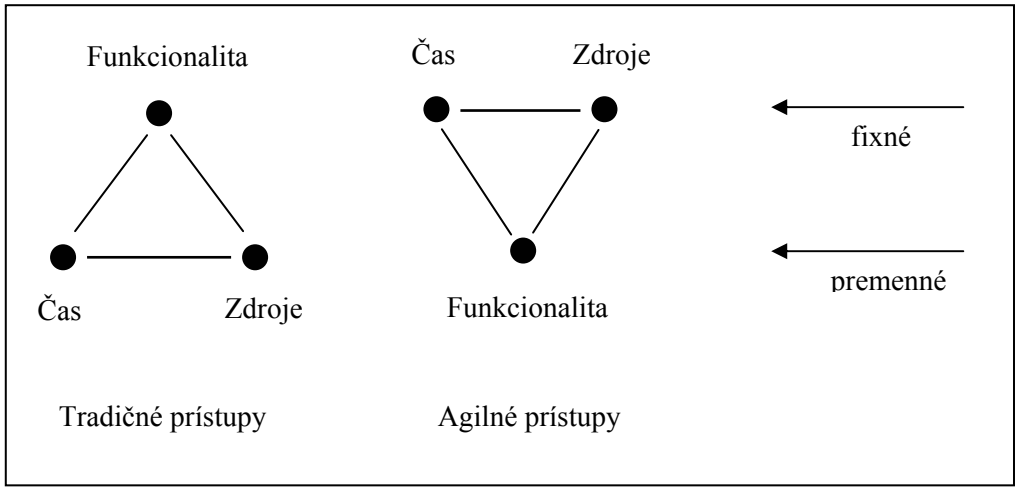
Svet sa však zmenil, a to hneď v niekoľkých smeroch. Predovšetkým, zákazník síce stále očakáva kvalitu, ale už odmieta na ňu dlho čakať. V prípade internetových projektov naviac zákazník často nie je dopredu ani definovaný. Budúci užívatelia nevedia (nebudú vedieť), čo má systém poskytovať. Aj u iných projektov sa však v praxi objavuje zaujímavý paradox – až pozoruhodne často je zákazník spokojný s relatívne nepoužiteľným produktom. Môže to byť spôsobené okrem iného tým, že pri samotnom zadaní projektu ešte ani presne nevie, čo vlastne od produktu požaduje. Agilné metodiky tento problém riešia typicky omnoho užším spojením zákazníka s vývojovým tímom.

Druhým faktom je, že programátorské firmy sa zmenšujú, pretože vzniká obrovské množstvo menších programov. „Firma“ je tak často zložená z piatich ľudí, výnimkou nie sú ani tímy, pozostávajúce z dvoch alebo troch programátorov.

Nakoniec, najdôležitejším parametrom pri vývoji softvéru je rýchlosť. Hlavným cieľom predsa je, aby bola aplikácia čo najskôr nasadená a mohla si získavať nových užívateľov. U internetových aplikácií pritom platí, že funkčnosť možno pridávať „za behu“, teda že je možné spustiť čo najskôr obmedzenú (avšak plne fungujúcu) verziu aplikácie a ďalšie požiadavky na funkcionalitu realizovať postupom času. Dnešné požiadavky na rýchly a flexibilný vývoj aplikácií sa snažia riešiť už vyššie spomenuté agilné metodiky, ktorým bude venovaná nasledujúca časť.

Agilné metodológie – základné princípy

Rozdiel medzi tradičnými prístupmi a agilnými metodikami možno najlepšie pozorovať na obrázku **Obr. 1**, ktorý porovnáva chápanie základných premenných pri vývoji softvéru:



Obr. 1. Rozdiel medzi tradičnými a agilnými programovacími prístupmi

Tradičné metodiky (znázornené ľavým trojuholníkom) vychádzajú z potreby naplniť za každú cenu dokument *Špecifikácia požiadaviek*. Požiadavky – teda funkcionalita – sú fixné, pričom v úlohe premenných vystupujú čas a potrebné zdroje. Pri tradičnom vývoji bolo teda jasné, čo bude program robiť, avšak ťažko sa odhadovalo, koľko to bude stáť a ako dlho to bude trvať.

Agilné prístupy oproti tomu považujú čas a zdroje za fixné (stanovené na začiatku projektu zadávateľom) a mení sa (resp. priebežne sa prispôsobuje) funkcionalita (požiadavky). Na začiatku projektu je teda presne stanovený najdlhší možný čas vývoja a najvyššie možné náklady. Do týchto atribútov sa potom vývoj musí vojsť. V priebehu

prác na projekte tím komunikuje so zákazníkom a priebežne prehodnocuje priority. Zákazník by v ideálnom prípade mal byť členom vývojového tímu, mal by komunikovať s vývojovým tímom, participovať na návrhu a spolurozhodovať o testoch. Agilné metodiky vyžadujú oveľa menší objem formálnych a byrokratických artefaktov. Vychádzajú pritom z pevného presvedčenia, že *základným, jednoznačným, exaktným, nespochybniteľným a vlastne jediným spoľahlivým nositeľom informácie je zdrojový kód*. Uvedená myšlienka je základným filozofickým východiskom aj pre extrémne programovanie (*Extreme Programming*, XP), pravdepodobne najrozšírenejšiu a najznámejšiu agilnú metodiku. Preto si ju prezrime trochu bližšie, hlavne v súvislosti s rolou zákazníka pri XP. [3]

Vzťah zákazníka a vývojára pri extrémnom programovaní

Extrémne programovanie je metodika vhodná pre malé až stredné tímy (zvyčajne sa uvádza počet dvoch až desiatich programátorov), ktorí sa pri vývoji musia vyrovnať s rýchle sa meniacim alebo nejasným zadaním.

Zákazník tu má pomerne dôležitú pozíciu, pretože hneď po programátorovi (vývojárovi) zastáva druhú kľúčovú rolu. Programátor síce dokáže programovať, ale nevie čo. Zákazník to síce neovláda, ale zato vie, čo je potrebné programovať. Preto je zákazník neoddeliteľnou súčasťou vývoja softvéru podľa XP. Na zákazníka sú však taktiež kladené niektoré neštandardné požiadavky: musí sa zžiť s niektorými zásadami XP, ako napríklad naučiť sa písať zadania a testy funkcionality a v neposlednom rade aj prijať možnosť ovplyvňovať projekt a zodpovedať zaň. Skutočný zákazník musí sedávať s vývojovým tímom, byť k dispozícii, odpovedať na otázky, riešiť konflikty a stanovovať drobné priority. Musí to byť niekto, kto bude skutočne používať systém po zavedení do prevádzky.

Formovanie vzťahu zákazník-vývojár pomocou *Pattern Language*

Svet možno vnímať ako systém zložený z malých obrazcov (*patterns*), pričom nikdy nemožno spozorovať, aby bol čo i len jeden obrazec (*pattern*) v izolácii. Vždy by mal byť súčasťou nejakej zbierky obrazcov alebo jazyka obrazcov (*pattern language*) [1]. Autor tejto myšlienky John Muir takýto jazyk pre interakciu zákazníka s vývojárom vytvoril. Podľa neho mnohé z týchto obrazcov, ak nie všetky, by sa mali dať aplikovať v agilnom vývojovom prostredí, aj keď neboli napísané vyslovene pre tento účel. Tu sú niektoré z nich (a ďalšie, ktoré sú v príbuzných jazykoch):

- *Ide o vzťahy, nie o predaj*: Rozvíjajte vzťahy so zákazníkom. Sústreďte sa na tento vzťah, nie na aktuálnu transakciu. Využitie: *Pochopenie zákazníka* a *Získanie dôvery*.
- *Pochopenie zákazníka*: Spoznajte zákazníka čo najviac: Využitie: *Efektívne počúvanie*, *Včasná odpoveď* a *Stretnutia popri stretnutí*.

- *Získanie dôvery*: Každý kontakt so zákazníkom je šancou na získanie dôvery. Využite ju. Využitie: *Efektívne počúvanie*, *Včasná odpoveď* a *Stretnutia popri stretnutí*.
- *Efektívne počúvanie*: Počúvajte zákazníka so zámerom pochopiť ho. Využitie: *Osobná integrita*, *Uvedomenie si hraníc*, *Pomoc zákazníkovi* a *Dobré spôsoby*.
- *Včasná odpoveď*: Ak dostanete od zákazníka nejakú požiadavku, dajte mu vedieť, že ste ju prijali a ako ju mienite riešiť.
- *Stretnutia popri stretnutí*: Príďte na schôdzku dostatočne skoro na to, aby ste sa stretli s ostatnými účastníkmi a strávili s nimi nejaký čas. Po stretnutí venujte trochu času a pohovorte si s ostatnými o bežných záležitostiach.
- *Osobná integrita*: Nezadržiavajte dôležité informácie od zákazníka, ale stanovte si určité hranice.
- *Pomoc zákazníkovi*: Nehádajte sa. Pokúste sa pochopiť, akým spôsobom zákazníkov biznis funguje. Nesnažte sa uchlácholiť zákazníka dávaním sľubov, ktoré nemôžete dodržať. Využitie: *Uvedomenie si hraníc* a *Dobré spôsoby*.
- *Uvedomenie si hraníc*: Berte každú konverzáciu so zákazníkom ako časť rokovania. Nediskutujte o tom, čo nie je súčasťou Vašej zodpovednosti. Využitie: *Dobré spôsoby*.
- *Dobré spôsoby*: Buďte zdvorilý. Oblečte sa adekvátne k tomu, čo očakáva od Vás zákazník. Preukážte rešpekt ku každému, vrátane konkurencie. Buďte opatrní v interakcii s ostatnými pred zákazníkom.

Funkcia liniek medzi zákazníkom a vývojárom

Mnohé z najlepších nápadov pre nové produkty a produktové zlepšenia prichádzajú zo strany zákazníka alebo koncového užívateľa produktu a práve tento fakt si vyžaduje zriaďovanie jednej alebo viacerých liniek medzi zákazníkom a vývojárom (*customer-developer links*). Tieto sú definované ako určité kanály, umožňujúce zákazníkom a vývojárom si vymieňať navzájom informácie. [4]

Dnes je k dispozícii obrovské množstvo liniek, ktoré predstavujú príležitosť ako aj výzvu pre manažérov softvérového vývoja. Príležitosť spočíva v tom, že nebolo nikdy ľahšie získať vstupy od zákazníka než teraz a výzva tkvie v tom, že vývojoví manažéri si v súčasnosti musia vedieť vybrať z veľkého množstva liniek, z ktorých sú mnohé často mätúce. Preto je rozhodnutie o tom, koľko a aké typy liniek sa budú využívať čokoľvek, len nie priamočiara záležitosť. Toto náročné rozhodnutie by mohlo pomôcť uľahčiť vymedzenie hlavných rozdielov medzi nimi, nachádzajúcich sa v tabuľke **Tab. 1.** Rozlišuje sa pritom medzi dvoma vývojárskymi prostrediami, teda či

ide o softvér na zákazku (určený na použitie v rámci zákazníckej firmy) alebo softvérový balík (určený pre komerčné využitie):

Dimenzia vývoja	Softvér na zákazku	Softvérový balík
Cieľ	Softvér, vyvíjaný pre použitie v rámci zákazníckej firmy (zvyčajne nie na predaj)	Softvér, vyvíjaný pre vonkajšie použitie (na predaj)
Moment, kedy je identifikovaná väčšina zákazníkov	Pred zahájením vývoja	Potom, ako sa vývoj ukončí a produkt ide na trh
Počet zákazníkov (organizácií)	Zvyčajne jeden	Veľa
Fyzická vzdialenosť medzi zákazníkom a vývojárom	Zvyčajne malá (zákazníci sídlia v rovnakej budove ako vývojári)	Zvyčajne veľká (zákazníci môžu byť vzdialení od vývojárov aj tisíce kilometrov)
Typ projektu	Projekt nového systému; zlepšovanie údržby	Nové produkty, nové verzie (hlavné a menšie)
Požiadavky na softvérového zákazníka	Užívateľ, koncový užívateľ	Zákazník
Všeobecné ukazovatele úspešnosti	Spokojnosť, priaznivé prijatie	Predaj, podiel na trhu, dobré ohlasy na produkt

Tab. 1. Hlavné rozdiely medzi vývojárskym prostredím softvéru na zákazku a softvérového balíku.

Tabuľka **Tab. 2** potom uvádza príklad, ako môžu byť definované jednotlivé linky medzi zákazníkom a vývojárom vzhľadom na spomenuté vývojové prostredia a ich kombináciu. Inventár týchto liniek tvorí základ koncepcie počítania liniek v rámci daného projektu, pretože čím väčšia je účasť zákazníka na procese vývoja, tým úspešnejší softvérový projekt môže byť.

Linka	Popis	Zvyčajne používané pri: Z=SW na zákazku, B=SW balík
Sprostredkovateľský tím	Sprostredkovaný štruktúrovaný kurz so zákazníkmi, slúžiaci na vylákanie požiadaviek od nich.	Z
MIS sprostredkovateľ	Niekto, kto definuje zákaznícke ciele a potreby pre dizajnérov a vývojárov.	Z
Podporná línia	Oddelenie, ktoré pomáha zákazníkom s každodennými problémami (tzv. zákaznícka podpora, technická podpora, help desk).	Z/B
Prieskum	Anketa vykonaná na vzorke zákazníkov.	Z/B
Prototypovanie užívateľského rozhrania	Zákazníkom je predvedená demo- alebo nejaká ranná verzia, aby sa odhalili problémy s užívateľským rozhraním.	Z/B
Prototypovanie požiadaviek	Zákazníkom je predvedená demo- alebo nejaká ranná verzia, aby sa spresnili požiadavky na systém.	Z/B
Interview	Rozhovor s koncovým užívateľom, otvorený a pološtruktúrovaný.	Z/B

Testovanie	Nové požiadavky a spätná väzba, prameniaca z testovania. Nezahŕňa zisťovanie chýb.	Z/B
E-mail/bulletin board	Dodatočné problémy zákazníkov, otázky a návrhy na bulletin board alebo cez e-mail.	Z/B
Laboratória pre užitočnosť	Špeciálne vybavené laboratória zamerané na identifikáciu vecí, ktoré užívateľ pri práci využíva.	Z/B
Pozorovacie štúdium	Užívatelia sú sledovaní na určité obdobie, aby sa zistilo, čo vlastne robia.	Z/B
Marketing a predaj	Zástupcovia sa pravidelne stretávajú so zákazníkmi (súčasnými aj potenciálnymi), aby si vypočuli ich návrhy a potreby.	B
Užívateľská skupina	Skupiny zákazníkov sa pravidelne schádzajú, aby prediskutovali používanie softvéru a jeho možné zlepšenia.	B
Trade show	Zákazníkom je na trade show predstavený prototyp s cieľom získať od nich spätnú väzbu.	B
Skupina, sústrediaca sa na niečo	Malá skupina zákazníkov a moderátor sa zídu, aby diskutovali o danom softvéri.	B

Tab. 2. Linky medzi zákazníkom a vývojárom (*customer-developer links*).

Záver

Komunikácia medzi zákazníkom a vývojárom je téma na mnoho diskusií. Ja som sa snažil podať náhľad na problematiku hlavne zo strany vývojára, pričom som dal dôraz na využitie komunikácie zákazníka a vývojára hlavne pri nových technikách vývoja softvéru, akými sú agilné metodiky.

Taktiež zaujímavý je pohľad na komunikáciu medzi zákazníkom a vývojárom prostredníctvom takzvaného jazyka obrazcov (*pattern language*), ktorý nachádza využitie nielen v oblasti komunikácie vývojára so zákazníkom, ale môže mať aj všeobecné uplatnenie. Tento jazyk by mal napomáhať do značnej miery pri vytváraní a udržiavaní medziľudských vzťahov ako takých.

No a nakoniec definovanie liniek medzi týmito dvoma subjektami podstatne uľahčí budúci vývoj komunikácie medzi zákazníkom a dodávateľom softvéru a proces vzájomnej výmeny informácií bude omnoho efektívnejší.

Použitá literatúra

1. Fraser, S., Martin, A., Biddle, R., Hussman, D., Miller, G., Poppendieck, M., Rising, L., Striebeck, M.: The role of the Customer in Software Development: The XP Customer – Fad or Fashion? *Companion to the 19th annual ACM SIGPLAN conference on Object-oriented programming systems, languages and applications*. ACM Presss, New York (2004), 148-150.
2. Heiskanen, A., Similä, J.: Gatekeepers in the action structure of software contracting: A case study of the evolution of user-developer relationships. *ACM SIGCPR ComputerPersonnel*, Vol. 14, No. 1-2 (1992) 30-44.
3. Kadlec, V.: *Agilní programování. Metodiky efektivního vývoje software.* Computer Press, Brno, 2004. (CZ)
4. Keil, M., Carmel, E.: Customer-Developer Links in Software Development. *Communications of the ACM*, Vol. 38, No. 5 (1995) 33-44.
5. Paleta, P.: *Co programátory ve škole neučí aneb Softwarové inženýrství v reálné praxi.* Computer Press, Brno, 2003. (CZ)

Annotation

The customer/developer relationship in software projects

The latest trends in software development area suggest even more needs of presence of customer-developer relationship. The need to improve the communication between these two subjects is just a logical ending of resident concurrent struggle of corporations, trying to develop the product in the shortest time. This article is dedicated to communication problems between both of the mentioned subjects, communication levels between them, as well as the role of customer in software development by itself. The essay further deals with the agile

methodics as a new trend of software development and explores the customer-developer relationship in extreme programming, the main representative of these techniques. In conclusion essay urges on existence of the customer-developer links which can in essential way make the software development in the future more effective.