Analýza a plánovanie rizík v softvérovom projekte - extrémne programovanie a manažment rizík

MICHAL KOBZA

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

Abstrakt. Jedným zo základných problémov pri vývoji softvéru je riziko. Riziko, že vývoj zlyhá a nedodá požadované výsledky. Toto zlyhanie môže mať obrovský dopad na ekonomický a ľudský potenciál aj pre veľké softvérové firmy. Úlohou analýzy a plánovania rizík v softvérovom projekte je identifikovať riziká včas a ich plánovaním minimalizovať dopad na výslednú kvalitu softvérového produktu. Esej poskytuje stručný prehľad najčastejších rizík. Uvádza spôsoby, ako sa s rizikami snažia vysporiadať metodiky extrémneho programovania a softvérového inžinierstva. Pri porovnávaní uvedených spôsobov sa zameriava na ich vhodnosť a použiteľnosť pre malé tímy podobné tímom vytvoreným v rámci tímového projektu na FIIT STU.

Trochu z histórie

Pre pochopenie, akú dôležitú úlohu zohrávajú riziká v tvorbe softvérových projektov, je dobré sa pozrieť späť na obdobie tzv. softvérovej krízy, ktorá vlastne predchádzala vzniku softvérového inžinierstva. Jej charakteristickými znakmi bolo neúnosné predlžovanie a predražovanie projektov, nízka kvalita programov, nemožnosť, prípadne neúnosná finančná náročnosť údržby a inovácií, nízka produktivita programátorov a podobne.

Ako príklad sa uvádza historka, podľa ktorej zo všetkých softvérových zákaziek pre americkú vládu, ktoré mali hodnotu miliónov dolárov, bolo viac ako 40% dodaných, ale nikdy úspešne použitých. Asi 25% zákaziek bolo zaplatených, ale nikdy nedodaných a asi 15% po drastických úpravách zahodených. Asi len necelé tri percentá projektov boli po úpravách použité a iba dve percentá mohli byť použité bezo zmien.

Dôvody vtedajšej krízy boli rôzne, bolo ich veľa, a čo je najhoršie, „pekne“ sa kombinovali a podporovali. Vo všeobecnosti možno povedať, že tieto dôvody sa aj

Manažment v softvérovom inžinierstve,október 2006, s. 1-8.

dnes považujú za hlavné problémy, ktoré môžu nastať pri tvorbe softvérových projektov. Niektoré hlavné dôvody sú:

- Zlá komunikácia na všetkých úrovniach, hlavne zákazník – analytik (aj keď vtedy priamo funkcia „analytik“ neexistovala). Dnes sa tento dôvod identifikuje ako riziko nepochopenia zadania.
- Nesprávny prístup jednotlivých osôb k vývoju softvéru, programátor bol prílišná individualita a ignoroval tím. Dnes sa na toto riziko, identifikované ako riziko ľudského faktoru, prihliada veľmi vážne a je snaha ho eliminovať už pri prijímaní nových ľudí do tímu. S týmto rizikom úzko súvisí aj riziko fluktuácie pracovníkov. Dneska sa už berie ako pravidlo, že pridaním ďalších ľudí do meškajúceho projektu spôsobíme iba ďalšie meškanie.
- Nesprávne odhady, a to vo všetkých smeroch: odhady času, ceny a rozsahu projektu. Tento dôvod možno identifikovať ako riziko meškania harmonogramu, ktoré môže viesť až k zrušeniu projektu.
- Zlé plánovanie – bolo veľmi obtiažne vypracovať taký plán projektu, ktorý by bol akceptovateľný zákazníkom aj programátormi (to je vlastne problematické dodnes). Tento dôvod sa dnes premieta do rôznych rizík, napríklad už do spomínaného rizika meškania harmonogramu.
- Veľmi nízka produktivita programátorov – programátori sa zaoberali všetkým možným len nie tým, čo potreboval zákazník. Tento dôvod je možno identifikovať aj ako riziko nadbytku funkcií (veľa pekných funkcií, žiadna použiteľná zákazníkom).
- Podcenenie hrozieb, ktoré sa pri vývoji vyskytovali. Mnohé mohli byť odstránené už v zárodku a s minimálnymi nákladmi. Dnes je snaha všetkými silami čo najskôr identifikovať a analyzovať možné hrozby (riziká), a plánovaním znížiť ich prípadný dopad. Rôzne metodiky však majú rôzne názory a prístupy k tomu, ako to čo najefektívnejšie dosiahnuť.

Definícia rizika

Riziko je ako pojem vo Websterovom slovníku vysvetlené ako možnosť utrpieť stratu, poškodenie alebo zničenie. Za najvýznamnejšie riziká sa dnes vo všeobecnosti považujú tieto (Boehm):

- Nedostatok personálu.
- Nerealistické rozvrhy a rozpočty.
- Vytvorenie inej alebo zlej funkcionality.
- Vytvorenie nevyhovujúceho používateľského rozhrania.
- Pozlátenie systému.

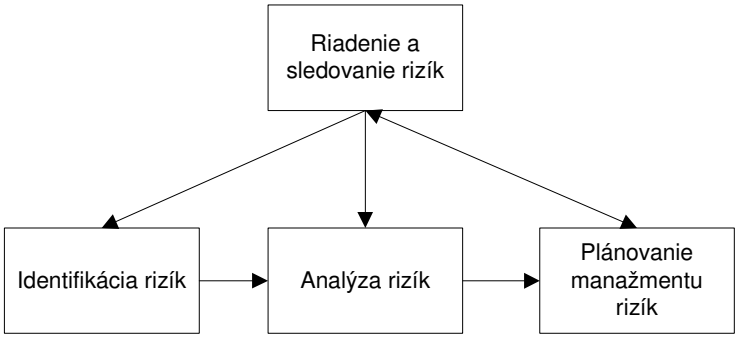
- Spojité zmeny požiadaviek.
- Nedostatky v externe vytvorených moduloch.
- Nedostatky v externe zabezpečovaných úlohách.
- Nedostatky výkonu v reálnom čase.
- Precenenie technológie.

V nasledujúcich riadkoch je vysvetlené, ako sa s rizikom vysporadúva softvérové inžinierstvo a extrémne programovanie.

Prístup k riziku podľa softvérového inžinierstva

Softvérové inžinierstvo obsahuje samostatnú časť, venovanú manažmentu rizík v softvérových projektoch [3]. Snaží sa nájsť a popísať spôsoby, ako identifikovať, analyzovať, zdokumentovať, roztriediť, klasifikovať a riadiť a zasa analyzovať všetky možné riziká, ktoré by mohli pri tvorbe softvérového systému nastať.

Základný model manažmentu rizík je zobrazený na obrázku 1 [2]. Jeho hlavnou časťou je riadenie a sledovanie rizík. Kladie si za cieľ správne reagovať na rizikové udalosti. Na udalosti a riziká vopred identifikované a zanalyzované. Ak nejaké takéto riziko nastane, proces riadenia rizika určí opravné akcie na zabezpečenie reakcie. Je však nepravdepodobné, že by i tá najlepšia analýza zabezpečila identifikáciu všetkých možných rizík, to vlastne uznáva aj manažment riadenia rizík. Preto sa snaží postupovať iteratívne a počas celého života projektu identifikovať a analyzovať možné riziká.



Obr. 1: Model manažmentu rizík.

Ako prvé sa v manažmente rizík riziká identifikujú. Identifikovať riziká znamená určiť, aké rôzne riziká môžu ohroziť projekt a zdokumentovať ich charakteristiky. Pre každé riziko sa určí pravdepodobnosť, že nastane a odhad rozsahu škôd pri výskyte udalosti. Pravdepodobnosť nastatia udalosti aj rozsah škôd sa neurčuje presnými hodnotami, ale v stupnici hodnôt ako napr. vysoká, nízka, veľmi nízka a podobne. Je

však zrejmé, že stanovenie pravdepodobnosti ako aj rozsahu možných škôd môže byť náročné, vlastne niekedy aj nemožné. Vtedy sa však podľa softvérového inžinierstva nejedná o riziko, ale o neurčitosť. Identifikáciu rizík je treba vykonávať počas celého života projektu, ale hlavne pri jeho plánovaní. Na identifikáciu rizík sa používajú rôzne metódy, ako napríklad:

- Zoznam rizík, klasifikačné schémy.
- Dekompozícia.
- Analýza rozhodnutí.
- Analýza predpokladov.
- Interview, dotazníky.

Teda ako výstup tejto fázy (kedykoľvek počas života projektu) sú určené a identifikované určité riziká, ktoré sme vedeli určiť a identifikovať. Ostatné riziká, ktoré môžu nastať, však môžu spôsobiť obrovské škody. Ak by takéto nejaké neidentifikované riziko nastalo, považuje sa to za chybu manažmentu a riadenia rizík.

Druhou fázou po identifikácií rizík je analýza rizík. Podľa mňa je analýza rizika nutná sčasti aj pri jeho identifikácií, hlavne kvôli odhadom pravdepodobnosti nastatia a prípadných spôsobených škôd. Každopádne, v manažmente rizík sa za analýzu rizika považuje jeho vyhodnotenie a určenie, ktoré riziká vyžadujú nejakú akciu. Ďalej sa analyzujú možné reakcie na udalosti spôsobujúce škody. Metód a techník na analýzu rizík je viacero, napríklad:

- Určenie vplyvu rizika, resp. očakávanej škody.
- Rozhodovací strom.
- Simulácia siete činností projektu v rozvrhu.
- Expertný odhad, napr. použitie metódy DELPHI.

Výstupom analýzy rizík je usporiadanie rizík podľa priority a z toho vyplývajúci zoznam udalostí, ktorými sa treba zaoberať a ktorými netreba (v druhom prípade sa škody akceptujú).

Poslednou fázou (ak nepočítame riadenie a sledovanie) je plánovanie manažmentu rizík. Plánovanie manažmentu rizík je vlastne definovanie krokov pri výskyte udalosti spôsobujúcej škodu, tzv. plánu reakcií. Pri jeho vytváraní sa berie do úvahy aj požiadavka na vyhnutie sa riziku, že následkom implementácie určitého spôsobu eliminácie rizika vzniknú riziká nové. Plán reakcií sa vytvára pomocou týchto metód a techník:

- Zoznam rizík, typy rizík a zodpovedajúce postupy.
- Vypracovanie alternatívnych stratégií.
- Skúmanie vzájomného vplyvu viacerých rizík.

Ako bolo v úvode tejto kapitoly napísané, manažment rizík sa považuje za samostatnú časť softvérového inžinierstva. Túto časť treba implementovať do

celkového manažmentu projektu. To si vyžaduje okrem iného aj zainteresovať všetkých zamestnancov, nie len vedenie projektu.

Prístup k riziku podľa extrémneho programovania

Extrémne programovanie [1], na rozdiel od softvérového inžinierstva, neobsahuje samostatnú časť venovanú rizikám projektu. Samozrejme, že pokladá riziko za problém, ktorý treba riešiť. K riešeniu tohto problému však pristupuje inak. Nedáva si za cieľ hľadať metódy a spôsoby ako odhadovať riziko vopred, ako sa na neho pripraviť a ako na neho reagovať, ak nastane. Podľa extrémneho programovania to ani nie je dosť dobre možné. Podľa mňa to do určitej miery možné je, ale len na projektoch, ktoré sú vopred zanalyzovateľné, popísateľné a jasné. Jasné z pohľadu ako a čo robiť. Ako dobrý príklad takýchto projektov sú projekty na tvorbu informačných systémov. Tam je to vždy skoro to isté: databáza, užívateľské rozhranie a interakcia medzi nimi. Nie však všetky projekty sú takéto. Napríklad vývoj nových technológií a systémov. V takýchto projektoch, ktoré často začínajú takpovediac „na zelenej lúke“, je skoro nemožné vopred identifikovať (a tobôž nie analyzovať a riadiť) možné riziká. Toto sú také dva extrémne príklady typov projektov, najčastejšie sa asi projekt pohybuje tak niekde v strede, že „vieme približne čo ideme robiť“.

Extrémne programovanie teda, ako bolo povedané, neobsahuje samostatnú časť venovanú rizikám v softvérovom projekte. Extrémne programovanie je štýl vývoja programového vybavenia, ktorý s rizikami počíta na všetkých úrovniach vývojového procesu. Čo to vlastne znamená? To je najlepšie pochopiteľné zo základných princípov extrémneho programovania.

Vývoj softvéru v extrémnom programovaní má nasledovné špecifické vlastnosti:

- Dvojica programátorov pracuje spoločne.
- Vývoj je riadený pomocou testov. Najskôr sa testuje a potom sa pridávajú nové funkcie.
- Dvojice programátorov nezaisťujú iba funkčnosť testov. Rozvíjajú tiež návrh celého systému. Zmeny nie sú obmedzené na žiadnu konkrétnu oblasť.
- Integrácia (vrátane testovania) nasleduje ihneď po vývoji.

Pre lepšie pochopenie extrémneho programovania je dobré uviesť, ako sa extrémne programovanie vysporadúva so základnými rizikami:

- *Meškanie harmonogramu*. Extrémne programovanie vyžaduje krátke cykly pre uvoľňovanie nových verzií (najviac niekoľko mesiacov), tým je rozsah akéhokoľvek meškania obmedzený. V rámci novej verzie sa používajú týždenné až mesačné iterácie pre funkcie požadované zákazníkom. Tým je zaistená veľmi podrobná spätná väzba, ktorá sa týka pokroku na projekte.
- *Zrušený projekt*. Extrémne programovanie žiada od zákazníka, aby určil najmenšiu možnú verziu, ktorá má ešte zmyslel. Tým sa minimalizuje to, čo sa môže pokaziť pred uvedením do prevádzky.

- *Krachujúci systém*. Extrémne programovanie udržuje kompletnú sadu testov, ktoré sa spúšťajú pred aj po každej zmene systému (kľudne niekoľkokrát denne). Tým udržuje systém stále v prvotriednej kondícií. Chybám nie je umožnené, aby sa hromadili.
- *Nepochopenie zadania*. Extrémne programovanie požaduje od zákazníka, aby sa stal integrálnou súčasťou tímu. Špecifikácia projektu sa neustále behom vývoja projektu upresňuje.
- *Zmeny zadania*. Skrátením cyklu uvoľňovania nových verzií sa zmenšuje pravdepodobnosť zásadných zmien. V rámci uvoľnenia verzie možno zákazníkovi vyjsť v ústrety, ak chce nahradiť ešte nedokončenú funkciu novou. Vývojový tím si ani nevšimne, či pracuje na novoobjavenej funkcii alebo na funkciách definovaných pred niekoľkými rokmi.
- *Príliš veľa funkcií s malou alebo žiadnou využiteľnosťou*. Extrémne programovanie trvá na tom, aby sa riešili iba úlohy s najvyššou prioritou.
- *Fluktuácia pracovníkov*. Extrémne programovanie sa snaží predchádzať zmenám v zložení vývojového tímu. Každý člen musí prijať zodpovednosť za odhadovanie termínov dokončovania vlastnej práce. Ich odhady práce rešpektuje. Tým sa znižuje pravdepodobnosť, že bude programátor frustrovaný zadaním evidentne nemožnej úlohy. Ďalej svojimi princípmi podporuje komunikáciu medzi členmi tímu a tým znižuje osamotenosť, ktorá býva často jadrom pracovnej nespokojnosti. Extrémne programovanie začleňuje explicitný model obmeny pracovníkov. Noví členovia tímu sú povzbudzovaní, aby postupne prijímali väčšiu zodpovednosť, a je im poskytovaná asistencia od ďalších nováčikov a aj od skúsenejších programátorov.

Porovnanie a použiteľnosť v malom tíme

Nie je jednoduché rozhodnúť, ktorý spôsob vedenia projektu je jednoznačne lepší pre použitie v malom, v prípade môjho tímu v Tímovom projekte, šesťčlennom tíme. Je zrejmé, že rizík je v takomto tíme, zloženom výhradne zo študentov, veľmi veľa. Ja sám ako študent poznám mentalitu študentov, resp. viem ju odhadnúť podľa seba. Za asi najväčšie naše riziko považujem riziko fluktuácie ľudí a nedodržania termínov. Už v prvom, teda zimnom semestri máme jedného člena tímu v zahraničí a keď sa v letnom semestri vráti, pravdepodobne odíde druhý. Problém s dodržiavaním termínov je a asi aj bude večným problémom študentov (minimálne mojím určite).

Podľa mňa je pre náš tím nereálne, aby sme venovali čas kompletnej procedúre manažmentu rizík popísanej v softvérovom inžinierstve. Stratili by sme tým priveľa času a výsledok by sa aj tak asi nedostavil, pretože by sme nedokázali identifikovať ani zďaleka všetky možné riziká a udalosti, ktoré môžu nastať (a určite aj nastanú) pri práci na našom projekte. Pre správny manažment rizík je rozhodne potrebné mať

schopnosti a skúsenosti s manažmentom projektu. Tieto skúsenosti na potrebnej úrovni podľa mňa nikto z tímu zatiaľ nemá. Funkčný manažment rizík je náročná, a nie vždy úspešná činnosť aj pre skúsených softvérových inžinierov a manažérov.

Pre náš tím je lepšie použiť aspoň sčasti prístupy extrémneho programovania, pretože priamo počíta s rizikami. Asi najzaujímavejšie sa mi javí použitie párového programovania a časté revízie kódu všetkými členmi tímu. Tým by sa mohlo dosiahnuť to, že každá časť vyvíjaného programového vybavenia bude známa väčšiemu počtu členov tímu, ako len jedinému. Je totiž kľudne možné, že by niekto pracoval na určitej časti systému samostatne, poriadne ju nezdokumentoval a potom z projektu odišiel (čo je úplne bežné aj v softvérových firmách), napr. na zahraničné štúdium. Vtedy by vlastne nebol nikto v tíme, kto by s tou časťou vedel reálne pohnúť, tým myslím zmeniť funkčnú časť alebo opraviť prípadnú chybu a nevniesť niekoľko nových chýb. Treba si uvedomiť, že dobrá dokumentácia je veľmi subjektívna vec. Existujú určité modely a metódy, ako dokumentáciu k softvéru písať (UML a podobne), ale ani tie nie sú všemocné a nie na všetky druhy softvérových projektov sa hodia. Potom je ťažké objektívne posúdiť, či je daná dokumentácia málo alebo až veľmi podrobná, alebo či je prehľadná a zrozumiteľná.

Myslím si však, že nie všetky prístupy extrémneho programovania sú pre náš tím vhodné. Napríklad je dobré mať aspoň základný časový plán projektu, odrážajúci čas, kedy sú kontrolné termíny na odovzdanie jednotlivých častí. My totiž nemôžeme postupovať pravidlom, že sa robia len najnutnejšie funkcie programu a ako dokumentácia slúži výhradne zdrojový kód. A taktiež nemôžeme žiadať od pedagogického vedúceho, aby určil minimálnu množinu spravených úloh (prípadne implementovaných funkcií), za ktoré ešte dostaneme ohodnotenie „E“. Predovšetkým určenie kontrolných termínov, a tým teda určité riadenie rizika, že sa nebude stíhať, je veľmi užitočné.

Záver

V tomto príspevku som sa snažil predložiť dva rôzne prístupy k rizikám v softvérových projektoch. Manažment rizík má snahu všetky možné riziká vopred odhadnúť a čo najlepšie sa na ne pripraviť. Extrémne programovanie sa zmierilo s tým, že nemožno všetky riziká vopred odhadnúť a snaží sa postupovať tak, že s nimi od začiatku počíta. Slovo extrémne je však naozaj na mieste. Ak sa rozhodneme túto metodiku v našom projekte použiť, treba ju čo najlepšie objasniť všetkým zainteresovaným stranám, teda programátorom, manažérom a aj zákazníkom, pretože tí sa podľa extrémneho programovania stanú členmi tímu. Ďalším extrémnym prístupom je používanie výhradne zdrojového kódu ako dokumentácie. Toto si vyžaduje minimálne objektovo orientovaný prístup k tvorbe systému, ale aj veľkú dávku slušnosti pri písaní zdrojového kódu.

Neviem dať odpoveď, ktorá metodika je lepšia. Všeobecne sa považuje extrémne programovanie vhodné pre malé až stredne veľké tímy. Pre veľké tímy, tvorené desiatkami programátorov a manažérov, je ale doporučované použiť prístupy popisované v softvérovom inžinierstve, a teda aj manažment rizík. Ani to však nie je

dokonalé a stále sa vyvíja a postupuje dopredu. Je možné, že v budúcnosti dôjde aspoň k čiastočnému prekrytiu oboch metód.

Použitá literatúra

1. Beck, K.: Extrémní programování. Grada Publishing, Praha, 2002.
2. Bieliková, M.: Softvérové inžinierstvo. Princípy a manažment. Vydavateľstvo STU, Bratislava 2000.
3. McGraw-Hill: Schaum's Outline of Theory and Problems of Software Engineering. The McGraw-Hill Companies,Inc, 2002.

Annotation

Risk planning and analyses in software project - extreme programming and risk management

One of major problems software development has to face is risk. Risk, that development fails and will not result in desired output. This failure may have tremendous impact on both economic and human potential even in case of large software houses. Role of analyses and risk planning in software project is to identify risks early enough and subsequently minimize their impact on final quality of software product by planning them. Essay provides brief overview of most common risks. It presents approaches, how do extreme programming and software engineering attempt to handle risks. While comparing stated approaches it focuses on their relevance and usability for small teams similar to those formed within team project at FIIT STU.