Kódellenőrzés

Bármilyen furcsán hangzik is, a legjobb kódellenőrzés az, amelyikre ténylegesen sor kerül.

A nyílt forráskód és a szabad program egyik legtöbbet hangoztatott előnye a kód kölcsönös ellenőrzése. „Több szem többet lát, és minden hibát megtalál” a megszokott refrén. Ez természetesen azt feltételezi, hogy tényleg rendelkezésedre áll számos figyelmes segítő, akik a kódod minden részletét képesek ellenőrizni, és tudják, hogy mit kell keresniük. A kód átvizsgálása olyan, mint a szex, bárki képes rá, de tanulással és gyakorlattal sokkal jobb eredmény érhető el.

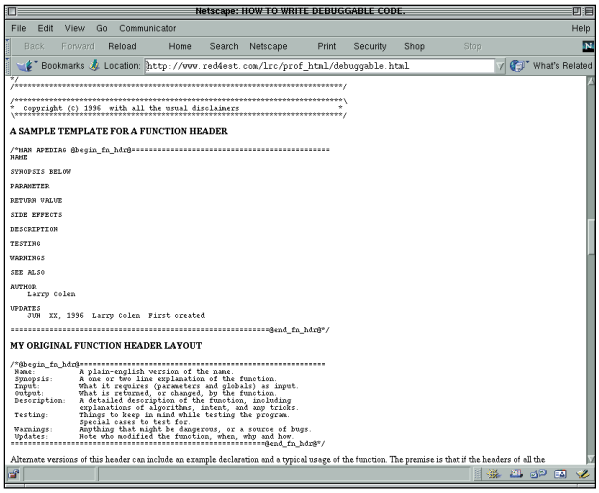
Azoknak a programozóknak, akik még nem ismernék a leckét, felsorolom a kód és a programterv ellenőrzésének előnyeit:

- A program életciklusa során korábban fellelt hibákat könnyebb javítani.
- Ha valaki más vizsgálja át a kódodat vagy a programtervedet, valószínű, hogy olyan hibákat is észre fog venni, amelyek felett te elsiklottál.
- Ha tudod, hogy a kódodat valaki más is meg fogja nézni, sokkal valószínűbb, hogy rendbe teszed és meggyőződsz róla, hogy a leírás naprakész.
- Sokat tanulhatsz mások kódjának olvasása során.
- Ha több ember is ismer egy kódot, biztosítva vagy a „Kamion szindróma” ellen. A Kamion szindróma az, amikor a programot ismerő egyetlen embert elüti egy kamion, vagy éppen nem érhető el, amikor szükség lenne rá.
- A kódellenőrzés segítségével lemérhető a különböző minőségbiztosítási folyamatok hatékonysága.
- Ha jól végzed a kód- és programterv-ellenőrzést, időt takaríthatsz meg és javíthatod a termék minőségét, a fejlesztés összes fázisában.

A programkód ellenőrzésének legfőbb hátránya, hogy időbe kerül és nemcsak az átvizsgálást végző személy idejébe, hanem másokéba is, akiket gyakran szintén határidők szorítanak. Bár tanulmányok sokasága kimutatta, hogy a projektre fordított munkaórák száma összességében kevesebb a jól végzett ellenőrzés esetén, mint anélkül, mindig ott a csábítás: hátha nincs is hiba a kódban és az ellenőrzés csak időpocsékolás. Ez azt jelenti, hogy megvárod, míg elkészül a kód, és csak utána fogsz hozzá a hibák kereséséhez, a gondok feltárásához.

A legjobb ellenőrzés az, amit ténylegesen el is végeznek. Egy gyors és felületes ellenőrzés, ami fényt derít a hibák egyharmadára többet ér, mint a jól átgondolt, tüzetes, mindenre kitérő, amelyre végül nem kerül sor.

Az ellenőrzéseknek két alapvető fajtája van: bemutató (walk through) és formális vizsgálat (formal inspection). A bemutató abból áll,



www.red4est.com/lrc/prof_html/debuggable.html

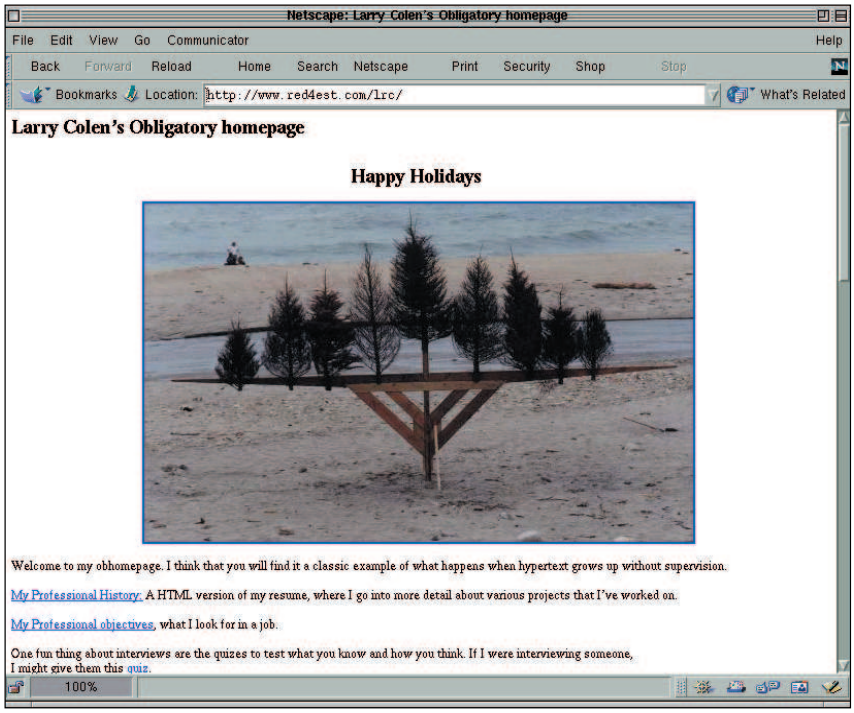
hogy a fejlesztő leül egy vagy több munkatársával és bemutatja nekik az ellenőrzésre váró kódot vagy tervet. Tapasztalataim szerint a bemutató alkalmával a hibák 80 százalékát maga a fejlesztő találja meg, magyarázat közben. *Eli Weber*, egy volt munkatársam, mondta egyszer: „Ha úgy tudnék a falakhoz beszélni, mint az emberekhez, nem lenne szükségem másra a programjaim ellenőrzéséhez.”

A formális vizsgálat esetén kiadják egy vagy több embernek az ellenőrzendő kódot vagy tervet. Gyakran az egyes emberek más dolgokra összpontosítanak a vizsgálat során, például az alkalmazott kódstílus vagy szabványok követésére, logikai hibákra vagy a leírás teljességére. A gyakorlatban ellenőrző listát szoktak készíteni arról, hogy kinek mire kell figyelnie. Ilyen szempontok például a kódolási stílus, a gyakori hibák, a biztonsági rések megtalálása stb.

Nincs abban semmi rossz, ha a kódolási stílust ellenőrző személy észrevesz egy VAGY helyett használt ÉS logikai operátort. Ilyenkor nyugodtan felírhatja azt, de nem kell törekedni arra, hogy mindenki minden szempontból ellenőrizze a kódot. Amikor a vizsgáló talál egy hibát, felírja annak helyét és súlyosságát. A fontossági sorrend változhat a „valami, amit esetleg megváltoztathatsz, ha nagyon nincs mit csinálnod, és a főnöknek rossz kedve van” tárgykörbe sorolandótól a „ha ezt a hibát nem javítod ki azonnal, olyan események láncreakcióját indítja el, ami az általunk ismert civilizáció megsemmisüléséhez vezet” csoportba tartozóig.

Ha mindenki elkészült a rábízott feladattal, az ellenőrök összeülnek megbeszélni a hibákat. Ezeket szinte bármilyen logikai sorrendben lehet tárgyalni: oldalanként, súlyosságuk alapján vagy aszerint, hogy melyik ellenőr találta meg őket. Az értekezletet általában a program szerzője vezeti, de ezt a szerepet más is vállalhatja. Az egybegyűltek minden feltárt hibánál megállapítják annak súlyosságát, vannak ugyanis esetek, amikor az átvizsgáló egymaga nem tudja eldönteni, hogy mennyire súlyos egy hiba, csak észleli, hogy valami nem jó. Ilyenkor megkérdezi a többiek véleményét. Ennél a pontnál nagyon fontos, hogy ne a hiba megoldására törekedjünk, hiszen a feladat most csak a feljegyzésük. Pontosan tudom, hogy milyen nehéz egy szobányi mérnököt lebeszélni arról, hogy ne rögtön a gondok megoldását keressék, de az ülés vezetőjének a megfelelő útra kell terelni a megbeszélés menetét, különben az hetekig eltart és többe kerül, mint amennyit megtakaríthatunk vele.

A kód átvizsgálásának így kell(ene) elméletileg működni céges környezetben vagy oktatási intézmények berkein belül, ahol a cégen belül az emberek együtt dolgoznak. De mi történik a nyílt forráskódos rendszerben fejlesztőkkel, akik gyakran nemhogy más városban, de esetleg más országban élnek és dolgoznak? Létezik néhány szűkebb körű projekt, amely levelezési listát használ kódellenőrzés céljára. Ilyen esetben a szerző bejelenti a változást, a lista tagjai pedig kedvük szerint ellenőrzik a kódot. Ha véleményük szerint valami nincs rendben („Te hitetlen, három oszloppal húztad be a kódot, amikor mindenki szerint csak kettővel kellett volna!”), írhatnak a szerzőnek, vagy a listára, és higgadtan és józanul megvitathatják a gondokat. Más projektek más modellt alkalmaznak: valakinek szüksége van egy programra, valamilyen célból. Keresi, de nem találja, így arra az elhatározásra jut,



www.red4est.com/lrc

hogy megírja. Amikor készen van, és annyira működik, amennyire a szerzőnek szüksége van rá, elérhetővé teszi a nagyközönség számára, például közzéteszi a freshmeaten vagy egy másik, hasonló jellegű fórumon. Valamikor később, valakinek kell egy program ugyanarra vagy nagyon hasonló célra. Megtalálja azt, amit te írtál, letölti, de nem tudja futtatni. Megkeresi azt a helyet, ahol a program elszáll, kijavítja a rosszalkodó kódrészletet és a szíve jósága által vezérelve, valamint a megszállott programozótársak iránti együttérzésből elküldi a javítást neked, azaz az eredeti szerzőnek.

Felhívnám a figyelmedet, hogy a fent vázolt két eset egyikében sincs biztosíték arra, hogy a kódodat ellenőrizni fogják, ha mégis, akkor nem biztos, hogy ez elég hamar történik meg ahhoz, hogy hasznodra váljék. Hogyan lehet a lelkiismeretes programozó biztos abban, hogy a kódját ellenőrzik, méghozzá időben, ahhoz, hogy haszna is legyen? A legtöbb megszállottnak megszállott barátai vannak. Az írók gyakran alkotnak irodalmi köröket, hetente egyszer találkoznak, és egymás írásait elemzik. Ugyanezt meg lehet tenni programok esetén is. A résztvevők megkaphatják a kódot vagy programtervet valamivel a találkozó előtt. A formális vizsgálat módszerét felhasználva megbeszélhetik a találkozón a felderített hibákat.

Egy másik megoldás: amikor a kód vagy programterv készen áll az átvizsgálásra, a szerző elküld egy levelet a megfelelő listára és meghívja programozótársait egy kellemes étterembe, ahol rendkívül finom fokhagymás pizza, és némi sör mellett kellemes hangulatban megejtik a programbemutatót.

Ha senki sincs helyben, aki segíteni tudna neked, ne keseredj el, a formális vizsgálat legnagyobb részét levélben is meg lehet oldani. A lényeg az, hogy kialakuljon egy csoport, melynek tagjai egymás kódját komolyan ellenőrzik, nem csak átfutnak a forráson. Hasonlóképpen a legtöbb hibát a programbemutatók során a szerző maga találja meg, a kód működésének ismertetése közben. Ha nincs a környéken senki, akit rá bírnál venni arra, hogy üljön le és figyeljen, amíg elmagyarázod a programod működését, esetleg a macskád sem bír *Richard Stallman* és *Dennis Ritchie* programozási tudásával, használhatod az Internetet, mint hallgatóságot: írj részletes leírást. Az, hogy leülsz a programhoz dokumentációt írni, rákényszerít arra, hogy kívülállóként nézzél rá és megtalálj olyan idegesítő hibákat, mint a rossz zárójelezés.

A kód átvizsgálása sokban növelheti a termék biztonságát. Először is, ha többen átnézik a kódot, kisebb az esélye annak, hogy hátsó kapukat találjanak a programban. Ilyen lehet például: „ha valaki f00Bidoo néven jelentkezik be, automatikusan rendszergazdai jogokat kap”. Ennek ellenére, a legtöbb támadás, főleg a nyílt forráskódú programoknál, ismert típushibák keresésével kezdődik, ezeket használják ki – mintegy rosszindulatú kódellenőrzést végezve. Ezen biztonsági rések lezárásához szükséges, hogy előbb találjuk meg a hibákat, mint az, aki kevésbé barátságos szándékkal keresi azokat.

A hibás kódrészletek felderítése – ahogyan ezt matematikus barátaink mondanák – szükséges, de nem elégséges feltétele a biztonságnak. Az OpenBSD biztonsága növelése értelmében kíméletlenül alkalmazni kell az ismertetett kódellenőrzési módszereket. Az OpenBSD Biztonsági csoportja számos kutatást végez ebben a témakörben. „Igazából erre összpontosítunk” – mondta *Theo de Raadt*. – „Megpróbálunk újfajta típushibákat felfedezni a régi kódban. A létező használatban lévő kóddal foglalkozunk, amelyben egyszerű hibák is lehetnek, amelyekre soha senki sem gondol. Ilyen például az fd_set túlcsordulása. Az OpenBSD-ből lényegében három éve kiirtottuk a puffertúlcsordulásokat.” Ha többet akarsz megtudni az OpenBSD-ről, a biztonsággal kapcsolatos elméleteikről és céljaikról, nézd meg a ➲ http://www.openbsd.org honlapot.

Összefoglalásul, a kódellenőrzés folyamán megtalált hibák többségét azért találják meg, mert valaki ránéz a kódra, ez lehet a kód ellenőre vagy a szerző egy bemutató során. Ugyanebben a szellemben állíthatjuk, hogy jobb, ha szerény vizsgálati módszert dolgozunk ki, mint ha elveszünk a részletekben, és a nagyra törő terveken alapuló átvizsgálásra időhiány miatt sohasem kerül sor. Az, akit foglalkoztat a biztonság, rengeteget nyerhet a kód gondos ellenőrzésével, de ez nem elegendő. Annak a szemével kell nézni a programot, aki ki akarja játszani a rendszer biztonságát. Nem csak azokra az esetekre kell figyelmet fordítani, amikor a program nem úgy működik, ahogyan azt elvárnánk. Azt is vizsgálni kell, hogy hogyan lehet a programot rávenni arra, hogy váratlan dolgokat műveljen.

Köszönetet szeretnék mondani *Theo de Raadtnak* a felbecsülhetetlen segítségéért, az OpenBSD-vel és a kódellenőrzés biztonsági szerepével kapcsolatban.

További érdekességek

Több hírt találsz róla, mint szeretnél a honlapján
➲ http://www.red4est.com/lrc.
Ugyanitt olvashatsz egy hosszadalmas és céltalan írást a programok minőségéről:
➲ http://www.red4est.com/lrc/prof_html/debuggable.html

Larry Colen
(lrc@red4est.com)
1994 márciusa óta foglalkozik Linuxszal és 1998 novembere óta hivatásául választotta.