PROBLÉMY PODPORY VÝVOJA PRI JEHO GLOBÁLNEJ DISTRIBÚCII

Poslať časť vývoja do Indie je pekné, no aké problémy to prináša?

Marek Tomčo

Slovenská technická univerzita
Fakulta informatiky a informačných technológií
Ilkovičova 3, 842 16 Bratislava
xtomco@fiit.stuba.sk

Abstrakt. *Tvorba softvéru bola už od svojich začiatkov vždy dynamický proces plný zmien, úprav a kompromisov. Vývoj zložitého softvérového systému v tíme skladajúceho sa z desiatok, niekedy až stoviek pracovníkov by nebol v takomto prostredí možný bez kvalitnej podpory vývoja. Rastúci tlak zo strany konkurencie v danom odvetví prinucuje firmy hľadať optimálne možnosti prevádzky. Presun časti vývoja do zahraničných pobočiek prináša jasné výhody, ako zvýšenie rýchlosti vývoja alebo zlepšenie kvality práce správnym rozložením úloh. Avšak toto riešenie so sebou prináša množstvo nových problémov vďaka nemožnosti priamej komunikácie, rozličnosti časových zón alebo kultúr. V tejto eseji sa zamyslím nad dôležitosťou podpory vývoja v takomto prostredí, a to či už všeobecne, tak pomocou ukážok získaných z praxe profesionálnou softvérovou spoločnosťou.*

Kľúčové slová: *podpora vývoja, manažment konfigurácií, distribúcia vývoja*

Globálna distribúcia vývoja?

Na úvod treba vysvetliť, prečo sa pri niektorých projektoch preferuje časti vývoja distribuovať do viacerých pracovísk namiesto štandardného vývoja v lokálnom prostredí. Dôvodov môže byť viacero. V takto veľkom a hlavne veľmi dynamickom sektore ako je IT, sa dá udržať na vrchole trhu len veľmi ťažko. Malou zmenou v dopyte sa môže úspešná

svetová firma dostať nedostatočným prispôsobením sa trhu na pokraj záujmu v priebehu pár rokov. Z tohto dôvodu musí mať každá úspešná firma dostatočnú odvahu a prostriedky nielen na tvorbu a údržbu softvéru, ale aj na jeho neustálu evolúciu. To sa najlepšie dosiahne práve vytvorením osobitných oddelení pracujúcich nezávisle na sebe, mnohokrát práve v iných krajinách, či už pre lacnejšiu prevádzku alebo nedostatok pracovnej sily v blízkom okolí. V prípade začlenenia takéhoto oddelenia a jeho dosiahnutých výsledkov do hlavnej vývojovej vetvy sa však dostávame k načrtnutému problému so synchronizáciou vývoja jedného produktu dvomi vzdialenými oddeleniami.

Podobný problém nastáva pri spolupráci dvoch svetových firiem za účelom kombinácie ich poznatkov do práce, ktorá by bola pre nich v prípade samostatnej práce nesplniteľná. To sa môže stať už v menšom merítku. Napríklad dve menšie firmy spoja svoje sily na dostatočnú konkurencieschopnosť väčšej firme alebo spojenie viacero svetových firiem pri vývoji nových, prelomových technológií.

V neposlednom rade sa nesmie zabudnúť na prípad vývoja systému s otvoreným zdrojovým kódom, kde viacero jednotlivcov z celého sveta spája svoje časti práce do jednotného produktu. Takýto vývoj zväčša obsahuje menšie množstvo dlhodobo aktívnych vývojárov, no zložitosť synchronizácie sa dostavuje rôznorodosťou pôsobísk a časov práce jednotlivých členov.

Je takýto manažment vážne nutný?

Predstavme si situáciu, že sa softvérová firma odrazu rozhodne rozdeliť svoju naplánovanú prácu do dvoch častí, napríklad do oddelených modulov, ktoré budú tvorené rozdielnymi pobočkami na opačných stranách pologule. Vzhľadom nato, o koľko lacnejšie je v súčasnej dobe zamestnať vývojárov z menej pokročilých krajín, nie je tento príklad vôbec hyperbolou. V menšom merítku to je presun vývoja z Nemecka na Slovensko, vo väčšom napríklad z USA do Indie.

V uvedenom príklade nastane problém hneď v začiatkoch. Napriek perfektnému plánu vývoja a analýzy vrstiev je možnosť nutnosti úpravy plánu takmer neodvratná. To môže vyžadovať zmenu implementácie prepojenia modulov, ktorú potrebujú jasne poznať na oboch stranách vývoja. Návrhy na úpravu a zjednodušenia implementácie zapadajú do tej istej kategórie.

Čo sa stane, keď sa jedna časť vývoja dostane do pozície, že ich implementovaná časť je pripravená na skúšku, no bez dokončenej druhej časti nie je vykonateľná? Alebo, že sa o rozhodnutí menšej zmeny komunikácie modulov dozvedia, čo by len o pár dní neskôr, kvôli chybe v komunikácii? Všetko toto v praxi znamená stratený čas, ktorý mohol byť venovaný produktívnejšej činnosti, tým pádom predĺženie času vývoja a jeho nákladov. Kategóriou samou o sebe je spájanie individuálnych modulov do fungujúceho celku, čo niekedy, najmä rozličnosťou použitých knižníc rôznych verzií, vôbec nemusí byť krátkodobá záležitosť. Po implementácii a testovaní ešte zostáva nutnosť ohlásenia možných chýb na oddelenie zodpovedné za daný chybný modul.

Na synchronizáciu tohto všetkého by bez kvalitného manažmentu podpory vývoja bolo potrebné obrovské množstvo komunikácie. Bolo by jej dokonca toľko, že by ním bola pokrytá väčšina času venovaná implementácii a bola by potrebná všetkými členmi tímu. Štúdia v takomto prostredí od firmy Perry, Staudenmayer, and Votta [3] ukázala, že

vývojár, t.j. nie pracovník zodpovedný za komunikáciu, ale samotný tvorca, mal denne v priemere 75 minút neplánovanej komunikácie. To je 75 minút, ktoré mohli byť každý deň využité na samotné programovanie, a to u každého vývojára! Nehovoriac o tom, že sa bavíme o globálnej distribúcii. Treba si teda klásť otázky: "Koľko je teraz u nich hodín?", "V akom čase sa u nich pracuje?" alebo dokonca "Dorozumiem sa s nimi?". Zaujímavým faktom je taktiež skutočnosť, že počas pokročilejších fáz vývoja je pri vzdialenej komunikácii omnoho menšia úroveň získanej pomoci, čo môže poukazovať na komplikovanosť nepriamej pomoci v záverečných fázach. Vynechaním podpory vývoja sa v skratke presúva omnoho viac potrebného úsilia do komplikovanej oblasti globálnej komunikácie.

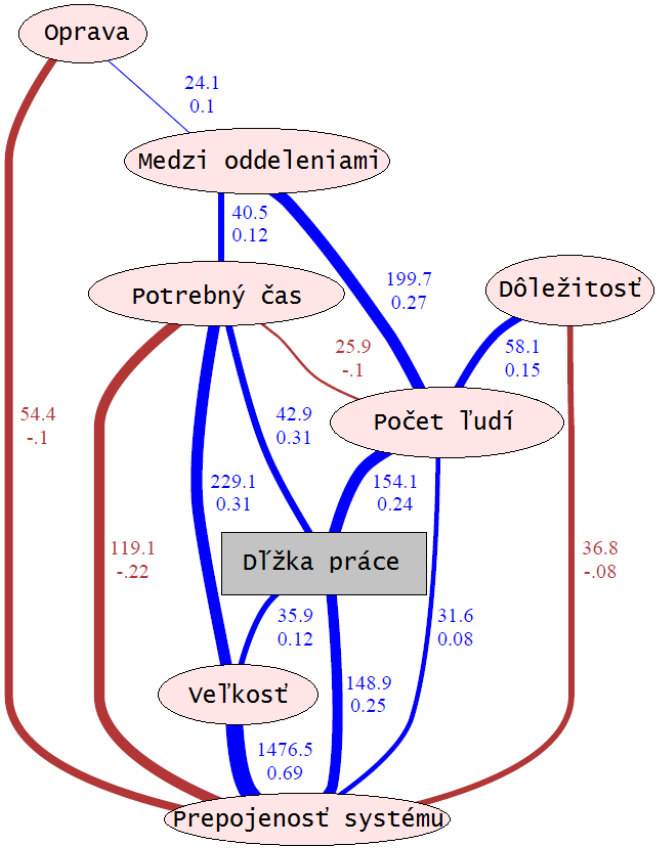
Problémové časti pri rozdelení vývoja

Základom každej správnej podpory vývoja je navrhnutie kvalitného spôsobu ukladania, verziovania a aktualizácií artefaktov ako je zdrojový kód, dokumentácia, konfiguračné súbory a podobne. Na to sú už v dnešnej dobe pripravené systémy ako Git alebo Subversion. Do takýchto pripravených repozitárov je bežné pri každej zmene kódu týkajúcej sa ako jeho doplnenia, tak opravy, najprv podať požiadavku na modifikáciu. Je ale dĺžka práce na takejto požiadavke podobná aj pri jej rozdelení do viacerých oddelení? A ak nie, čo jej vykonanie v takomto prostredí spomaľuje?

Túto otázku si kládli aj zamestnávatelia vo firme Lucent Technologies [1]. Na zistenie odpovede zorganizovali prieskum pomocou dotazníka pre 160 zamestnancov v pobočkách vo Veľkej Británii, Nemecku a Indii. Kládlo sa viacero otázok, medzi nimi aj ako dlho trvala práca na takejto požiadavke od začatia práce až po poslednú modifikáciu. Požiadavky sa rozdelili do dvoch skupín. Na tie, ktoré boli smerované lokálnemu oddeleniu a tie, ktoré bolo potrebné vyriešiť v spolupráci so vzdialeným oddelením. Výsledky jasne ukázali výrazné zdržania pri požiadavkách na viaceré oddelenia. Priemerná dĺžka vybavenia požiadavky v lokálnom oddelení trvala v priemere 5 dní, zatiaľ čo v prípade požiadavky na vzdialené oddelenie jej vybavenie trvalo v priemere až 12.7 dní. Ukázalo sa teda, že požiadavky do vzdialeného prostredia trvali v priemere až 2.5-krát dlhšie na svoje vypracovanie. Čas na plánovanie vykonania požiadavky a vyhodnotenie jej priority nebol v tomto čakaní zarátaný, no jeho dĺžka sa nijako výrazne medzi požiadavkami na lokálne a vzdialené oddelenie nelíšila.

Lucent Technologies však pri tomto zistení s prieskumom neskončilo. Potrebovali zistiť, čo je hlavnou príčinou tohto zdržania. Boli identifikované základné možné príčiny zdržania: počet ľudí podieľajúcich sa na práci na požiadavke, prepojenosť požiadavky s rôznymi časťami systému, veľkosť potrebnej zmeny, dĺžka času na vytvorenie prvej zmeny, podmienka, či je to požiadavka na opravu chyby (náprava chyby sa brala ako špeciálny typ požiadavky s možnými rozdielnymi časmi), dôležitosť požiadavky a rozptýlenosť práce do rozdielnych oddelení. Po bližšom preskúmaní došli k výsledkom, ktoré ukázali, že ako najväčší dôvod k predĺženiu práce na požiadavke prekvapivo nebola podmienka jej rozptýlenosti do rôznych oddelení, ale jej veľkosť, prepojenosť so systémom a množstvo ľudí podieľajúcich sa na nej. Výsledok skúmania bol graficky znázornený na Gaussovom modeli (Obr. 1.), so znázornením spojení, ktoré majú výraznejší vzájomný vzťah súvislosti (červené negatívny a modré pozitívny).

Tieto výsledky vyšli napriek predchádzajúcemu zisteniu o zvýšenej dĺžke práce pri požiadavkách zahrňujúcich viacero oddelení. Je preto možné sa domnievať, že požiadavky vyžadujúce väčšie zmeny, prepojené s väčšou časťou systému alebo zahrňujúce viacero pracujúcich ľudí, majú väčšiu pravdepodobnosť rozdelenia medzi viaceré oddelenia a sú v tomto prostredí vývoja najčastejším dôvodom predĺžených vypracovávaní požiadaviek. Je teda logické, že najlepším spôsobom by mal byť vývoj na tieto prípady už dopredu pripravený správnym rozdelením úloh. V prípade, že sa takáto možnosť nedá realizovať, je na manažmente podpory vývoja, aby pomohlo pri zaistení čo najrýchlejšej a bezproblémovej implementačnej časti vývoja správnym verziovacím systémom. Táto oblasť je obzvlášť kritická pri požiadavkách s veľkým prepojením v systéme. Mnohé prepojenia do systému znamenajú väčšiu možnosť modifikácie jednej z prepojení, o ktorých je nutné sa dozvedieť čo najskôr a verziovací systém s notifikáciami zmien je tá najjednoduchšia možnosť, ako ich na to upozorniť. Bolo by určite príjemnejšie a užitočnejšie dozvedieť sa o prípadných zmenách zasahujúcich do mojej vypracovávanej úpravy formou správy z úložiskového systému, než neskôr nutnou priamou komunikáciou.



Obr. 1. Grafický model závislosti dĺžky práce na požiadavke od charakteristiky požiadavky. [1]

Globálne rozdelenie vývoja v praxi

V ďalšej časti by som chcel priblížiť situáciu z praktického hľadiska, porovnaním niekoľkých vybraných spôsobov, ako boli problémy s podporou vývoja v danom prostredí riešené na špecifických projektoch. Nemenovaná IT firma situovaná v Brazílii v roku 2006 uverejnila dokument [2] o svojich skúsenostiach s globálne distribuovaným vývojom a poznatkami, ku ktorým svojou prácou došli.

V prvom zo svojich globálne distribuovaných projektov pracovali s pobočkou z USA, pričom implementácia bola rozdelená v každej z častí systému. Vzhľadom nato, že ani jedna pobočka nemala predchádzajúce skúsenosti distribúciou vývoja, bola zvolená oddelenosť manažmentu softvérovej konfigurácie (*angl. Software configuration management - SCM*). Navyše brazílska časť mala v pláne riadiť sa vývojovým modelom CMM (*Capability Maturity Model*), ktorého podstatnou súčasťou sú práve presne definované procesy SCM. Na spájanie implementačných častí z oboch oddelení bol špeciálne vybraný jeden pracovník.

Je jednoduché si domyslieť, čo sa v takomto prostredí stalo. Prvý problém, ktorý vyšiel najavo už v skorých fázach vývoja, bolo nesprávne rozdelenie úloh do oddelení. Americký tím v niektorých úlohách nestíhal dokončiť naplánované časti implementácie načas a vzhľadom na ich prepojenosť na úlohy v brazílskom tíme spomaľoval rýchlosť vývoja. To jasne odpovedá zisteniam spomenutým v predchádzajúcej sekcii. Je nutné naplánovať si maximálnu nezávislosť medzi úlohami rozdelenými medzi viaceré oddelenia. Rozhodnutie viesť si v každom oddelení vlastný SCM priniesol taktiež viaceré komplikácie. V prvom rade človek zodpovedný za synchronizáciu implementácie musel stráviť spájaním častí každý týždeň priemerne 3 až 4 hodiny. Najväčší problém bol hlavne so súbormi editovanými oboma oddeleniam, ako konfiguračné súbory alebo používané zdroje. Ručná editácia takýchto súborov je veľmi náchylná na chyby, ktoré môžu v horšom prípade dočasne zastaviť vývoj, napríklad v prípade nefunkčnosti spustenia systému. Je teda jasné, že automatizácia týchto procesov, použitie jediného verziovacieho a konfiguračného systému je v praxi podstatná výhoda.

Vo svojom druhom projekte firma pracovala s pobočkami v USA a Indii. Po poučení sa z prvého projektu bolo zvolené jednotné SCM prostredie. Brazílska pobočka opäť pracovala vývojovým modelom CMM, zatiaľ čo zvyšné na tento model neboli dostatočne rozvinuté. Z tohto dôvodu bol ako podpora pre všetky oddelenia špeciálne zvolený koordinátor. Tento koordinátor pomáhal zvyšným oddeleniam s inštaláciou produktu, synchronizáciou a validáciou implementácií a mal presný prehľad o naplánovaných časových termínoch ohľadom vývoja. Rozdelenie práce bolo navyše jasne pridelené s dôrazom na samostatnosť implementácií v jednotlivých oddeleniach.

Vo výsledku sa výber jediného SCM prostredia ukázal ako správna voľba. Napriek každodenným testovaniam implementácie dochádzalo iba k ojedinelým synchronizačným chybám. Jeden koordinátor sa však pre tri vzdialené oddelenia ukázal ako nedostatočný vzhľadom na vysoký počet rozdielnych stavov implementácií a termínov. Z toho jasne vidieť, že pre každé vzdialené oddelenie je najlepšie používať vlastného koordinátora, ktorý bude vedieť o činnostiach vo svojom oddelení a o externých záležitostiach získa informácie od iných koordinátorov. Počas vývoja taktiež nastal problém so súbormi na konfiguráciu. Systém mal v konfiguračných súboroch databázové skripty, ktoré

upravovali súbory nezahrnuté v konfiguračných súboroch. Konfiguračné súbory však mali mať previazanosť len medzi sebou a nemali by byť závislé od súborov nezahrnutých v manažmente konfigurácií. Týmto spôsobom by sa teda dalo aplikačné prostredie reprodukovať len použitím konfiguračných súborov. Pri takýchto skriptoch by bolo možné riešenie externé súbory namiesto ich úprav nanovo vytvárať. Pokiaľ by sa za týchto podmienok nemohlo z konfiguračných súborov prostredie vytvoriť, je potrebné nanovo prehodnotiť, ktoré súbory do manažmentu konfigurácií zaradiť.

V treťom projekte sa vytváralo nové rozhranie do existujúceho systému a boli v ňom zahrnutí dvaja ľudia z Brazílie a dvaja z USA. Implementácia sa medzi tímy rozdelila do nezávislých modulov. Tímy sa dohodli na práci podľa modelu CMM a každý z nich mal vlastného koordinátora, ktorý validoval implementovanú funkcionalitu.

Problém sa ukázal v slabej komunikácii na začiatku projektu, keď sa jasne nestanovili zodpovednosti koordinátorov a rozsahy ich vyhodnocovania. Z tohto dôvodu v pokročilejších častiach vývoja dochádzalo k nezrovnalostiam pri vyhodnocovaní kvality implementovanej funkcionality. Toto jasne poukazuje na to, že napriek skúsenostiam zúčastnených strán je na začiatku projektu potreba dohody kvalitného SCM. Problémy taktiež nastali kvôli nedostatočnému plánovaniu. Plány ďalšej práce boli získavané aktuálne, podľa potreby a v prípade zaneprázdnenosti zadávateľa bolo na ne treba čakať. Kvôli týmto problémom nastali mnohé neplánované zdržania vývoja. Z toho vyplýva, že plán základných prác a jeho zdokumentovanie v SCM je najlepšie vykonať čo najskôr. Ďalšie problémy vznikli pri preplánovaniach rozsahu činností jednotlivých tímov, najmä vďaka vzniknutým meškaniam. Toto preplánovanie spôsobilo rozdelenie jedného modulu do oboch tímov, čo zapríčinilo už viackrát spomenutý problém previazanosti práce medzi tímami. V takýchto prípadoch je najlepšie jasne zanalyzovať previazanosť rozdeleného modulu a naplánovanie jeho implementácie do tímov s čo najväčšou nezávislosťou rozdelených úloh.

Ako vidno, v týchto skúsenostiach je jasná podobnosť s poznatkami zistenými z prieskumov, o ktorých bola reč vyššie. Najväčší problém bol a vždy bude previazanosť prác rozdelených medzi vzdialené oddelenia. Kvalitný SCM tento problém dokáže iba zmenšiť, no nie eliminovať. Je dôležité si oddelenosť vývoja naplánovať hneď na začiatku, ináč hrozí možnosť zdržania vývoja z dôvodu čakania na druhý tím a nutnosť preplánovania časti úloh. Rozdelenie SCM je taktiež príčina problémov vzhľadom na zložitosť ručnej synchronizácie zmenených súborov. Preto je doporučené pracovať dohromady len s jedným SCM. Práca s SCM by mala byť jasne stanovená a v ideálnom prípade totožná v každom oddelení. V opačnom prípade to znamená viac práce pre pracovníkov zodpovedných za koordinácie tímov. Čím viac ľudí, prípadne čím dlhší čas sa na systéme robí, tým viac sa môžu problémy v synchronizácii vyplavovať na hladinu, preto treba brať tento faktor taktiež do úvahy.

Záver

Ako na globálnych skúmaniach, tak na prípadoch z praxe je jasne vidno, že vývoj rozdelený do vzdialených oddelení potrebuje špeciálny prístup ku podpore vývoja. Podpora vývoja v takomto prostredí dosahuje omnoho vyššiu dôležitosť z toho dôvodu, že v mnohých prípadoch je to tá najrýchlejšia, najjednoduchšia a niekedy aj jediná cesta k

zisteniu, ako je na tom práca v susednom oddelení. Vzhľadom na rozrastanie IT priemyslu do celého sveta je pravdepodobné častejšie použitie tohto typu vývoja, či už z finančného alebo praktického dôvodu. Je preto veľmi užitočné uchovať všetky sprostredkované skúsenosti z tejto oblasti, ktoré môžu radikálne pomôcť pri práci v týchto podmienkach na budúcich projektoch.

Použitá literatúra

1. James D. Herbsleb, Audris Mockus, Thomas A. Finholt, and Rebecca E. Grinter. 2001. An empirical study of global software development: distance and speed. *In Proceedings of the 23rd International Conference on Software Engineering (ICSE '01).* IEEE Computer Society, Washington, DC, USA, 81-90.
2. Leonardo Pilatti, Jorge Luis Nicolas Audy, and Rafael Prikladnicki. 2006. Software configuration management over a global software development environment: lessons learned from a case study. *In Proceedings of the 2006 international workshop on Global software development for the practitioner (GSD '06).* ACM, New York, NY, USA, 45-50
3. Perry, D. E., N. A. Staudenmayer, and L. G. Votta. People, Organizations, and Process Improvement. *IEEE Software 11*, 4, 1994, 36-45.

Annotation

Problems with development support in globally distributed environment

Software development was since the beginning dynamic process full of changes, adjustments and compromises. Development of complex software system in a team consisting of dozens, sometimes hundreds of workers would not be possible in such an environment without good development support. The increasing pressure from competitors in the industry makes companies find better possibilities of bussines. Transfer of development parts to foreign subsidiaries brings big benefits, such as increased speed of development or improvement of quality by correct distribution of tasks. However, this solution brings many new challenges due to the impossibility of direct communication, diversity of cultures and time zones. This essay talks about importance of development support in such environment in both, general thoughts and practical examples from experience of professional software company.