AKO SPRÁVNE RISKOVAŤ A ČO NAJMENEJ STRATIŤ PRI RIZIKÁCH V MALOM TÍME?

K výšinám bezrizikovosti cesty niet.

Miroslav Čorba

Slovenská technická univerzita
Fakulta informatiky a informačných technológií
Ilkovičova 3, 842 16 Bratislava
miroslav.corba[zavináč]gmail[.]com

Abstrakt. *Riziko je všade okolo nás. V každej sfére každodenného života na nás číhajú nástrahy, pred ktorými nie je možné uniknúť. Inak to nie je ani pri vývoji softvéru. Ba čo viac, vývoj softvéru môže byť často sprevádzaný ešte väčším rizikom. V tejto eseji chcem poukázať práve na riziká, ktoré číhajú pri vývoji softvéru na malý tím, ktorý rieši dvojsemestrálny projekt. Vychádzam pri tom z rizík, ktoré patria do TOP 10 rizík podľa Boehm-a a snažím sa vyjadriť svoj názor na mieru jednotlivých rizík, ktorá projekt ohrozuje. Všetky riziká, identifikované v zozname TOP 10 rizík rozoberám z pohľadu malého tímu i používaných metodík vývoja. Taktiež poukazujem na rôzne stratégie, ktoré sa pokúšajú minimalizovať riziká, vznikajúce pri tvorbe softvéru. Snažím sa najmä identifikovať, aké stratégie použiť v konkrétnych situáciách. To závisí najmä od skúsenosti tímu a povahy projektu.*

Kľúčové slová: *riziko, TOP 10 rizík, analýza rizík, malý tím*

Riziko je tu stále

Softvér sa už dávno stal pre svet, biznis, či všedného človeka neoddeliteľnou súčasťou každodenného života. Málokto z jeho používateľov ale vie, čo všetko sa za jeho tvorbou skrýva. Nie je žiadnym tajomstvom, že obyčajný človek vidí častokrát v práci počítačového špecialistu a experta len človeka, ktorý sa celé dni „hrá“ s počítačom. Vývoj softvéru je ale samozrejme oveľa komplexnejší proces, za ktorým sa skrýva množstvo úskalí a rizík. Nie

jeden softvérový projekt preto končí neúspechom. Rôzne štatistiky [6] dokonca uvádzajú, že až 70% softvérových projektov každoročne končí neúspechom. Je evidentné, že takáto situácia si vyžaduje okamžité riešenie. Tento problém sa preto v určitej miere už dlho snaží riešiť manažment rizík, ktorý hľadá spôsoby a metódy, ktorými by čo najviac minimalizoval riziká, vznikajúce pri vývoji softvérového produktu.

To ale nie je vôbec ľahké. Rizík je v softvérom procese viac než dosť. Môžu to byť riziká externé, či interné, niektoré sa dokonca ani nedajú identifikovať a objavia sa, keď ich najmenej čakáme a keď je na nich veľmi ťažké reagovať. Preto je potrebné čo najviac rizík zadefinovať, aby ich ďalšie znovuobjavenie nebolo sprevádzané tak veľkým negatívnym dopadom. Jedným z takýchto pokusov o definovanie rizík bolo TOP 10 rizík, ktoré definoval Barry Boehm.

Riziko a manažment rizík v malom tíme

Riziko

Riziko môžeme definovať viacerými spôsobmi. Riziko sa totiž nachádza všade okolo nás, nielen pri vývoji softvérových produktov. Vo všeobecnosti [1] ho môžeme definovať ako možnosť utrpieť stratu, poškodenie, nevýhodu alebo zničenie. Spojením tejto definície v kontexte softvéru, spolu so softvérovým inžinierstvom môžeme túto definíciu aplikovať aj na riziká, spojené so softvérovým procesom. Tieto riziká je potrebné identifikovať už v skorých fázach vývoja, čím je umožnené znížiť náklady, spojené s vývojom a chrániť softvér pred neúspechom [3].

Manažment rizík

Ako už zo spojenia „manažment rizík" vyplýva, úlohou tohto procesu je riadenie a manažovanie rizík, ktoré sa v softvérovom procese vyskytujú. Z množstva nájdených definícií som pre definíciu manažmentu rizík použil [1], [3]: *Úlohou manažmentu rizík je skorá identifikácia, analýza a reakcia na riziká s cieľom minimalizovať možnosť výskytu nežiaducich udalostí, skorého odhadu dôsledkov a v prípade výskytu takýchto udalostí skoré identifikovanie riešenia a minimalizácia ich dopadu.*

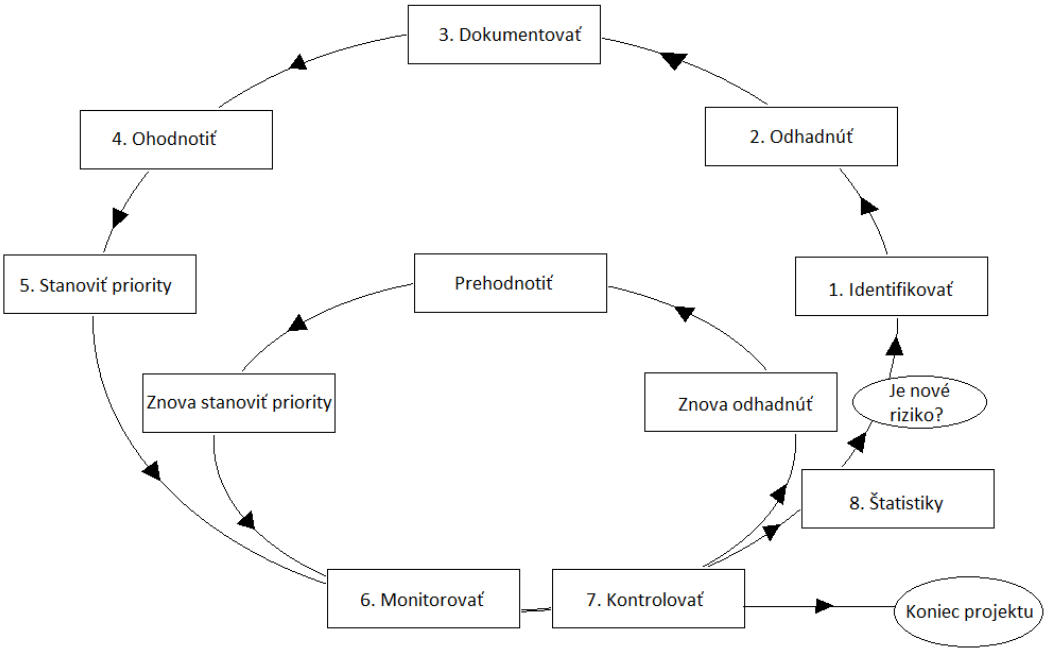
Ako riešiť problémy s rizikami v softvérovom projekte?

Aj ten najmenší softvérový projekt so sebou nesie svoje riziká. Nech už ide o akékoľvek riziká, je potrebné ich riešiť. Ako to vo svojej knihe [3] uviedol Boehm, riziká je potrebné odhadnúť čo najskôr a taktiež začať s ich analýzou už v skorých fázach riešenia projektu.

Otázkou je, ako efektívne využívať manažment rizík pre čo najlepšiu analýzu rizík. V [4] definovali model pre riadenie procesu manažmentu rizík. Tento model využíva 8 - krokový proces. V tomto procese sa nachádza ďalší 5 – krokový proces, ktorý umožňuje rýchlejšie reagovať na vznikajúce riziko. Celý tento model je uvedený na obrázku 1. Na tomto obrázku vidieť, koľkými krokmi prechádza proces po identifikácii nového rizika:

- identifikovanie nového rizika
- odhad pravdepodobnosti rizika
- zdokumentovanie rizika

– ohodnotenie rizika podľa jeho pravdepodobnosti
– stanovenie priorít pre identifikované riziko



Obr. 1. Model pre riadenie procesu manažmentu rizík(podľa [4]).

Pri tejto definícii vidieť celý cyklus, ktorým prebieha proces vývoja softvéru. To znamená, že je možné sledovať aj fázy vývoja pred identifikovaním rizika a taktiež po jeho vyhodnotení. Dôležité je spomenúť, že identifikovaniu rizika predchádza silná kontrola, monitorovanie a analýza, pri ktorých sa monitorujú všetky aspekty vývoja softvéru. Taktiež je dôležité, že ihneď po identifikovaní a definovaní rizika nastáva opäť fáza striktného monitorovania a kontroly procesu. Takýmto spôsobom je identifikácia rizika oveľa jednoduchšia a úspešnejšia.

Podľa mňa je práve fáza monitorovania a analýzy softvéru tou najťažšou. Práve tam je totiž potrebné zosúladiť všetky poznatky o produkte do takej miery, pri ktorej môžeme jednoducho odhadnúť závažnosť rizika. Manažéri rizík by preto mali byť ľudia trpezliví, precízni a s logickým uvažovaním. Zvládnutie práve týchto predpokladov dáva sľubné vyhliadky pre vyvíjaný softvérový produkt.

Malý tím

V tejto eseji budem pod malým tímom definovať tím piatich až siedmich ľudí, ktorí sa viac-menej dobre poznajú. Niektorí spoločne už pracovali na menších zadaniach

a projektoch spolu. Projekt nie je veľmi rozsiahly a mal by byť ukončený v presne stanovenom termíne odovzdania. Keďže ide o dvojsemestrálny projekt, dĺžka trvania projektu je asi jeden rok. Dôležité sú týždenné stretnutia so zákazníkom, na ktorých je možnosť konzultovať všetko, čo sa za posledný týždeň udialo. Projekt môže byť vyvíjaný podľa viacerých metodík, najmä pomocou metodiky SCRUM[1] alebo iteratívneho a inkrementálneho vývoja. Môžu ale byť použité aj iné metodiky. Podľa použitej metodiky som rozdelil vývoj na dve skupiny:

- *vývoj s prvotnou analýzou* – vývoj, pri ktorom počiatočné fázy projektu analyzujú a navrhujú softvérový produkt. Implementácia nastáva až po tejto fáze.
- *vývoj s pravidelnými modulmi* – vývoj, pri ktorom je zákazníkovi pravidelne predvedený novú funkčný modul produktu.

Toto rozdelenie je mnou definované a je urobené len pre lepšiu názornosť v neskorších častiach eseje. Pomáha jednoduchšie predstaviť, čím sa dané metodiky líšia pri jednotlivých rizikách.

TOP 10 rizík podľa Boehm-a

Zoznam TOP 10 rizík, ktorý definoval Boehm vo svojej knihe [3], bol vytvorený už v roku 1991 a špecifikuje nasledovné riziká:

- nesprávne chápanie alebo podcenenie požiadaviek zákazníka a nesprávna funkcionalita
- použitie nových technológií
- nereálne rozvrhy a nedostatok času na plnenie úloh
- zmena požiadaviek
- výkon softvéru v reálnom čase
- nedostatok personálu
- vytvorenie nevyhovujúceho používateľského rozhrania
- pozlátenie systému
- nedostatky v externe zabezpečovaných úlohách
- nedostatky v externe zabezpečovaných moduloch

Tieto riziká majú za úlohu predstaviť manažérovi rizík najčastejšie sa objavujúce riziká, spojené s vývojom softvéru a majú nabádať k úspešnému zvládnutiu základných pravidiel, ktorými možno riziká minimalizovať. Napriek pomerne dlhému časovému odstupu od vytvorenia tohto zoznamu, môžeme po malej modifikácii niektorých rizík, použiť tento zoznam aj pri dnešnom vývoji softvéru.

Analýza jednotlivých rizík v malom tíme

Myslím si ale, že nie je správne „hádzať všetky tieto riziká do jedného vreca." Nie všetky spomínané riziká podľa môjho názoru majú rovnaký vplyv na prácu v malom tíme a na malý projekt. Dôvodom môže byť samotný malý tím, použitá metodika vývoja ale aj

[1] http://www.scrumalliance.org/

mnoho ďalších faktorov. Nie je predsa možné porovnávať riziká v malom a veľkom tíme. Taktiež je jasné, že riziká pri rôznych metodikách nie sú tie isté.

Nesprávne chápanie alebo podcenenie požiadaviek zákazníka a nesprávna funkcionalita

Takéto riziká je vhodné riešiť na pravidelných stretnutiach k projektu, ktoré sú každý týždeň. Práve na týchto stretnutiach je možné spýtať sa zákazníka na správne chápanie jednotlivých požiadaviek.

Myslím si, že menšiu nevýhodu pri tomto riziku majú malé tímy, ktoré používajú vývoj s prvotnou analýzou. Dôvodom je neskoršia implementácia, ktorej výsledok je často prekvapením pre zákazníka, keďže očakával inú funkcionalitu. Na reparát je ale väčšinou málo času a projekt môže skončiť neúspechom. Preto je potrebné pravidelne konzultovať so zákazníkom funkcionalitu celého produktu, aj keď ešte nie je implementovaná.

Naproti tomu, vývoj softvéru s pravidelnými modulmi, môže zákazníkovi pomôcť skoro identifikovať zlú funkcionalitu produktu. Pravidelne totiž vidí, ako sa produkt vyvíja a zlú funkcionalitu dokáže skoro identifikovať. Takúto zmenu funkcionality je jednoduchšie opraviť a zvyčajne nemá katastrofálny vplyv na projekt, keďže identifikovanie rizika nastáva v podstate hneď po implementácii nevhodnej funkcionality a predvedení tejto funkcionality.

Použitie nových technológií

Dnešný softvérový svet už zďaleka nie je len o zopár programovacích jazykoch a technológiách. Množstvo nových a často lepších technológií vedie tímy k ich využívaniu. Nepreskúmané a nové technológie ale so sebou nesú veľké riziko. Je spojené najmä so skutočnosťou, že niektoré vlastnosti softvéru môžu byť pri danej technológii len ťažko implementované alebo členovia tímu danú technológiu neovládajú na požadovanej úrovni, ktorou by splnili ciele projektu. Pri tomto riziku je potrebná dôkladná analýza problémovej oblasti. Pravdepodobnosť, že tím má členov, ktorí ovládajú danú technológiu navyše s nižším počtom členov klesá.

Aj pri tomto riziku môže mať nevýhodu vývoj s prvotnou analýzou. Implementácia totiž začína až po dôkladnej analýze a vývojári môžu neskoro zistiť, že s danou technológiou nebudú schopní projekt dokončiť. Preto je veľmi dôležité novú technológiu dobre analyzovať najmä s pomocou vývojárov a rozhodnúť, či je možné túto technológiu použiť.

Vývoj softvéru s pravidelnými modulmi môže vďaka častým stretnutiam a implementácii od začiatku projektu skoro identifikovať, že používaná technológia nie je pre členov tímu vhodná a práca s ňou je pre vývojárov príliš ťažká.

Nereálne rozvrhy a nedostatok času na plnenie úloh

Nereálne rozvrhy sú často zdrojom neúspechu projektu. Podľa môjho názoru je extrémne ťažké odhadnúť dĺžku procesu vývoja. Svedčia o tom aj rôzne výskumy [5], ktoré uvádzajú, že len jedna šestina softvérových projektov bola ukončená podľa rozvrhov. Toto riziko je pri malých tímoch určite menšie ako pri tímoch väčších. Nie je ho ale možné úplne vylúčiť z vývoja softvéru. Aj pri malých tímoch sa práve termíny stávajú nočnou

morou pre vývojárov a často sú porušované. Je preto potrebné analyzovať a monitorovať dodržiavanie jednotlivých termínov a pri porušení skoro začať s ich nápravou.

Toto riziko neposkytuje pri jednotlivých metodikách nejaké významné rozdiely. Pri vývoji s prvotnou analýzou je ľahšie odhadnúť dĺžku procesu keďže je jasné, čo všetko je potrebné implementovať a ľahšie odhadnúť, koľko času to bude vyžadovať. Pri vývoji softvéru s pravidelnými modulmi je zas ľahšie počas vývoja zistiť, v akom stave je produkt a akú odchýlku má od rozvrhu. To je umožnené vďaka pravidelne dodávaným modulom, ktorými je možné sledovať postup vývoja produktu.

Zmena požiadaviek

Zmenou požiadaviek môže zákazník znepríjemniť život celému vývojovému tímu. Zmena požiadaviek je veľmi nebezpečným rizikom, ktoré môže so sebou priniesť aj riziko nereálnych rozvrhov. Každá zmena totiž stojí nejaký čas, ktorý môže v konečnom zúčtovaní chýbať. Proti tomuto riziku je ťažká obrana. Najmä malé tímy môžu mať s týmto rizikom problémy. Počet členov je totiž nízky a pri veľkej zmene požiadaviek nemusí byť postačujúci na to, aby zmena požiadaviek bola úspešne implementovaná.

Metodiky vývoja podľa mňa neposkytujú dostatočné nástroje pre rýchle odstránenie tohto rizika. Vývoj softvéru s pravidelnými modulmi síce môže nové požiadavky zabudovať do ďalšieho modulu, zmena už existujúcej funkcionality ale znamená čas a financie na viac ako pri ostatných metodikách.

Výkon softvéru v reálnom čase

Toto riziko je potrebné analyzovať už v skorých fázach vývoja a určiť, čo všetko bude pre beh systému potrebné. Pomocou simulácií je potom vhodné produkt testovať a zisťovať, či sú požiadavky splnené. Toto riziko nemá rozdielny význam pre rôzne veľké tímy. Preto pre malý tím bude riešenie tohto rizika podobné ako pri tímoch väčších.

Jednotlivé metodiky vývoja pre toto riziko neponúkajú konkrétne riešenia, ktorými by bolo možné riziko odstrániť. Myslím si, že vhodnou možnosťou by mohla byť práve simulácia a prototypovanie. Najmä pri inkrementálnom vývoji je možné testovať každý vytvorený modul a aspoň takto sa pokúšať minimalizovať riziko.

Nedostatok personálu

Zloženie tímu je často zložitým procesom, ktorý nie vždy vedie k optimálnemu zoskupeniu ľudí a je možné ho nájsť aj pri malých tímoch. Riziká, spojené s personálom môžu byť spojené s nedostatočnou skúsenosťou jednotlivých členov tímu, slabou motiváciou členov tímu, či nevhodnými podmienkami na prácu. Je preto potrebné čím skôr identifikovať prácu jednotlivých členov tímu a v prípade nespokojnosti s výsledkami začať tento problém riešiť.

Toto riziko by podľa môjho názoru mali všetky prístupy k vývoju riešiť podobným spôsobom. Dôležité je pravidelné monitorovanie a sledovanie práce všetkých členov tímu. Zabezpečiť sa to dá pomocou rôznych podporných prostriedkov, ktoré umožňujú zaznamenávať aktivitu jednotlivých členov na projekte. Dôležité ja taktiež vhodne motivovať jednotlivých členov tímu. Je zrejmé, že v prípade nedostatočnej motivácie klesá aj zanietenosť členov tímu do práce, čím sa znižuje aj jej úspešnosť. Motivovať sa dá nielen

finančne. Poskytnutie rôznych zamestnaneckých výhod môže motivovať niekedy viac ako by sa očakávalo. Nemenej dôležitou časťou je vytvorenie vhodných podmienok pre prácu. Na to môže slúžiť napríklad poskytnutie rôznych školení a seminárov, ktoré poskytnú členom tímu predpoklady pre úspešné zvládnutie úloh. Medzi takéto podmienky môžeme zaradiť aj kultúrne pracovné prostredie, ktoré poskytne príjemnú atmosféru v tíme.

Vytvorenie nevyhovujúceho používateľského rozhrania

Spokojnosť zákazníka práve s týmto aspektom produktu je veľmi dôležitá. Ide o jednu z viditeľných častí produktu a preto je dôležité často konzultovať používateľské rozhranie so zákazníkom. Na to môžu slúžiť pravidelné stretnutia so zákazníkom.

Pri vývoji softvéru s prvotnou analýzou je funkčný výsledok predstavený až na konci projektu. Preto je potrebné vytvoriť návrhy používateľských rozhraní už v skorších fázach a postupne ich so zákazníkom konzultovať. Takto môžeme aspoň čiastočne zabrániť nevyhovujúcemu používateľskému rozhraniu. Ani to nám ale nemusí zabezpečiť plnú spokojnosť zákazníka. Dôležitý je totiž funkčný výsledok, na ktorom sa môže aj používateľské rozhranie, ktoré bolo predtým pre zákazníka vhodné, stať nevyhovujúcim. Môže to byť podľa mňa spôsobené tým, že návrh rozhraní nemôže zákazník priamo otestovať a overiť či je to naozaj to, čo si predstavoval. Pri testovaní výsledku potom zistí, že rozhranie vôbec neposkytuje to, čo vyžadoval. Spôsob, ktorý by som navrhol pre riešenie tohto rizika, je dodávanie funkčných prototypov, na ktorých môže zákazník overiť splnenie požiadaviek na rozhranie.

Vývoj s pravidelnými modulmi potláča toto rizika ešte viac. Pravidelne dodané moduly totiž môže zákazník ihneď testovať a vysloviť svoju spokojnosť či nespokojnosť s vytvoreným rozhraním. V prípade nespokojnosti je možné zakomponovať požadované zmeny ihneď do ďalšej časti implementácie a riziko je tak dostatočne minimalizované.

Pozlátenie systému

Podobne ako pri predchádzajúcom bode, aj pre toto riziko sú pravidelné stretnutia základným kameňom pre úspešné minimalizovanie rizík. Nemusí to ale ani zďaleka stačiť. Zákazník môže mať viacero zástupcov s protichodnými požiadavkami. V tom prípade môžu vývojári pozlátiť softvér a iný zástupca takéto pozlátenie identifikuje ako nesprávnu funkcionalitu.

Vývoj softvéru s prvotnou analýzou špecifikuje všetky požiadavky na výsledný produkt už počas prvotných fáz analýzy. Pravidelné konzultácie so zákazníkom i jeho zástupcami môžu pomôcť predísť takýmto protichodným požiadavkám a produkt je potom implementovaný pomocou ucelených požiadaviek, s ktorými väčšinou súhlasia všetci zainteresovaní. Aj tu ale môže vývojár do produktu pridať svoje pozlátenie a takáto doplnená funkcionalita nemusí byť pre produkt vhodná. Takto pridaná funkcionalita je často identifikovaná až po odovzdaní projektu, kedy už nie je možnosť nápravy. Preto je pre minimalizovanie tohto rizika potrebné konzultovať každú pridávanú funkcionalitu so zákazníkom ešte pred jej implementáciou.

Vývoj s pravidelnými modulmi špecifikuje požiadavky počas celého vývoja a tieto požiadavky sú ihneď implementované do funkcionality produktu. Každá funkcionalita je pravidelne predstavená a konzultovaná so zákazníkom a tak je minimalizovanie rizika

zakomponované priamo do takejto metodiky vývoja. Zabezpečí sa to tak, že na pravidelnom stretnutí zákazník predstaví funkcionalitu, ktorú sám nedefinoval a ktorá sa mu nepáči.

Nedostatky v externe zabezpečovaných úlohách

Externe zabezpečované úlohy (angl. *outsourcing*[2]) sa stávajú fenoménom doby. Aj keď sú tieto úlohy často spájané s väčšími tímami, nie je vylúčené ich použitie pri malých tímoch. Typickým príkladom, aj keď nie z oblasti IT, je spoločnosť Redbull, ktorá začínala s malým počtom ľudí. Všetky úlohy spoločnosti boli zabezpečované pomocou externe zabezpečovaných úloh. Každý pravdepodobne vie, ako je na tom Redbull dnes.

Takéto úlohy so sebou ale nesú aj veľké riziká. Externe zabezpečení ľudia, či produkty, nemusia zapadnúť do koncepcie tímu a je možné, že ich úžitok nebude spĺňať požadované kritéria. Je preto dôležité rozhodnúť sa, ako veľmi takéto externé úlohy malý tím potrebuje a či by nebolo vhodnejšie pre riešenie problému použiť vlastných ľudí, ktorí majú s daným projektom skúsenosti a vytvoriť vlastné produkty, ktoré bude potrebné pre splnenie problému.

V prípade, že externú úlohu je nutné použiť, je dôležité, aby tím úlohu vhodne analyzoval. Analyzovať je potrebné najmä riziká, ktoré sú s touto úlohou spojené. V závislosti od týchto rizík je potom potrebné vhodne zareagovať na vzniknuté podmienky a vybrať vhodné technológie a postupy pre ďalšiu tvorbu softvéru.

Nedostatky v externe vytvorených moduloch

Riziká, ktoré takto môžu vzniknúť, sú napríklad spojené s kompatibilitou používaného modulu s produktom, spoluprácou modulu, či vhodnosťou modulu pre daný systém. Vďaka dôkladnej analýze sa aj v prípade použitia externe vytvorených modulov minimalizujú riziká, ktoré spôsobujú vznik problémov. Pri prípadnom použití externých modulov je preto potrebné analyzovať moduly už v skorých fázach projektu. Výsledok celej analýzy modulu je potom potrebné zahrnúť do analýzy celého produktu a v závislosti od toho vytvoriť vhodné riešenie.

Aké stratégie použiť v manažmente rizík?

V tejto časti by som rád predstavil stratégie, ktoré pomáhajú pri vývoji softvéru. Ich rozdelenie úzko súvisí s rizikom, ktoré môže byť s daným vývojom spojené. O týchto stratégiách pojednáva práca [2], ktorá definuje nasledujúce typy stratégií:

- „opatrná“ stratégia
- „typická“ stratégia
- „flexibilná“ stratégia

Opatrná stratégia je určená pre nové, neskúsené tímy, ktoré používajú nové technológie, s ktorými nemajú skúsenosti. Môže byť použitá aj pre tímy, ktoré začínajú na neznámych projektoch, pri ktorých číha veľká miera rizika. Riziko je sledované na viacerých úrovniach

[2] http://www.wisegeek.com/what-is-outsourcing.htm

– individuálnej, tímovej či na úrovni celej organizácie. Vďaka tejto stratégii je možné identifikovať riziká už v skorých fázach vývoja softvéru.

Typická stratégia je určená pre tímy s určitou mierou skúsenosti či už s používanými technológiami alebo s projektmi daného typu. Pomocou tejto stratégie má byť riziko identifikované počas celého projektu. Pri tejto stratégii odpadá sledovanie na individuálnej úrovni.

Flexibilná stratégia je určená pre tímy, ktoré sú dokonale oboznámené s používanými technológiami a daným typom projektu. Väčšinou je sledovanie ohraničené len na úrovni celej organizácie. Riziká sú pri tejto stratégii definované na začiatku projektu a občas aj počas vývoja projektu.

Použitie daných stratégií úzko súvisí s tým, aký typ projektu bude vyvíjaný, aké technológie budú použité a ako skúsený tím bude softvér vyvíjať. Napriek tomu si myslím, že nie je možné rozhodnúť sa len na základe takýchto charakteristických čŕt jednotlivých stratégií. Napriek dokonalým znalostiam technológií, či dobre známemu typu projektu, je niekedy lepšie celý proces výberu stratégie prehodnotiť aj z iných hľadísk. Myslím si, že vhodným kompromisom by bolo začať projekt pomocou opatrnej stratégie a postupne prechádzať ďalšími stratégiami. Na jednej strane by to mohlo spôsobiť zbytočné plytvanie časom častými a silnými kontrolami a analýzou projektu. Na strane druhej by ale práve začiatok projektu opatrnou stratégiou mohol značne minimalizovať mnoho rizík. Je už na každom tíme, ako veľké riziko je ochotný podstúpiť.

Ako teda správne riskovať?

Je zrejmé, že vývoj softvéru je vo veľkej miere sprevádzaný rizikami. Nezáleží na veľkosti tímu, použitej metodike vývoja či ďalších faktoroch. Riziká tu stále boli, sú a budú. Vďaka správnemu manažmentu rizík je ale možné aspoň čiastočne riziká minimalizovať. Čo je ale k tomu potrebné?

Ešte pred samotným identifikovaním rizika je potrebné silno kontrolovať a monitorovať všetky aspekty vývoja softvéru. Vďaka tomu je možné identifikovať riziko čo možno najskôr, kedy jeho dopady nie sú na projekt tak výrazné. Každé jedno vznikajúce riziko je samozrejme potrebné vhodne analyzovať. Dôležité je odhadnúť dopad daného rizika na celý projekt a navrhnúť vhodné riešenie rizika. V rámci toho je potrebné analyzovať riziko z hľadiska používanej metodiky vývoja i z hľadiska veľkosti tímu. Jednotlivé riziká totiž majú rôzny dopad na projekt, vyvíjaný rôznymi metodikami a tiež rôzny dopad na rôzne veľké tímy.

Riziká je teda potrebné analyzovať a hľadať už v skorých fázach vývoja softvéru. Bez manažmentu rizík je totiž veľmi ťažké úspešne vyriešiť akýkoľvek projekt. Dôvodom je skutočnosť, že *každý projekt so sebou nesie svoje riziká a nie je možné sa im vyhnúť bez dodatočnej práce a úsilia riešiť problémy, spojené s rizikami*. Mieru týchto rizík je potom možné použiť ako vhodný parameter pre ideálnu voľbu stratégie riadenia rizík pri vývoji softvéru. Práve vhodne zvolená stratégia riadenia rizík môže výrazne zmierniť dopady rizík na softvérový projekt.

Vo väčšine prípadov je potrebné pre všetky tieto úlohy manažéra rizík. Je preto samozrejmé, že aj v malom tíme má manažér rizík svoje miesto.

Použitá literatúra

1. Bieliková, M.: Softvérové inžinierstvo. Princípy a manažment. Bratislava: STU, 2000. 220 s.
2. Boban M., Pozgaj Z., Sertic H.: Strategies for Successful Software Development Risk Management, Management Vol. 8, 2003, 2, pp. 77--91
3. Boehm, B. W.: Software Risk Management: principles and practices. 1991.
4. Keshlaf, A.A., Hashim, K.: A model and prototype tool to manage software risks. Quality Software, 2000. Proceedings. First Asia-Pacific Conference on , vol., no., pp.297-305, 2000
5. May, L. J.: Major causes of software project failures. 1998, Dostupné na: http://www.stsc.hill.af.mil/crosstalk/1998/07/causes.asp, 2010
6. Wallace, L. Keil, M.: Software project risks and their effect on outcome. In Communications of the ACM, Vol. 47, No. 4, 2004

Annotation

How to take risk and how do not lose by taking risk?

Nowadays, risk is all around us. Every day we are in contact with dangers, from which it is impossible to escape. There is the same problem in software development. What's more, software development can often be accompanied by an even greater risk. In this essay I want to point out the risks that influence small team that solves the two-semester project. I follow the risks which belong to list of TOP 10 according to Boehm and I try to express my views to various dangers that threaten the project. I discuss these risks identified in the list of TOP 10 from the perspective of a small team. I also point out different strategies that attempt to minimize the risks arising in the software. First of all I try to identify what kind of strategies to use in specific situations. It depends especially on experiences of the team and nature of the project.