Analýza rizík založená na vyhodnocovaní metrík

MIROSLAV ŠTOLC

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

Abstrakt. Vyhodnotenie možných rizík pri vývoji softvérového produktu je neodmysliteľnou súčasťou jeho životného cyklu. Niektoré techniky oceňovania rizík sú založené na znalostiach expertov, tieto sú značne subjektívne. Táto práca sa zaoberá analýzou rizík na základe softvérových metrík, ktoré sú používane aj pri manažmente kvality softvéru. Vyhodnocovanie priebehu projektu na základe sledovania metrík nám umožní včasnú identifikáciu možného rizika a to najmä v oblasti omeškania resp. predraženia projektu. Rizikám môžeme predchádzať, alebo sa na ne pripraviť už pri architektúre systému, použitím dynamických metrík. Na analýzu zdrojového kódu sú určené statické metriky ako napr. počet riadkov kódu. Týmito metrikami sledujeme a vyhodnocujeme priebeh vývoja softvéru v porovnaní s plánom.

Úvod

Analýza hodnotenia rizík je jedna z podstatných častí manažmentu softvérového projektu. Manažment rizík pomáha predchádzať neželaným situáciám najmä katastrofám a veľkým stratám hlavne na strane rozpočtu projektu. Bez akýchkoľvek pochybností môžem povedať, že každý čo i len menší projekt má svoje riziká. Táto práca sa zaoberá identifikáciou a analýzou rizík v procese architektúry systému a v procese implementácie systému za pomoci softvérových metrík.

Identifikácia rizík je určenie, aké riziká môžu ohroziť projekt a dokumentovanie ich charakteristík. Identifikáciu treba vykonávať počas celého života projektu, v prvom rade však pri plánovaní projektu [1]. Čím skôr je riziko lokalizované, tým pre projekt lepšie. Preto sa práca zaoberá metódami hľadania rizík v jednej z prvých fáz projektu, menovite, v architektúre systému. Metriky použité pri UML (Unified Modeling Language) diagramoch sú nezávislé od programovacieho jazyka a aj od samotného programátora a jeho návykov.

Členenie eseje je nasledovné: kapitola0obsahuje popis pojmu metrika a stručnú históriu metrík. Nasledujúca kapitola0obsahuje z hľadiska rizika vedľajšie použitie

Manažment v softvérovom inžinierstve, október 2007, s. 1-8.

metrík a to vo fáze plánovania, súčasne táto kapitola obsahuje popis metrík pre fázu implementácie, kde na základe týchto metrík hodnotíme priebeh projektu a tým aj možnosť vzniku rizika. Ďalšia kapitola0podrobnejšie opisuje použitie dynamických metrík vo fáze architektúry systému. Posledná kapitola0je zhodnotenie prístupu riziko - metrika.

Metrika

Metriky možno definovať ako kritériá určujúce atribúty softvérového projektu; umožňujú hodnotenie vytvoreného výrobku a procesu, ktorý sa použil pri jeho tvorbe [1]. Metriky sa používajú na sledovanie istého objektu, najčastejšie programu. Prvá časť práce sa venuje týmto metrikám a ich výhodám a nevýhodám, použití najmä na sledovanie procesu vývoja. V druhej časti sa venujem metrikám, ktorých objektom sledovania je model, konkrétne model architektúry softvéru. Návrh systému prebieha podľa štandardu UML, čo je asi najpoužívanejší prístup pre modelovanie systému v dnešnej dobe.

Metriky používané nad UML modelmi môžeme rozdeliť podľa cieľa merania na statické a dynamické. Dynamické metriky opisujú vzťahy medzi objektmi, naproti tomu statické opisujú vzťahy medzi triedami. Asi je zrejmé, že z hľadiska testovania produktu a hľadania možných rizík sú statické metriky takmer nepoužiteľné.

História metrík

Softvérová kríza v deväťdesiatych rokoch spôsobila posun v oblasti zdokonaľovania metrík [6], existoval dopyt po lepšom rozvrhu projektu, odhadu ceny a kontroly kvality. Všetko toto bolo dosiahnuteľné zlepšením manažmentu procesu vývoja a to spôsobili práve metriky.

Definícia metriky podľa [6]: softvérová metrika je merateľná časť produktu alebo procesu, ktorým je tento produkt vyvíjaný. V našom prostredí je produktom softvér, ktorý je v podstate abstraktný objekt, vyvíjaný z prvého stavu - potreby, po konečný stav - funkčný systém, vrátane zdrojového kódu a rozličných foriem dokumentácie.

Softvérové metriky môžeme klasifikovať na dve podskupiny a to produktové a procesné metriky. Produktové opisujú samotný produkt, od analýzy cez architektúru až po zdrojový kód. Procesné popisujú priebeh vývoja produktu.

Plán a priebeh projektu, metriky a riziká

Plánovanie je súčasťou projektu, intenzívne je najmä v začiatočných etapách vývoja softvéru. Neskôr v priebehu vykonávania a riadenia projektu sa plány podľa potreby môžu meniť [1]. Na základe porovnania plánu so skutočnou situáciou resp. so skutočným vývojom môžeme identifikovať hroziace riziko najmä typu nedostatku zdrojov a času. V tomto momente sa ako vhodné ukazuje použitie metrík na riadenie, hlavne v časti implementácie produktu.

Pri vytváraní plánu sa jeho koncept konfrontuje so skúsenosťami z minulých projektov na základe nameraných metrík. Veľkú rolu tu hrá štatistika, Gaussova krivka funguje aj pri vývoji softvéru. Za plán v časovom rozsahu a alokácii zdrojov niektorej časti, komponenty, môžeme ako dobrý odhad považovať priemernú vyťaženosť porovnateľnej komponenty z minulých projektov. Takýmto spôsobom vieme základe analýzy a skúseností vytvoriť plán pre projekt. Sledovaním metrík zdrojového kódu vieme potom včasnejšie odhadnúť možnú hrozbu.

Metriky používané vo fáze implementácie

Klasické metriky [6] používané na sledovanie vývoja projektu sú porovnávané s plánom a následne vyhodnotené. Toto vyhodnotenie spočíva najmä v porovnaní s "úspešne" dokončenými projektmi, alebo stanovené expertom v danej oblasti so skúsenosťami z iných projektov.

Medzi najpoužívanejšie metriky patrí:

- počet riadkov kódu (Lines of code - LOC), najpriamejšia a asi aj najpoužívanejšia metrika, ktorá sa možno zdá byť na prvý pohľad nepoužiteľná. Štatisticky je to ale veľmi silná a efektívna metrika na meranie postupu v projekte.
- počet volaní funkcie
- veľkosť súboru, v objektovom programovaní veľkosť triedy (priemerná), najlepší je stred, nie veľa malých tried a ani nie málo veľkých (rozsiahlych) tried
- počet tabuliek v databáze
- počet klientskych požiadaviek
- a mnoho ďalších

Priebeh vývoja, sledovanie metrík manažérom

Pred začiatkom programovania máme analýzu, na základe analýzy sa plán projektu konkretizuje a upresňuje. Ak v minulom projekte resp. ako priemer minulých projektov trval vývoj jednej tabuľky, formulára istý čas pri istých zdrojoch je predpoklad, že teraz to bude podobne.

Pri samotnom vývoji, vidím konkrétne koľko trvala istá časť vývoja (napr. spomínaná tabuľka). Ak v polovici projektu zistím, že som mal mať vyvinutých o 5 jednotiek viac. Tak sa tu črtá problém do budúcna v podobe oneskorenia projektu - čo je riziko.

Metriky nám umožnia predvídať tieto riziká, samozrejme, sú tu aj iné. Iná práca by mohla analyzovať, ktoré riziko je horšie. Príklad pre iné ťažko porovnateľné riziko spojené s použitím cudzej netriviálnej technológie (napr. objektovo relačná perzistencia).

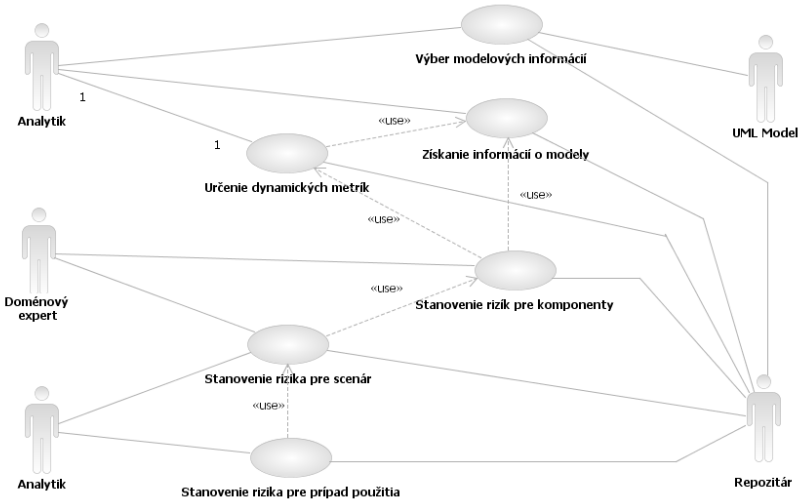
Podľa môjho názoru (a nie iba môjho), by sa metriky nemali použiť na jednotlivcov! To môže byť kontraproduktívne, aj keď by sa zdalo, že môžeme stimulovať efektivitu vývojového tímu. Ale ak si budú vývojári vedomí, že sú meraní, môže to spôsobiť nechcený tlak na jednotlivca. Ale toto už spadá do oblasti psychológie.

Analýza rizík vo fáze architektúry

Model systému pozostáva z diagramu prípadov použitia, ktorý obsahuje niekoľko samotných prípadov použitia. Každý z nich následne spresníme pomocou sekvenčného diagramu (alebo viacerých), ktorý určuje akýsi scenár pre daný prípad použitia. Sekvenčný diagram opisuje skupinu komponentov (najmä objektov) a ich komunikáciu posielaním správ medzi nimi [2]. Rozsah tejto komunikácie je merateľný práve dynamickými metrikami, ktoré budú podrobnejšie spomenuté neskôr.

Stanovenie rizík na základe architektúry systému

Podľa známeho, empiricky získaného poznatku, že osemdesiat percent chýb je spôsobených dvadsiatimi percentami programu je vhodné tieto chybné časti (komponenty) identifikovať čo najskôr. Preto je tu snaha nájsť metódu hľadania týchto chýb už prvých fázach životného cyklu programu. Táto metóda [2][3] je založená na analýze resp. na ohodnotení už existujúcej architektúry systému. Názorný popis realizácie tejto metódy pomocou diagramu prípadov použitia je na obrázku nižšie (Obr. 1). Fáza architektúry nie je celkom na začiatku, ale už obsahuje dostatočné množstvo informácií na efektívne hľadanie možných rizík. Následné odstránenie týchto rizík resp. presunutie viac zdrojov do implementácie a testovania rizikových komponent nás odbremení od neskorších nutných modifikácií a možných ťažkostí. Toto je určite úspora aj času aj prostriedkov.



Obr. 1 Diagram prípadov použitia pre stanovenie rizík v architektúre

Samotná metóda má dve fázy, ktoré určia celkové riziko skúmanej komponenty. Definuje riziko ako kombináciu dvoch častí: pravdepodobnosť poruchy a dôsledky poruchy. Je rozdiel ak neškodná porucha nastane toľko krát koľko závažná porucha systému. Nájsť tieto zložky rizika nie je triviálne. Preto sú používané kvantitatívne faktory pre komponentu ako zložitosť komponenty a spojitosť s inými komponentmi. Tieto faktory sú merateľné práve dynamickými metrikami, ktoré budú popísane v ďalšej z častí.

Tento prístup pre odhad rizík vo fáze architektúry je vyžadovaný resp. potrebný najmä pri produktoch ktoré predstavujú aplikácie v skutočnom čase alebo v produktoch veľmi citlivých na možnosť rizika, napr. zdravotnícke softvéry.

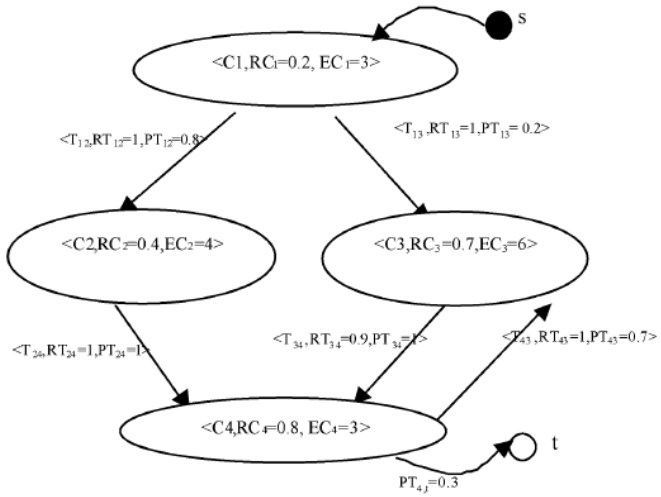
Dynamická metrika

Komplexné dynamické správanie aplikácie motivuje k nahradeniu klasických statických metrík dynamickými metrikami. Aktívne komponenty systému sú častejšie zdrojom pádu systému, kvôli častému vykonávaniu a častému zmenu stavu [3].

Metrika je získaná na základe istej postupnosti udalostí - scenára. Tento scenár je výsledkom analýzy a je získaný z UML, a to konkrétne zo diagramu prípadov použitia a sekvenčného diagramu. Dynamická metrika modeluje interakciu medzi komponentmi. Jednou z najpoužívanejších dynamických metrík je meranie previazanosti komponentov. Dva komponenty sú prepojené, ak jeden ovplyvňuje históriu druhého. História objektu je postupnosť jeho stavov, stav opisujú aktuálne hodnoty atribútov. Nasleduje príklad dynamickej metriky - prepojenie komponentov.

Graf závislosti komponentov

Tento prístup je podrobne opísaný v [3][8], graf závislosti komponentov (Component Dependecy Graph - GCD). Je to pravdepodobnostný model pre analýzu spoľahlivosti (čo je v podstate opak rizika) na úrovni architektúry. Modelujú systém ako kompozíciu podsystémov, komponent a rozhraní. GCD je orientovaný graf obsahujúci už spomenuté modelované časti spolu so spoľahlivosťou komponent, rozhraní a pravdepodobností prechodov. GCD sú konštruované na základe tzv. scenárov, čo je množina interakcií medzi komponentmi, ktorá môže byť získaná napríklad z UML sekvenčných diagramov. Na nasledujúcom obrázku (Obr. 2) je jednoduchý príklad GCD, pod obrázkom sa nachádza popis prvkov grafu.



Obr. 2 Príklad graf závislosti komponentov [8]

Vrchol grafu: <popis komponentu, spoľahlivosť komponenty, priemerný čas vykonávania v danom scenári>. Hrana grafu: <T_{ij} - hrana z komponentu i do komponentu j, spoľahlivosť prechodu, pravdepodobnosť prechodu>

Metodológia pre odhad rizika podľa [8]:

1. Modelovanie architektúry použitím opisného jazyka pre architektúru (Architecture Description Language), keďže v súčasnosti je takmer každá architektúra popisovaná pomocou UML, existuje nástroj na simuláciu týchto modelov. Na túto simuláciu sa používa technika Objektovo Orientovaného Modelovania v skutočnom čase (ROOM - Real-time Object Oriented Moddeling) [10].
2. Analýza zložitosti [8]
3. Analýza dôsledkov poruchy, metódou FMEA (Failure Modes and Effects Analysis) [9]
4. Získanie heuristických faktorov rizika pre komponenty a rozhrania
5. Vytvorenie grafov závislosti komponentov GCD
6. Získanie rizika z GCD pomocou prehľadávaním tohto grafu [8]

Presnejší opis tejto dynamickej metriky aj jej konkrétne použitie resp. vyčíslenie by mohlo byť obsahom inej práce. Ja som chcel podať iba akýsi prvotný pohľad na dynamickú metriku a jej možné použitie pri vývoji.

Záver

Metriky sú podľa mňa neoddeliteľnou súčasťou stredných a väčších softvérových projektov. Sú spojené s prvotným plánovaním po obdŕžaní špecifikácie a analýzy. Následne ako súčasť životného cyklu programu prichádza architektúra, po ktorej vieme uskutočniť prvú iteráciu hľadania možných rizík v projekte a to použitím dynamických prípadne aj statických metrík. Počas implementácie produktu sledujeme vývoj metrík nad zdrojovým kódom a porovnávame ich s plánom projektu. Týmito metrika je sledovaná aj kvalita vyvíjaného softvérového produktu a kvalita procesu vývoja softvéru. Metriky sú súčasťou minimálne troch častí vývoja softvérového produktu: plánovania, kvality softvéru a v neposlednej rade analýzy rizík na základe architektúry ale aj zdrojového kódu. Preto používanie metrík a tým aj ich zdokonaľovanie je všeobecne odporúčané pre všetky vývojové tímy.

Použitá literatúra

1. Bieliková M.: Manažment v softvérovom inžinierstve, Bratislava 1999
2. Katerina Goseva-Popstojanova, Senior Member, IEEE, Ahmed Hassan, Student Member, IEEE, Ajith Guedem, Walid Abdelmoez, Student Member, IEEE, Diaa Eldin M. Nassar, Student Member, IEEE, Hany Ammar, Member, IEEE, and Ali Mili, Member, IEEE: Architectural-Level Risk Analysis Using UML, *IEEE Transactions on software engineering*, Vol. 29, No. 10, October 2003, 946 - 960
3. Sherif M. Yacoub, Hany H. Ammar, and Tom Robinson: A Methodology for Architectural-Level Risk Assessment using Dynamic Metrics, *Proceedings of the 11th International Symposium on Software Reliability Engineering*
4. Youssef Hassoun, Roger Johnson, Steve Counsell: A Dynamic Runtime Coupling Metric for Meta-Level Architectures, *Proceedings of the Eighth European Conference on Software Maintenance and Reengineering*
5. Vittorio Cortellessa, Katerina Goseva-Popstojanova, Senior Member, IEEE, Kalaivani Appukkutty, Ajith R. Guedem, Ahmed Hassan, Student Member, IEEE, Rania Elnaggar, Walid Abdelmoez, Student Member, IEEE, and Hany H. Ammar, Member, IEEE Computer Society: Model-Based Performance Risk Analysis, *IEEE Transactions on software engineering*, Vol. 31, No. 1, January 2005
6. Everald E. Mills: Software Metrics, *SEI Curriculum Module* SEI-CM-12-1.1
7. Sherif M. Yacoub, Hany H. Ammar, and Tom Robinson: Dynamic Metrics for Object Oriented Designs, West Virginia University
8. Sherif M. Yacoub, Member, IEEE Computer Society, and Hany H. Ammar, Member, IEEE Computer Society: A Methodology for Architecture-Level Reliability Risk Analysis, *IEEE Transactions on software engineering*, Vol. 28, No. 6, June 2002
9. Yiannis Papadopoulos, David Parker, Christian Grante: A Method and Tool Support for Model-based Semi-automated Failure Modes and Effects Analysis of

Engineering Designs, *9th Australian Workshop on Safety Related Programmable Systems (SCS'04)*

10. B. Selic, G. Gullekson, and P. Ward, Real-Time Object Oriented Modeling. John Wiley and Sons, 1994.

Annotation

Risk assessment based on metrics evaluation

Risk assessment is essentially part of software product life cycle. Several methods of risk assessment are based on expert's knowledge so that is subjective view. This paper is a brief description of risk analysis based on software metrics which are used for quality assurance. Evaluation of project progress by observing metrics can enable us well-timed identification of feasible risk. We can prevent from risk or make ready for risk already at architecture of software system by using of dynamic metrics. To analysis of source code we could use common metrics i.e. lines of code. These metrics also helps managers to pursue history of project and to compare history with project plan.