SYSTÉMY NA SLEDOVANIE CHÝB
CHYBA ALEBO FUNKCIA?

Ak ladenie pokladáme za umenie odstraňovania chýb
tak programovanie musí byť umením ich vkladania.
[Autor neznámy]

Ondrej Topoľský

Slovenská technická univerzita
Fakulta informatiky a informačných technológií
Ilkovičova 3, 842 16 Bratislava
xtopolsky[zavináč]is.stuba[.]sk

Abstrakt.

Jadrom eseje sú systémy, ktoré slúžia ako komunikácia medzi zadávateľmi chýb a príslušnými vývojármi daného produktu – systémy na sledovanie chýb. V eseji sa venujem výberu správneho spôsobu zadávania chýb do systému na sledovanie chýb. Predstavme si situáciu, že niekto zadal do nášho systému novú chybu. Ak túto chybu zadal príliš rozvláčne, zodpovedný človek strávi veľa času čítaním a pochopením, o čo vlastne ide. Ak však tento človek popíše danú chybu príliš stručne, môže sa stať, že v popise nebude dostatok informácií k odhaleniu a náprave chyby. Táto esej sa teda zaoberá otázkami „Ako presne definovať chybu?“ a „Ako čo najzrozumiteľnejšie a najpresnejšie zapísať chybu?“. Vo veľkých projektoch je navyše potrebné vytvoriť štandard, alebo vzor zadávania chýb. Takto každý vie, čo čakať v danej správe o chybe. Týmto sa minimalizuje čas riešenia chýb.

Kľúčové slová: *bug, chyba, bug-track, vývoj, manažment, podporné prostriedky, bug report*

Úvod

Systémy na sledovanie chýb slúžia na zlepšenie komunikácie medzi vývojármi a ľuďmi, čo daný produkt používajú, alebo sú inak zainteresovaní jeho správnym fungovaním (môžu to byť aj ďalší vývojári, manažéri kvality a podobne). Títo ľudia sa nazývajú reportéri. Systémy na sledovanie chýb navyše slúžia na sledovanie kvality softvéru.

Tieto systémy umožňujú lepšiu komunikáciu medzi ľuďmi zainteresovanými na odstránení chyby pomocou zadávania chybového hlásenia (bug queries, bug reports).

Tieto chybové hlásenia väčšinou nemajú vopred určený rámec, ktorý by presne definoval, ako majú vyzerať. Inými slovami každý píše chybové hlásenie svojím spôsobom. Tento fakt spôsobuje, že časy strávené vývojármi na danom probléme alebo chybe sa zvyšujú. Toto je veľmi nepríjemný dôsledok a je veľmi žiaduce, aby sa daný čas riešenia a odstraňovania chyby minimalizoval.

Kladieme si teda otázky *„Čo je to chyba?“* a „Ako presne definovať chybu?“Podľa magazínu PCMag.com je to: *„Problém, ktorý spôsobuje, že program produkuje zlý výstup alebo padá“* [2].

Z uvedeného citátu vieme že chyba je problém v programe. Avšak tento problém potrebujeme nejako jednoznačne opísať tak, že ho príslušný vývojár dokáže odstrániť. Keďže jednoznačne opísať chybu nie je jednoduchá záležitosť, použijem výskumu z roku 2008 [1]. Tento výskum sa zaoberá kladením otázok reportérom a vývojárom ohľadom toho, čo presne si predstavujú, že by mal opis chyby obsahovať .

Ak vieme presne opísať danú chybu, ktorú chceme ohlásiť, je vysoká pravdepodobnosť, že ju zodpovedajúci človek(vývojár) bude vedieť nájsť a opraviť.

Zhromaždil som údaje niekoľkých testov a článkov zaoberajúcich sa danou problematikou. Niektoré výstupy môžete nájsť v tejto eseji, buď vo forme obrázkov, alebo priamo výstupov opísaných v zoznamoch alebo texte.

Na základe zozbieraných údajov a informácií sa pokúsim sformulovať stručné zhrnutie, tak aby bolo možné dané informácie čo najrýchlejšie znovu-použiť v praxi.

Čo tvorí dobrý report?

V tejto časti sa budem venovať rozboru pokusu uskutočnenom v roku 2008, o ktorom podáva správu článok *What makes a Good Bug Report* [1] .

Myšlienkou tohto testu bolo overiť, ako veľmi sa odlišujú názory vývojárov a reportérov o tom, čo by mal obsahovať dobrý chybové hlásenie a čo najviac pomôže vo vyriešení chyby. To, čo je skutočne dôležité samozrejme najlepšie vie vývojár, avšak v niektorých prípadoch reportéri nie sú schopní poskytnúť takýto obsah pre vývojárov. Daný test sa nesnaží zodpovedať, čo exaktne je potrebné na najlepšie odstránenie chyby, pretože každá chyba požaduje špecifické informácie k jej vyriešeniu (screenshot, stack-trace..)

Boli vytvorené ankety, ktoré boli rozoslané vývojárom a reportérom. Otázky pre vývojárov sú označené ako D(developers) a pre reportérov písmenom R. [1]

D1: Ktoré z nasledujúcich položiek(ľubovolne veľa) ste v minulosti použili na odstránenie chýb?

D2: Ktoré 3 položky[1] vám pomohli najviac?

R1: Ktoré z uvedených položiek ste naposledy uviedli v reporte?

R2: Ktoré tri položky bolo najťažšie poskytnúť

R3: Ktoré 3 položky sú podľa vás najrelevantnejšie pre vývojárov na to aby mohli opraviť chybu?

Na výber boli tieto položky:

- produkt (product)
- komponent (component)
- verzia (version)
- závažnosť (severity)
- hardvér
- operačný systém (operating system)
- zhrnutie (summary)
- informácie o builde (build information)[2]
- spozorované správanie (observed behavior)
- očakávané správanie (expected behavior)
- kroky na spustenie chyby (steps to reproduce)
- výpisy zásobníka (stack traces)[3]

1 položky chybového hlásenia – anglicky items, predstavujú druh informácií poskytnutých v chybovom hlásení, tieto položky tvoria chybové hlásenie

2 informácie o inštalácii daného programu

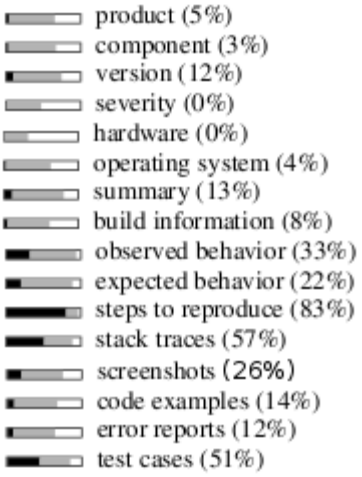
- zachytenie obrazovky
- príklady kódu (code examples)
- hlásenia vzniknutých chýb (error report)
- testovacie prípady (test cases)

Nasleduje rozbor výsledkov daných ankiet. Spôsob vytvárania daných štatistík z matematického hľadiska môžeme nájsť na strane 310 kapitola 2.3 v [1].

Otázky na vývojárov

Pre obe otázky D1 a D2 boli na výber rovnaké množiny položiek. Pre prvú otázku bolo možné vyznačiť ľubovoľný počet a pre druhú maximálne 3 položky.

Tieto otázky spolu súvisia. Dané otázky majú dva hlavné účely. Prvý je zistiť mieru dôležitosti danej položky v procese odhaľovania chyby. Druhý je relevantnosť konkrétneho vyplnenia ankety. Ak sa totižto zvolené odpovede z D2 nenachádzajú v D1, tak výber položiek zrejme daný vývojár neprehodnotil správne.



Obr. 1. Otázky D1/D2 na vývojárov, v zátvorkách dôležitosť položky [1]

Výstupom ankety boli vedomosti o tom, ktoré položky sú dôležité pre vývojára resp., ktoré položky z uvedeného listu vývojári reálne použili pri odstraňovaní chyby. Tento výstup je možné vidieť na obr. 1.

Najdôležitejšou informáciou sú očividne *kroky na spustenie alebo reprodukovanie chyby*. Je to popis toho, čo treba spraviť, aby nastalo neželané správanie programu. Ďalšími položkami sú výpisy zásobníka a testovacie prípady. Obe tieto položky pomáhajú zúžiť priestor, v ktorom budeme hľadať chybu.

Je potrebné spomenúť aj fakt, že určité položky úzko súvisia s komponentom produktu, v ktorom sa daná chyba vyskytla. Ak sa jedná o grafické rozhranie, bude

---

3 „trasovanie zásobníka“ – chybové hlásenie priebehu programu detailnými informáciami

potrebné získať obrazový výstup programu resp. zachytenie obrazovky a podobne. To znamená dôležitosť jednotlivých položiek pre rôzne druhy komponentov(súvisí aj s typom produktu) sa v určitom rozsahu mení.

Určité položky ako napríklad druh hardvéru, operačný systém boli použité na odstraňovanie chýb v menšom rozsahu, pretože sa táto anketa týka konkrétnych produktov ktoré sú nezávislé od týchto položiek, čo však nemusí platiť pre iné produkty.

Vo všeobecnosti môžeme povedať, že pre odstraňovanie chýb budeme potrebovať niektoré konkrétne položky reportu. Na základe zozbieraných informácií som zostavil tento zoznam (zoradené podľa dôležitosti zhora):

- kroky na vyvolanie, reprodukovanie chyby (83%)
- testovacie scenáre (51%)
- spozorované správanie (33%)
- očakávané správanie (22%)

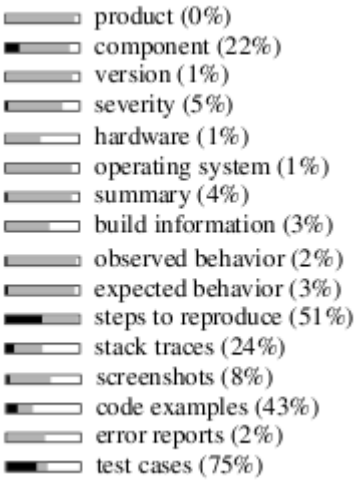
Vieme určiť ďalšie položky, ktoré bude s vysokou pravdepodobnosťou potrebné uviesť do chybového hlásenia:

- výpisy zásobníka (57%)
- zachytenia obrazovky (26%)
- príklady kódu (14%)
- zhrnutie (13%)
- verzia (12%)

Zoznam č. **1** Položky potrebné na efektívne riešenie chyby

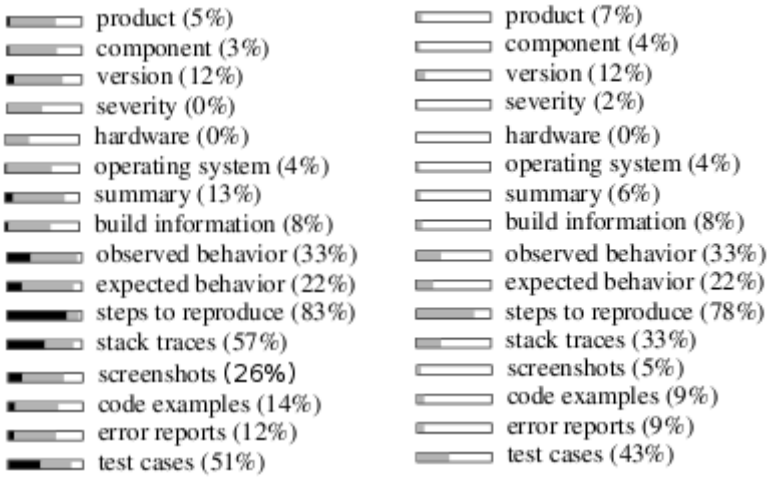
Vyššie uvedený zoznam však napríklad nehovorí nič o tom, aké zložité je pre reportérov poskytnúť dané položky do chybového hlásenia a či sú vôbec schopní takéto výstupy vyprodukovať. Sú však aj prípady keď reportéri nevedia , že je vhodné poskytnúť informácie takéhoto charakteru. Informácie o zložitosti poskytovania daných dát pre reportéra sú uvedené na obrázku č 2.

Otázky na reportérov



Obr. 2. Otázky R1/R2 na reportérov [1]

Na základe logickej konštrukcie otázok R1 a R2 vieme z uvedených odpovedí reportérov zistiť, ako zložité je pre nich poskytnúť danú položku v reporte. Na obrázku č. 2 môžete vidieť výsledky danej ankety.



Obr. 3. Čo si myslia vývojári(D1,D2) vs reportéri (R3) [1]

Na obrázku č. 3 môžete vidieť porovnanie toho, čo potrebujú vývojári (vľavo) a čo si reportéri myslia, že vývojári potrebujú (vpravo). Toto je však len informácia o tom, čo si myslia reportéri, že je správne, avšak sami tieto informácie (položky) neposkytujú. Dôvod vidíme na obrázku č. 2. Väčšinou tie najžiadanejšie informácie je práve ťažké poskytnúť. Sú to napríklad *kroky na reprodukciu chyby*. Táto jedna položka je najdôležitejšou informáciou k rýchlemu odstráneniu chyby. Avšak takáto položka sa veľmi ťažko produkuje.

Nekompletné a chýbajúce informácie

Ďalším problémom vznikajúcim pri písaní chybových hlásení sú chýbajúce, alebo dokonca nesprávne údaje poskytnuté reportérmi.

Na základe ďalšej ankety máme k dispozícii informácie týkajúce sa problémov súvisiacich s neuvedením údajov, alebo uvedením chybných údajov. V nasledujúcom zozname sú údaje hovoriace o tom, čo najviac spomalilo vývojárov v odstraňovaní chýb v súvislosti s chybovými hláseniami vývojárov[1] :

- chyby v krokoch na reprodukciu, vyvolanie chyby (79%)
- nekompletné informácie (74%)
- nesprávne spozorované správanie (48%)
- chyby v testovacích prípadoch (38%)
- neštruktúrovaný text(34%)
- nesprávne očakávané správanie (27%)
- príliš dlhý text (26%)

Zoznam č. **2** Čo spomaľuje vývojárov najviac

Skracovanie životného cyklu chyby

Dĺžku životného cyklu môže ovplyvniť veľa faktorov, ako napríklad nedostatočná sebakontrola pri písaní chybového hlásenia, nejednoznačnosť informácií atď. Na základe týchto nedostatkov sa diskusia k chybe predlžuje. Sú kladené dodatočné informácie k veciam, ktoré mali byť určite uvedené v chybovom hlásení.

V tejto časti rozoberiem ďalší prieskum [3], ktorý sa týka životného cyklu chyby, hlavne procesu diskusie k chybe. Toto zahŕňa frekventovanosť otázok objavujúcich sa v diskusiách k chybe, ich typom a klasifikáciu.

V danom prieskume bolo preskúmaných 600 chybových hlásení. Každé toto hlásenie obsahuje diskusiu k riešeniu danej chyby (dodatočné otázky, overenie poskytnutých informácií a podobne). Z týchto diskusií boli zozbierané otázky kladené reportérmi a vývojármi. Tieto otázky boli podrobne preskúmané a utriedené, tak aby vyhovovali našim zámerom. Vznikla tak množina 947 otázok. Z týchto otázok bolo vytvorených 8 hlavných kategórií otázok:

- chýbajúce informácie (chýbajú základné informácie)
- objasnenie (dodatočné informácie)
- triedenie (kam chybu zaradiť, komu, duplikáty)
- ladenie (ladenie aj za pomoci reportérov)
- korekcie (otázky týkajúce sa spôsobu nápravy)
- stav vyšetrovania (dlhá nečinnosť, či oprava vytvorí novú verziu)
- rozsah (či problém ostáva po upgrade [4])
- proces (administratíva, oprávnenia k zdrojom)

Zoznam č. **3** kategórie často kladených otázok

Uvedené kategórie ešte obsahujú podkategórie, tieto informácie sú dostupné v uvedenej práci [3].

Z uvedeného zoznamu vidíme podstatu kladených otázok. Ak dokážeme nejakým spôsobom automatizovať proces kladenia otázok, alebo ho nahradiť iným spôsobom zadávania týchto informácií, tak získame zmenšenie životného cyklu chyby a tým aj zmenšenie počtu nevyriešených chýb.

4 Prechod na novú verziu softvéru.

V danej práci [3] boli vytvorené 4 odporúčania na zlepšenie tohto procesu:

1. **Informácie ktoré sa vyvíjajú**
   Na začiatku životného cyklu sú kladené otázky týkajúce sa chýbajúcich informácií, alebo takých informácií, ktoré napomôžu k lokalizácii chyby a pochopeniu toho ako chyba nastala. Neskôr sú kladené otázky zamerané na nápravu chyby a stav riešenia chyby. Z toho dôvodu by pri počiatočnom vytvorení chyby mali byť k dispozícií rozhrania umožňujúce zadávanie obrázkov zachytenia obrazovky, výpisov zásobníka alebo krokov na vyvolanie chýb. Neskôr budú nahradené inými rozhraniami, ktoré by umožňovali korekciu a zistenie rozsahu chyby.
2. **prostriedok na kladenie častých otázok**
   Pri kladení otázok na oboch stranách (vývojárov i reportérov) boli otázky kladené v rôznych diskusiách v rôznom poradí. Toto vnáša neorganizovanosť do diskusie a spôsobuje predlžovanie riešenia chýb. Určité časové zdržanie nastáva aj pri otázkach typu, kto si niečo zoberie na starosť (napr. preverenie navrhovaného riešenia). Takéto administratívne rozdelenia by mohli byť vopred rozdelené a vložené do systému. Potom by systém automaticky priradil požadované úlohy, ktoré sú obsiahnuté v danej otázke. Diskutuje sa aj o tom že by reportér zvoli, kto danú chybu bude riešiť, avšak toto má ešte horší vplyv na efektivitu riešenia chýb. Reportér by preto nemal mať právo priradenia úlohy, pretože by mohol priradiť úlohu nesprávnej osobe. Preto toto priradenie vykoná systém na základe jeho predošlého nastavenia.
   Všeobecným riešením je teda vytvoriť prostriedok, ktorý by vedel adresovať dané otázky konkrétnym ľuďom a organizovať poradie ich pýtania sa.
3. **Explicitná manipulácia sledovania otázok**
   Reportéri potrebujú vidieť, že je požadovaná ich priama interakcia na riešení chyby. Niekedy nerozumejú požiadavkám vývojárov, ktoré sa im zdajú byť nejednoznačné, alebo napríklad ani nevyzerajú ako otázka. Preto by systém na sledovanie chýb mal podporovať prostriedok, ktorý by napríklad označil riešenie chyby stavom „čakanie na odpoveď", čím by zabezpečil pozornosť zainteresovaných osôb, od ktorých sa očakáva nejaká akcia.
4. **Odstraňovanie chýb v komunite**
   Pri riešení chýb sa stáva aj to, že reportéri po počiatočnej iniciácii chyby neodpovedajú na dodatočne kladené otázky. Cítia, že poukázaním na chybu spravili dosť. Tento problém treba odstrániť tak, že by reportéri cítili nutnosť podieľať sa aj naďalej na odstraňovaní chyby. Toto uvedomenie reportérov a osôb zainteresovaných na odstránení chyby zabezpečíme tak, že by sa tento systém chápal viac ako sociálna aktivita. V takomto systéme by sa napríklad ukladala história práce reportéra, alebo reputácia reportéra. Na základe toho by zainteresovaným osobám mohlo viac záležať na ich aktívnom prispievaní k riešeniu chyby.

Zoznam č. **4** Odporúčania na zlepšenie procesu zadávania chybových hlásení

Záver

Na záver zhrniem poznatky získané v tejto eseji. Zistili sme, že pre efektívne odstránenie chyby potrebujeme aby všetci zainteresovaní spolupracovali na riešení chyby. Reportéri by mali uviesť všetky položky (v závislosti od chyby), ktoré si myslia, že sú potrebné pre vývojárov na vyriešenie chyby (pretože podľa prieskumu tieto predstavy korešpondujú s tým čo vývojári potrebujú obr. č. 3). Tieto položky podľa prieskumu [1] sú hlavne tie, ktoré sú uvedené v zozname **1.**

Pri písaní chybového hlásenia je potrebné dať si pozor na správne uvedené údaje. Jediný spôsob ako zamedziť takýmto chybám, je informovať nových reportérov o tom, že by si mali chybové hlásenie po sebe prečítať a overiť, či je možné z danej správy rozpoznať a lokalizovať vzniknutú chybu v produkte. Existuje ďalší spôsob ako zabezpečiť aby poskytnuté informácie boli čo najkompletnejšie. Čiastočne je tak možné dosiahnuť tak, že v rôznych fázach riešenia chyby budú k dispozícii iné rozhrania na zadávanie vstupu (screenshoty, výpisy zásobníka a pod.). Na uvedenom zozname problémov v chybových hláseniach (zoznam č **2**) môžeme vidieť tieto nedostatky v reálnom nasadení.

V životnom cykle chyby sa stráca veľa času na čakanie na odpoveď, alebo na dodatočné informácie. V zozname číslo **3.** je vidieť aké typy otázok sú kladené v diskusiách k chybovým hláseniam. Na základe frekventovanosti týchto otázok a neusporiadanosti ich kladenia boli vytvorené určité návrhy a nápady ako zefektívniť proces výmeny informácií. Tieto navrhované riešenia takýchto problémov súvisiacich s príliš dlhou diskusiou je vidieť v zozname číslo **4**.

Keď zhrnieme dané informácie z jednotlivých testov vidíme, že časovo najnáročnejšia je synchronizácia reportérov a vývojárov. Tieto časové straty vieme odstrániť hlavne správnou konštrukciou systému, ale aj zmenou prístupu zadávateľov chyby a vývojárov, možno aj čiastočnou zmenou princípov súvisiacich s odstraňovaním chýb (motivovať zamestnancov určitým spôsobom).

Použitá literatúra

1. BETTENBURG , N. - JUST, S. - SCHRÖTER , A. et. al. 2008. *What Makes a Good Bug Report?* USA New York : ACM ISBN 978-1-59593-995-1
   Použitá časť dostupná z internetu
   <http://portal.acm.org/ft_gateway.cfm?id=1453146&type=pdf&coll=Portal&dl=GUIDE&CFID=109881664&CFTOKEN=35735781>
2. *Definition of: software bug* [online] [cit. 2010-10-19] Dostupné na internete:
   <http://www.pcmag.com/encyclopedia_term/0,2542,t=software+bug&i=51664,00.asp>
3. BREAU S., PREMRAJ R., SILLITO J. et. al. 2010. *Information needs in bug reports : improving cooperation between developers and users.*
   USA New York : ACM ISBN 978-1-60558-795-0
   Použitá časť dostupná z internetu
   <http://portal.acm.org/ft_gateway.cfm?id=1718973&type=pdf&coll=Portal&dl=GUIDE&CFID=109881664&CFTOKEN=35735781>
4. AVNON Y., BOGGAN S.L. 2010. *Fit and Finish using a bug tracking system: challenges and recommendations.* USA Atlanta Georgia : Microsoft Inc. ISBN 978-1-60558-930-5
   Použitá časť dostupná z internetu
   <http://portal.acm.org/ft_gateway.cfm?id=1754219&type=pdf&coll=Portal&dl=GUIDE&CFID=106908989&CFTOKEN=47395662>

Annotation

Bug tracking system, Feature or bug.

This essay is about bug-tracking systems and methods of writing bugs reports.
Imagine this situation: Someone is writing a bug report. If that person writes this report too long the appropriate developer would spend too much time of understanding of report. On the other side if the bug report would be too short the developer wouldn't find what he was looking for.

We ask ourself these questions:
What exactly are bugs?
How to write bug reports?

These are the main questions asked in this essay.
If we use some methods and principles of writing bug queries we will lower the amount of time spent on understanding on side of developer. This time is in most cases longer because the developer must ask additional questions.
Using these principles and methods we spent less time in dealing with one bug.