SRAMÓ ANDRÁS

A szoftverminőség összetevői

Bevezetés

Egy szoftver kifejlesztése feltétlenül kudarcba fullad, ha a fejlesztő csoport nem képes a szoftvertermék megfelelő minőségét biztosítani. Nem mentség, hogy mindenki keményen dolgozott, ha a szoftver nem felel meg az elvárásoknak. Minél többet tudunk arról, hogy mi az, ami javítja a minőséget, és a fejlesztő csoport tagjai hogyan vehetnek részt a minőség megteremtésében, annál közelebb kerülünk a magas minőségi követelményeknek eleget tevő szoftvertermékek előállításához.

A minőség fogalmához számos összetevő kapcsolódik. A szoftvertechnológia jelenlegi fejlettségi szintjén nem tudjuk ezeket az összetevőket olymódon definiálni, hogy azzal a szoftver megrendelői, a fejlesztők és a leendő felhasználók egyaránt egyetértsenek, és a termékek könnyen, félreértések nélkül használhatók, ellenőrizhetők és mérhetők legyenek. A szakirodalom különböző felsorolásokat és csoportosításokat ismer.

Az összetevőket a szoftverfejlesztésben érdekelt mindhárom csoportnak (megrendelők, fejlesztők, felhasználók) ismernie kell. Vannak olyan összetevők, amelyek rejtett formában épülnek be a termékbe, ezért aztán akár el is lehetne feledkezni róluk. Ezek csak speciális vizsgálatokkal mutathatók ki, viszont hiányuk komolyan veszélyezteti a szoftver integritását és a biztonságos szoftverhasználatot.

A szoftver minősége

A „minőség" kifejezést gyakran félreértik a szoftverekkel kapcsolatban. Általában valamilyen „kiválósággal" hozzák összefüggésbe, de (mint azt látni fogjuk) *szoftvereknél a minőség fogalma a működés helyességére, tökéletességére vonatkozik.*

A szoftver minőségének meghatározásához két utat követhetünk.

- Az *egyik* út a felhasználók elégedettségének elemzése, amelyből megkaphatjuk azokat az összetevőket, amelyek a megelégedettséget leginkább befolyásolják.
  A legismertebb ilyen tesztrendszerek az IBM CUPRIMDSO tesztje, illetve a Hewlett-Packard FURPS tesztje. A felhasználók (fogyasztók) megelégedettségének mérése, elemzése elsősorban a marketing szempontjából rendkívül fontos. Ez a szemlélet a szoftver használójának (a fogyasztónak) a sajátos igényeiből kiindulva vizsgálja a szoftverek minőségét, mivel – Kotlert idézve – „az elégedett vásárló újra vásárol, másoknak is dicséri a gyártó céget, kevesebb figyelmet szentel a versenytárs márkáknak és a gyártó cég más termékeit is megveszi." (Kotler 1991) Marketingszemlélettel fogalmazhatjuk meg legpontosabban a megelégedettség definícióját is: ez a termékre, szolgáltatásra vonatkozó elvárások és tapasztalatok különbsége.
- A *másik* lehetőség a szoftverminőség meghatározásához, ha a minőség általános definíciójából indulunk ki. Erre talán legalkalmasabb az ISO 8402 szabvány, mely szerint a minőség: „Egy termék, folyamat vagy szolgáltatás azon sajátosságainak és jellemző tulajdonságainak összessége, amelyek hatással vannak arra a képességre, hogy teljesüljenek köz-

A cikk a szoftverminőség meghatározásának kérdésével foglalkozik. A minőség egy általános definíciójából kiindulva definiálja a minőségi összetevő fogalmát, majd a szoftvertechnológia jelenlegi szakirodalmára támaszkodva bemutat 23 minőségi összetevőt.
Az összetevők ismertetése során felvázolja a velük kapcsolatos legfontosabb ismereteket, problémákat, megoldási módszereket, ezzel tisztázva az egyes összetevőket érintő félreértéseket.

Sramó András egyetemi adjunktus a JPTE Közgazdaságtudományi Kar Gazdasági Informatika Tanszékén

vetlenül megállapított vagy közvetve megállapítható igények." A fenti definíció alapján az a feladatunk, hogy meghatározzuk és leírjuk azokat a sajátosságokat és jellemző tulajdonságokat, amelyek befolyásolják a szoftver minőségét, s ennek összetevőit.

Az összetevők felkutatásához a definíció második fele nyújt némi segítséget, mely szerint az összetevők mindig valamilyen igény kielégítésében szerepet játszó képességre vonatkoznak. Eszerint a minőségi összetevőket az igények környékén kell keresnünk. Egy szoftverre vonatkozó igények a megrendelőktől és/vagy a felhasználóktól származnak, és az igényspecifikációban öltenek formát.

Az igények lehetnek funkcionális igények (a szoftvernek mit kell tudnia) és nemfunkcionális igények (a szoftvernek hogyan kell működnie. A nem funkcionális igények szorosan kapcsolódnak a szoftver minőségéhez. Amikor a fejlesztőcsoport leszállítja az elkészült szoftvert, az igényspecifikációban szereplő elvárások teljesülését ellenőrizni kell ahhoz, hogy a megrendelő meg legyen elégedve a szoftver minőségével. Ezeket az elvárásokat felhasználói elvárásoknak nevezzük. A fejlesztő csoport nyilvánvaló célja, hogy ezeknek az elvárásoknak megfeleljen.

Ahhoz, hogy ezt a célt elérjék, a fejlesztő csoportnak biztosnak kell lennie abban: Ennek megfelelően egyes minőségi összetevők a felhasználói elvárásokból, míg mások a fejlesztői elvárásokból származnak.

A legfontosabb összetevők

A következőkben ismertetem a szoftverekkel kapcsolatos legfontosabb minőségi összetevőket. (Az összesnek a felsorolása szétfeszítené e cikk kereteit, hiszen Sage és Palmer 47 összetevőt sorol fel könyvében.)

Adaptálhatóság

Mennyire felel meg a szoftver annak a problémának a megoldására, amiért készítették? Ennek a minőségi összetevőnek az alapproblémája, hogy a fejlesztők mennyire értették meg a megrendelők által felvázolt igényt.

Befejezettség

Képes-e a szoftver mindenre, amit igényeltek? A szoftver azon termékek körébe tartozik, amely „félkészen" is forgalomba hozható, üzembeállítható. A szoftver félkész volta nemcsak az időhiányból származhat, hanem a választott implementálási módszertől is: a szoftvert modulonként állítják elő és helyezik üzembe. (Például egy könyvelési szoftvernek a napi tételfeladás részét üzembe állítják, de az éves zárást és a mérlegkészítést végző modulon még dolgoznak a fejlesztők.) Ez a fajta fejlesztés azonban rejt néhány veszélyt, amely károsan befolyásolhatja a szoftver minőségét: a modulok illeszthetőségével problémák lehetnek, egy már üzembe helyezett modulból hiányozhat például egy olyan funkció, amelynek eredményére egy későbbi modulnak szüksége van.

Biztonság

Mennyire védett a szoftver a környezet fenyegetéseivel szemben? Nyilvánvaló, hogy ha egyszer pénzt, időt és fáradságot fordítottunk egy szoftver létrehozására, a létrehozott értéket védeni kell. Ezzel az összetevővel kapcsolatban azt kell tisztázni, hogy mi a szoftver környezete, és milyen fenyegetettséggel kell szembenézniük a szoftver felhasználóinak. A szoftverkörnyezetet feloszthatjuk fizikai és logikai környezetre. A fizikai környezet a számítógép és annak környezete lesz, míg a logikai környezetbe az operációs rendszer, a szoftverrel kommunikáló egyéb szoftverek, a felhasználók munkamódszerei tartoznak.

A fenyegetettség szempontjából megkülönböztethetünk

- adatrögzítési hibát (a szoftver helytelen adatot fogad el),
- adatvesztést (a szoftver által elfogadott adatok nem kerülnek adathordozóra),
- károkozást az adatokban (szándékosan vagy véletlenül valaki tönkreteszi a rögzített adatokat),
- csalást (jogosulatlan információszerzés vagy adatmódosítás valaki érdekében),
- balesetet (emberi gondatlanságból, időjárásból vagy egyéb környezeti tényezőből származó fenyegetettség, mint például tűz, villámcsapás, tartós áramkimaradás, földrengés stb.)

Bővíthetőség

Milyen könnyen lehet a szoftver szolgáltatásait bővíteni? Minden szoftver egy problémakör megoldására készül. Idővel azonban a problémakör változik, újabb igények vetődnek fel, amit természetes módon a szoftver által nyújtott megoldás kibővítésével szeretnének a felhasználók kielégíteni. Kérdés, hogy a szoftver felépítése mennyire teszi ezt lehetővé. Ennek a minőségi összetevőnek a legnagyobb problémája, hogy teljesen rejtve van a megrendelők és felhasználók előtt, így a szoftver átvételekor szinte lehetetlen ellenőrizni.

Egyszerűség

Mennyire könnyű a szoftver használata és működésének ellenőrzése? Rendkívül kiterjedt hatású összetevő, kérdéseivel a szoftverergonómia foglalkozik.

Ellenállóképesség

Hogyan tudja a szoftver túlélni a nem várt, helytelen inputokat? Ez az összetevő tulajdonképpen a „biztonság" összetevőnél említett adatrögzítési hibák kezelésére vonatkozik, de külön is meg szokták említeni.

Érthetőség

Mennyire könnyű megérteni, hogy a szoftver mit csinál? Az adaptálhatósághoz és az egyszerűséghez kapcsolódó összetevő. A szoftver működése akkor érthető, ha jól elkülönülnek az egyes munkafázisok, a szoftver jelzi, hogy éppen hol tart (feldolgozásban, menürendszerben) vagy éppen mit csinál.

Használhatóság

Mennyire könnyű a szoftver használata anélkül, hogy hibákat követnénk el? Bár az ennél az összetevőnél megfogalmazott kérdés majdnem megegyezik az egyszerűségnél megfogalmazott kérdéssel, mégis egészen másról van szó: a hangsúly a hiba nélküli szoftverhasználaton van.

Minden programozó le tudja futtatni saját programját hiba nélkül, mert tudja, hogy programja mikor milyen inputot vár. A probléma kettős: a felhasználó nem biztos, hogy tudja ezt, illetve, indokolt-e az inputok korlátozása.

Hasznosság

Mennyire felel meg a szoftver a megrendelő/felhasználó igényeinek? Az adaptálhatósághoz nagyon közel álló minőségi összetevő, azonban a szoftver által a megrendelő/felhasználó számára nyújtott előnyökre (haszonra) helyezi a hangsúlyt.

Hatékonyság

Mennyire hatékonyan használja a szoftver a rendelkezésére bocsátott erőforrásokat? Ez a minőségi összetevő a szoftver teljesítményéhez kapcsolódó paraméterekre vonatkozik (gyorsaság, memóriaméret stb.) azonban ezeknek megfelelő voltát nem az igényspecifikációban megfogalmazott korlátokkal veti egybe, hanem általános szakmai elvárásokkal.

Hordozhatóság

Mennyire marad működőképes a szoftver más környezetben? A más környezet jelenthet másik gépet, más géptípust, más operációs rendszert, más hálózati szoftvert, összefoglaló néven más platformot.

Az összetevő által felvetett probléma természetes módon adódik: egy cég hálózatba kötött PC-ken üzemeltette raktárnyilvántartó rendszerét több éven keresztül, majd amikor ezt a konfigurációt kinőtte, IBM AS/400-as számítógép vásárlása mellett döntött. Átvihető-e a régi szoftver az új, nagyobb gépre, vagy új szoftver kifejlesztésére van szükség? (Tehát ez az összetevő nem olyan kérdésre irányul, hogy például működik-e a szoftver hordozható számítógépen is)

Igazolhatóság

Mennyire könnyen lehet igazolni, hogy a szoftver helyesen dolgozik? Ez a minőségi összetevő is több más összetevővel hozható kapcsolatba, így például a tesztelhetőséggel és a kifogástalansággal. A feltett kérdés hangsúlyában van különbség: a mindennapi szoftverhasználat közben végzett/végezhető ellenőrzésről van szó. Ha ezzel az összetevővel foglalkozunk, akkor azt kell megértetni a felhasználóval, hogy ha a szoftver nem küldött hibaüzenetet, akkor ez nem feltétlenül jelenti azt, hogy a feldolgozás hibátlan volt. A szoftver felhasználói kézikönyvében le kell írni azokat az ellenőrzési lehetőségeket, amelyeket a felhasználónak el kell végeznie, vagy maga a szoftver végez önellenőrzést. (Példaértékű annak az operátornak az esete, aki boldogan ment haza az éjszakai műszakból, amikor az egyébként nyolcórás listázó program két óra alatt lefutott ...)

Integrálhatóság

Egybeépíthető-e a szoftver más szoftverekkel? Manapság népszerű termék az „integrált szoftver". Ezekkel a szoftverekkel a felhasználó több különböző típusú műveletet hajthat végre többnyire ugyanolyan módon, azaz ugyanazokat a szerkesztő parancsokat használhatja például egy dokumentum, egy táblázatsor vagy egy adatbázisrekord szerkesztésére. Nem meglepő, hogy az integrált szoftverek a legjobban eladható programok. A felhasználók méltányolják, hogy korlátozott számú szabállyal több feladatot is elvégezhetnek.

Integritás

Mennyire képes a szoftver védeni magát a vírusfertőzésektől és a jogosulatlan hozzáférésektől? Ez a minőségi összetevő a „biztonság" összetevőnél említett károkozás és csalás elleni védekezésre vonatkozik. Külön akkor szokták megemlíteni, ha a szoftverbe épített védelmi technikákat akarják hangsúlyozni. (Például felhasználó azonosítás, ellenőrző mezők használata, adatok rejtjelezése stb.)

Interoperabilitás

Milyen könnyen tud a szoftver kommunikálni más szoftverekkel? A számítógépek elterjedésével együtt a legkülönfélébb alkalmazások készülnek, és egy bizonyos szint felett előtérbe kerül az alkalmazói szoftverek közötti kommunikáció. Kérdés, hogy a szoftverek ezt hogyan oldják meg?

Egyik lehetséges megoldás integrált szoftverek fejlesztése: természetes elvárás, hogy az ilyen szoftvereken belül az adatokat könnyen mozgathassuk egyik szoftverfunkcióból a másikba.

Másik megoldás, amikor a szoftverhez különböző interfészeket dolgoznak ki, már ismert szoftverekkel történő kommunikációhoz.

Harmadik megoldás, amikor az adatokat valamilyen alapformátumra konvertálják (pl. DOS esetén TXT kiterjesztésű szövegállomány).

Karbantarthatóság

Milyen olcsón lehet a szoftvert javítani vagy módosítani? Szintén egy nehezen ellenőrizhető, a megrendelők/felhasználók előtt rejtett minőségi összetevő. Csak a szervezett fejlesztő munkával, programkódolási szabályok (szabványok), TQM módszerek alkalmazásával garantálható.

Kifogástalanság

A szoftver az igényspecifikáció szerint helyesen működik-e? Ez az összetevő azt vizsgálja, hogy a szoftver mennyiben felel meg az igényspecifikációnak, a szoftverekkel szembeni általános elvárásoknak, a minőségi követelményeknek. A szoftveralkalmasság egyik alkotórésze az adaptálhatóság és a befejezettség mellett. Előfeltétele a tesztelhetőség.

Megbízhatóság

Mennyi ideig képes a szoftver úgy működni, ahogy azt az igényspecifikáció előírja? Milyen gyakorisággal fordulnak elő hibák, és az előforduló hibák mennyire kritikusak? Látható, hogy a

szakirodalom ezzel a minőségi összetevővel foglalkozik a legtöbbet. A megbízhatóság mérésének külön irodalma van, amelynek áttékintésére itt nincs lehetőség. Bár a szoftverminőség egész kérdésköre kapcsolatban van a TQM néven ismert menedzsment-irányzattal, azonban a TQM fő támadási felülete a hibák minimalizálása. Csak néhány adat Arthur [1993] könyvéből: egy szokványos amerikai üzleti program 3000 hibát tartalmaz egymillió programsorban, az űrsikló programja 70-et, a Fujitsu 10-et, a Motorola 3.4-et ért el 1994-ben. (Egy másik érdekes információ: az USA-ban többen lehetetlennek tartották az SDI ballisztikus rakétaelhárító rendszer irányító szoftverjének kifejlesztését, mivel annak becsült mérete 10 millió programsor volt, kifejlesztéséhez a pesszimista becslés szerint 80 ezer emberév kellett volna; az ellenzők nem láttak lehetőséget egy ilyen tömegű munka hiba nélküli elvégzésére.)

Rugalmasság

Mennyire olcsón lehet a szoftvert módosítani egy olyan igény kielégítésére, amelyet eredetileg nem szerepelt az igényspecifikációban? A karbantarthatósággal van kapcsolatban ez az összetevő, de annak egy speciális dimenzióját ragadja meg. Azokat a szoftvereket nevezzük rugalmasnak, amelyek módosítása semmibe se kerül, ha egy új igény kielégítését várják el tőle. Azaz, a fejlesztő csoport előrelátásáról van szó, hogy mennyiben tudják előrevetíteni a szoftverrel szemben támasztott jövőbeli igényeket, és ezekre mennyiben tudják felkészíteni a szoftvert. (Ellenpélda annak a programozónak az esete, akinek a bérszámfejtő rendszerével komoly gondok támadtak akkor, amikor a munkavállalalói járadék úgy változott, hogy fél százalékra végződött. Az ő rendszere csak egészértékű százalékokat tudott kezelni.)

Teljesítmény

Mennyire felel meg a szoftver az igényspecifikációban meghatározott teljesítménykövetelményeknek? Nem elegendő, ha a szoftver csak kifogástalan, a teljesítménykövetelményeknek is eleget kell tennie (gyorsaság, memóriaigény, háttértárak mérete stb.) Általában speciális méréseket, esetleg külön tesztprogramokat igényel ennek a minőségi összetevőnek a vizsgálata.

Tesztelhetőség

Milyen mértékben ellenőrizhető, hogy a szoftver kifogástalan? Ez a minőségi összetevő erősen kapcsolódik az igazolhatósághoz, csak míg az igazolhatóság a szoftverműködés ellenőrzését hangsúlyozza, a tesztelhetőség a megfelelő szoftver vizsgálatára vonatkozik, azaz arra, hogy milyen módszerekkel, hogyan ellenőrizhető a kifogástalanság.

Túlélési képesség

Hogyan képes a szoftver teljesíteni a funkcióit a hardver üzemzavara esetén? Ez az összetevő a hardverhibák kezelésére vonatkozik. Sajnos, ezekre mindig fel kell készülni, mert a legüzembiztosabb számítógéppel is előfordulhat, hogy a felhasználó fordítva teszi be a mágneslemezt a meghajtóba; hiába van a számítógép szünetmentes tápegységre csatlakoztatva, ha azt a felhasználó elfelejti bekapcsolni stb. Két kérdést kell ezzel kapcsolatban vizsgálni: érzékeli-e a szoftver a hardverhibát, illetve a hardverhiba milyen kárt okoz a szoftver által kezelt/tárolt adatokban. Ide tartozik a szoftver újraindíthatóságának kérdése is.

Újrahasznosíthatóság

Milyen könnyen használható a szoftver vagy annak részei egy másik rendszerben? Az itt felsorolt minőségi összetevők közül talán ez a legérdekesebb. Ha egy szoftver teljesíti az újrahasznosíthatóság követelményeit, annak előnyei többségükben a fejlesztő csoport számára jelentkeznek. Az újrahasznosíthatóság lényege: a szoftvert úgy fejlesztik ki, hogy annak egyes részei, moduljai más szoftverekben is használhatók legyenek. Előnye, hogy az új szoftverbe kevesebb munkával kipróbált programrészek, modulok kerülnek, így továbbadható egy az előző szoftverben megvalósított minőség.

Összefoglalás

A szoftver minőségét tehát számos összetevő együttes megvalósulása garantálja, amely összetevők a fejlesztés alapvető szempontjaiként is értelmezhetők. Ezen minőségi összetevők származhatnak mind a megrendelőktől és leendő felhasználóktól, mind pedig magától a fejlesztő csoporttól. Az összetevők ismerete azonban a fejlesztők és nem fejlesztők számára egyaránt fontos, és a minőség biztosításának be kell épülnie a fejlesztési folyamat során a megrendelők és a fejlesztő csoport között kiépülő kommunikációba.

Irodalom

Arthur, Lowell J. (1993): Improving Software Quality. New York: John Wiley & Sons

Hall, Tracy (1995): Confessions of a Software Quality Addict. (In: Professional Awareness in Software Engineering. Ed.: Colin Myers) London: McGraw-Hill Book Company

Kan, Stephen H. (1995): Metrics and Models in Software Quality Engineering. Reading: Addison-Wesley Publishing Company

Kotler, Philip (1991): Marketing Management: Analysis, Planning, Implementatioon and Control (7/e.) London: Prentice-Hall International

Lin, Herbert (1986): „Szoftverfejlesztés ballisztikus rakéta elhárításához." *Tudomány,* 1986. február, 14–22 o.

Ould, Martyn A. (1991): Quality control and assurance. (In: Software Engineer's Reference Book, Ed.: John McDermid) Oxford: Butterworth-Heinemann Ltd.

Sage, Andrew P. and James D. Palmer (1990): Software Systems Engineering. New York: John Wiley & Sons

Sage, Andrew P. (1992): Systems Engineering. New York: John Wiley & Sons

Schach, Stephen R. (1993): Software Engineering. Homewood, Ill.: Irwin

Sommerville, Ian (1992): Software Engineering. Addison-Wesley Publishing Company

Stanton, William-Michael. J. Etzel-Bruce J. Walker (1991): Fundamentals of Marketing (9/e.) New York: McGraw-Hill

Steward, Donald V. (1987): Software Engineering. Monterey: Brooks/Cole Publishing Company