OPLATÍ SA ROBIŤ OUTSOURCING NAPRIEK ZVÝŠENÝM KOMUNIKAČNÝM PROBLÉMOM?

Slová a činnosť nejdú dobre dovedna. Pozri len na prírodu: tá je v ustavičnej činnosti, no mlčí!

Igor Slotík

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

Abstrakt. *Pri vytváraní softvérového produktu, hlavne na začiatku jeho životného cyklu je veľká potreba komunikácie. Implementácia by sa ani poriadne nezačala, keby každý vývojár nevedel, čo má vyvíjať, do ktorých súčiastok môže zasahovať, kde a ako má ukladať zmeny vyvíjaného produktu, kto je jeho nadriadení. Nedostatok komunikácie a organizácie sa vráti v podobe konfliktných zásahov do zdrojových kódov, zmenami technológií, neschopnosťou dodržať kontrolné termíny, prípadne celkovým neúspechom vývoja softvérového produktu. Potreba komunikácie je o to citeľnejšia pri distribuovanom vývoji. Ak distribuované tímy majú od začiatku problém dohodnúť sa na dôležitých otázkach, vývoj softvérového produktu je ohrozený. V eseji sa chcem zamyslieť nad tým, kedy a či vôbec sa oplatí prácu distribuovať a ak áno, aké metódy a prístupy riešia komunikačný problém distribuovaného vývoja.*

Kľúčové slová: *Distribuovaný vývoj, outsourcing, komunikácia, spôsoby komunikácie, komunikačné problémy.*

Nebezpečenstvá distribuovaného vývoja

V modernom svete sa distribuovaný vývoj stal bežnou súčasťou vývoja softvéru. Používa sa dokonca aj mimo oblasti softvérového vývoja. Keďže v mnohých veciach sa odlišuje od

ne-distribuovaného vývoja, viažu sa k nemu aj rôzne špecifické problémy, ktoré priamo či nepriamo súvisia s komunikáciou. Pri hľadaní tých najlepších prístupov som sa nestretol s takým prístupom, ktorý by riešil všetky nebezpečenstvá a úskalia distribuovaného vývoja. Ale dá sa použiť taký prístup, respektíve kombinácia prístupov, ktoré pomôžu s riešením najnebezpečnejších problémov a tým uľahčia cestu distribuovaného vývoja. V nasledujúcej časti opíšem problémy a prístupy a nakoniec tieto prístupy zhodnotím.

Problémy distribuovaného vývoja a prístupy na ich riešenie

Strategické problémy

Súvisia s nutnosťou rozdeliť vývoj projektu medzi niekoľko strán. Rozdelenie medzi viaceré strany je obmedzené rôznymi zdrojmi, stupňom skúseností, infraštruktúrou a inými ohraničeniami. Ideálne by bolo keby strany mohli pracovať čo možno najviac nezávisle od seba a zároveň aby komunikácia prebiehala hladko, efektívne a flexibilne [1].

Kultúrne problémy

Pokiaľ by sa do distribuovaného vývoja zapojili strany z rôznych štátov prípadne kontinentov, treba počítať s kultúrnymi rozdielmi. Je úplne prirodzené, že ľudia z rôznych oblastí majú iný pohľad na podstatné veci, ako je potreba štruktúry, prístup k hierarchii, vnímanie času a komunikačné štýly. Aby som videl tieto rozdiely, nemusím chodiť ani za hranice Slovenskej republiky. V Bratislave mám možnosť vidieť jemné odchýlky v správaní ľudí z rôznych kútov Slovenska.

Pokiaľ ľudia z rôznych strán vedia o svojom rozdielnom prístupe a rešpektujú ho, nemalo by dôjsť k vážnym problémom. Skutočný kultúrny problém nastane vtedy, keď o svojich rôznych pohľadoch nevedia. Napríklad e-mail od človeka z oblasti, kde sa komunikuje veľmi formálne a priamo, by mohol byť považovaný za veľmi strohý alebo dokonca drzí v inej oblasti. Iné ponímanie času môže viesť k rôznej interpretácií vážnosti a naliehavosti hraničných termínov.

Kultúrny rozdiel môže viesť ku komunikačným problémom. Napríklad ak je človek zmätený z prijatého e-mailu, často ho ignoruje alebo si môže spraviť predčasný úsudok o charaktere alebo úmysle odosielateľa [1].

Nedostatočná komunikácia

Pri softvérovom vývoji, hlavne v jeho začiatočnom štádiu je potrebné dostatočné množstvo komunikácie. V tíme a vo firmách existujú dve navzájom sa dopĺňajúce formy komunikácie. Prvá je formálna, oficiálna komunikácia, ktorá potrebuje byť jasnou, zrozumiteľnou a dobre pochopiteľnou. Slabá, nejasná špecifikácia kritických úloh ako aktualizácia stavu projektu, pribúdanie problémov s vývojom a určenie zodpovedností medzi vývojármi je stratou času a vedie len k problémom.

Druhá forma komunikácie je neformálna, bežná komunikácia. Prekvapilo ma, že podľa výskumov je aj pri vývoji softvérového projektu veľmi dôležitá. Neformálna komunikácia dáva informácie o tom, čo sa deje naokolo, prehľad o vývoji, kto je za čo zodpovedný, kto je vo firme expertom na akú oblasť, v akom štádiu vývoja sú jednotlivé

časti produktu a mnoho iných dôležitých informácií, ktoré pomáhajú vývojárom efektívne pracovať. Nedostatok tejto komunikácie môže mať nepríjemné a prekvapujúce následky. Čím viac je projekt nejasný a problémový, tým viac je potrebná práve neformálna komunikácia.

Myslím, že ani nemusím zdôrazňovať, že práve nedostatok neformálnej komunikácie je veľkým nedostatkom distribuovaného vývoja [1].

Zdieľanie znalostí

V distribuovanom vývoji je dôležitý aj manažment podporujúci zdieľanie znalostí, bez ktorého by nemohli byť rozvinuté výhody distribuovaného vývoja. Ak manažéri nevhodne šíria stav projektu medzi vývojármi, tímy nemôžu zistiť, ktoré úlohy sú na kritickej ceste. Potreba odborného posudku by mala byť prístupná, ale nemala by byť sústredená len pre niektorých ľudí. Taktiež malé znalosti a slabý informačný manažment môžu zapríčiniť tímom zmeškanie príležitostí na znovupoužitie, čo by im ušetrilo veľa času a námahy. Neefektívny vývoj viacerých skupín môže byť zapríčinený taktiež slabou dokumentáciou. V distribuovanom vývoji musí byť navyše jasné, ktorý tím dokumentáciou používa a ktorý ju vypracoval [1].

Synchronizačné problémy

Kritickým bodom distribuovaného vývoja je aj synchronizácia. Napríklad, ak vývojový tím a testovací tím si predstavujú pod Unit testami niečo iné, musia riešiť synchronizačné problémy. Synchronizácia vyžaduje všeobecné definovanie medzníkov a jasné prístupné a výstupné kritéria. Efektívne použitie synchronizácie v distribuovanom vývoji často sťažujú miznúce požiadavky, nestála špecifikácia, nedostatok kvalitných nástrojov podporujúcich súčinnosť v rôznom čase a priestore, nedostatok neformálnej komunikácie [1].

Technické problémy

Ďalším problémom môže byť používanie nekompatibilných dátových formátov a rôznych verzií toho istého nástroja. Tieto problémy voláme technickými problémami [1].

Použitie kvantitatívneho a komponentového prístupu na spomenuté problémy

V rešerši som podrobne rozobral dva prístupy, ktoré riešia rôzne špecifické problémy distribuovaného vývoja. Prvý je kvantitatívny prístup. Jeho hlavným cieľom je nájdenie spoľahlivých expertov na rôzne úlohy pri vývoji. Myšlienkou komponentového prístupu je dekompozícia projektu, aby mohli tímy na rôznych stranách pracovať na projekte čo možno najviac nezávisle.

V tabuľkách 1 a 2 je vidieť, ktoré problémy distribuovaného vývoja podľa môjho názoru kvantitatívny a komponentový prístup ovplyvňujú a ktoré nie. Ďalej uvediem dôvody, prečo si to myslím. Výsledok je dosť subjektívny, pretože mnohé problémy sú ovplyvňované kvantitatívnym alebo komponentovým prístupom nepriamo. Takže o tom, čo ktorý prístup rieši by sa dalo dokonca aj diskutovať.

Tab. 1. Prístupy distribuovaného vývoja.

	Problém nájsť správneho experta	Strategické problémy	Kultúrne problémy	Nedostatočná komunikácia
Kvantitatívny prístup	Áno	Áno	Nie	Áno
Komponentový prístup	Nie	Áno	Nie	Áno

Tab. 2. Prístupy distribuovaného vývoja.

	Manažment znalostí	Synchronizačné problémy	Technické problémy
Kvantitatívny prístup	Áno	Nie	Nie
Komponentový prístup	Nie	Áno	Áno

V nasledujúcich častiach pre kvantitatívny a komponentový prístup uvediem, prečo si myslím, že spomenutými problémami sa venujú a prečo nie.

Kvantitatívny prístup

Zaoberá sa tým, ako nájsť správneho experta. Ľudia v tíme sú hodnotení na základe množstva získaných skúseností. Základnou myšlienkou je vyprodukovať krátky zoznam odporúčaných expertov na základe heuristík špecifikovaných žiadateľom. Ak sa nenájde žiadny vyhovujúci expert, môže sa vygenerovať širší list potencionálnych expertov zmenou prahových hodnôt heuristík. Na efektívne využitie kvantitatívneho prístupu je potrebná aplikácia, ktorá by slúžila ako prehliadač znalostí ľudí v tíme [3].

Keď vieme, ktorá strana má v danej oblasti lepších expertov, vedeli by sme, ktorú časť projektu priradiť ktorej strane. Preto si myslím, že kvantitatívny prístup pomáha riešiť strategické problémy.

Kvantitatívny prístup sa môže považovať za podporný prostriedok manažmentu znalostí, pretože podporuje celkový prehľad o odborných skúsenostiach účastníkov projektu. Manažéri potrebujú predsa rýchlo zistiť, kto je v čom dobrý, aby mohli úlohy rozdeliť tak, aby boli vykonávané čo najefektívnejšie. A navyše, vývojárom môžu byť informácie o odborných znalostiach užitočné vtedy, keď potrebujú rýchlo s niečím poradiť a pomôcť.

Keďže nie je potrebnej toľko komunikácie o tom, kto je na čo odborník, môžem povedať, že do istej miery rieši problém nedostatočnej komunikácie. Ľudia sa nemusia pýtať a zisťovať, kto je v čom dobrý. Na to slúži prehliadač odborných znalostí.

Nájdenie správneho experta podľa môjho názoru nesúvisí s kultúrnymi, synchronizačnými a technickými problémami.

Komponentový prístup

Rozdeľuje projekt na nezávislé komponenty, aby sa komponenty mohli vyvíjať rôznymi stranami [2]. Pomáha riešiť teda strategické problémy, pretože tie priamo súvisia s rozdeľovaním projektu medzi rôzne strany.

Komponentový prístup sa snaží riešiť problém nedostatočnej komunikácie medzi stranami. Páči sa mi, že na začiatku je centralizovaná analýza a dizajn, ktorého sa zúčastnia všetci účastníci. Analyzujú sa požiadavky na projekt a vytvorí sa maketa projektu a špecifikácia inter-operability. Pri špecifikácií inter-operability je snaha predvídať problémy s prepojiteľnosťou komponentov, ktoré môžu nastať pri vývoji, a definovať štruktúru (*framework*), ktorá by tieto problémy spracovávala [2].

Ak je projekt dobre dekomponovaný, znamená to, že strany medzi sebou nemusia toľko komunikovať po centralizovanej analýze a dizajne. Je podľa mňa ideálne, stretnúť sa zoči-voči a snažiť sa dohodnúť na všetkých dôležitých veciach na začiatku životného cyklu projektu, aby v budúcnosti komunikácia nebrzdila vývoj.

Dekomponovanie projektu do istej miery rieši aj synchronizačné problémy. Ak je tím schopný pracovať na svojom komponente čo možno najviac nezávisle, nie je potrebné toľko synchronizácie.

Keďže komponentový prístup umožňuje používať rôzne nástroje a platformy medzi tímami, pretože vývoj jedného komponentu by mal spadať pod jeden tím, technické problémy by sa nemali objaviť medzi tímami

Rozdeľovanie projektu na viaceré nezávislé komponenty nepomáha s kultúrnymi rozdielmi, problémom ako nájsť experta ani manažmentom znalostí.

Zhodnotenie

Je zrejmé, že žiadnym prístupom sa nedá vyhnúť všetkým úskaliam. Avšak pri analýze prístupov som zistil, že sú nápomocné s viacerými problémami. Podľa tabuliek č. 1 a č. 2 je vidieť, že kvantitatívny s komponentovým prístupom ovplyvňujú takmer všetky dôležité problémy spomenuté v tejto eseji. Pri kvalitnom použití kvantitatívneho a komponentového prístupu je možné rozvinúť výhody distribuovaného vývoja.

V ďalšej časti je možné sa zamerať na analýzu ďalších prístupov na riešenie problémov distribuovaného vývoja, prípadne na hlbšiu analýzu kvantitatívneho a komponentového prístupu.

Použitá literatúra

1. Herbsleb, J., D., Moitra, D.: Global Software Development. In.: IEEE SOFTWARE, 2001, s. 17 – 18
2. Ioannidou, A., Payton, M., Repenning, A., Roschelle, J., Ye, W.: Using Components for Rapid Distributed Software Development. In: Global Software Devlopment, 2001, s. 38-45
3. Herbsleb, J., D., Mockus, A., : A Quantitative Approach to Identifying Expertise. In: Expertise Browser, 2002, s. 503 – 506

Annotation

It is worth doing outsourcing despite the increased of communication problems?

When creating a software product, especially at the beginning of its life cycle is a great need for communication. Implementation would not even really begun, if every developer did not know what has developed into components which can interfere with where and how to save changes to the developed product, who are his superiors. Lack of communication and organization will return in the form of conflict intervention in the source code, changes in technology, failure to meet inspection deadlines, or total failure of software development. Need for communication is particularly appreciable for distributed development. If you have distributed teams since the beginning of the problem to agree on important issues, software product development is threatened. In the essay, I want to think about when and even whether it is worthwhile to distribute the work and if so, what methods and approaches to solve the communication problem of distributed development.