X. FIATAL MŰSZAKIAK TUDOMÁNYOS ÜLÉSSZAKA

Kolozsvár, 2005. március 18-19.

SZOFTVERVIZSGÁLAT: GYAKORLAT-ORIENTÁLT OLVASÁS AZ EVOLÚCIÓS VAGY BOEHM-FÉLE FEJLESZTÉSI MODELLEK ALAPJÁN FEJLESZTETT SZOFTVERBEN

Görög Levente-Károly, Máthé Zsolt

Abstract – Software inspection is a method to monitor and control the software quality throughout the development cycle. Reading techniques help reviewers to stay focused on the important parts of a software artifact. Usage-Based Reading is a special reading technique, where the artifact is inspected from the user's point of view. Besides Usage-Based Reading, there have been developed other reading techniques, which differ from each other in both their approach and methodology. The debate over which technique performs better is still going on. Many experiments have been carried out, testing the efficiency and effectiveness of different reading techniques. However, the reliability of the results is controversial. This paper presents, why it is advantageous and cost-effective the Usage-Based Reading by software designed on the basis of Evolutionary or Boehm's Spiral Development Models.

Összefoglalás – A szoftvervizsgálat módszer a szoftver minőségi ellenőrzésére a fejlesztési ciklus során. A recenzenseket (szoftvervizsgálókat) különböző olvasási technikák segítik a gyors orientálódásban és a szoftverdokumentumok releváns részeire való fókuszálásban. A Gyakorlat-Orientált Olvasás olyan technika, amely a felhasználó szempontjai alapján vizsgálja a szoftverdokumentumokat. A Gyakorlat-Orientált Olvasás mellett léteznek más olvasási technikák is, amelyek mind megközelítésben, mind módszerekben különböznek egymástól. Egyelőre nyitott kérdés, melyik olvasási technika használata előnyösebb, bár számos kísérletet végeztek a különböző technikák hatékonyságának és hatásosságának tesztelésére. Mindazonáltal, ezeknek a kísérleteknek az eredményei vitathatóak. Jelen tanulmány bemutatja, miért előnyös és költség-hatékony a Gyakorlat-Orientált Olvasás az Evolúciós vagy Boehm-féle spirál modellek alapján tervezett szoftverek esetében.

Bevezetés

A szoftvervizsgálat már több mint 25 éves múltra tekint vissza. Mind akadémiai-tudományos, mind pedig ipari körökben egyre nagyobb jelentőséggel bír, arányosan a számítógépen megoldandó feladatok komplexitásának növekedésével. A szoftvertechnológia egyik álagazata, és definíció szerint elhatárolódik a szoftvertesztelésztől, bár kétségtelenül vannak átfedések a két szakterület között. A szoftvervizsgálat célja a szoftvertervben (követelmény-dokumentumok, design-dokumentumok, használati eset-diagramok), valamint a par excellence megvalósításban (például forráskód) található hibák detektálása. Ennek megfelelően a szoftvervizsgálat egyaránt a Követelmény-technológiában

(Requirements Engineering – RE), illetve a Verifikáció és Validáció (Verification and Validation – V&V) folyamatában jelen van [4]. A szoftervizsgálat leginkább használt módszerei az ún. olvasási technikák (reading techniques), amelyek segítenek a recenzenseknek a vizsgált dokumentum releváns részeire való fókuszálásban, annak jobb átlátásában, a gyorsabb tájékozódásban, összegzésképpen tehát a hibafelderítés hatékonyságának növelésében. A legismertebb olvasási technikák közé tartozik a *katalógus-orientált (checklist-based)* [2], *perspektíva-orientált (perspective-based)* [1][5], *hiba-orientált (defect-based)* [3], *gyakorlat-orientált (usage-based)* [6] és *kinyomozhatóság-orientált (traceability-based)* [7] olvasás. A **katalógus-orientált olvasás** (CBR) a klasszikus standardnak minősül a legtöbb szoftverszervezet esetében. A recenzensek egy lista alapján, a listában felsorolt hibatípusokat keresve vizsgálják át a szoftverdokumentumot. A **perspektíva-orientált olvasás** (PBR) különböző szerepeket, perspektívákat rendel a recenzensekhez (Basili et al. [1] dolgozatában designer-i, tesztelői és felhasználói szerepeket különböztet meg). A **hiba-orientált olvasás** (DBR) többféle scenáriót definiál a speciális hibák detektálására. A scenáriók ebben a kontextusban eljárásokat jelentenek, amelyeket követve a szoftervizsgálók ugyanazokat a hibákat hivatottak megtalálni, mint a CBR esetében. A **kinyomozhatóság-orientált olvasás** (TBR) az objektum-orientált design specifikációk nyomon követésére használatos.

A gyakorlat-orientált olvasás

A gyakorlat-orientált vagy használat-alapú olvasás (UBR) a vizsgált objektum *legkritikusabb* hibáira fókuszál. Ezek jelentősége a felhasználó szempontjából konkretizálódik: milyen mértékben érinti egy adott hiba a felhasználás során leginkább használt / legfontosabb operációkat. Ez az olvasási technika használati eset-diagramok, más néven scenáriók használatára épül, amelyek a felhasználónak a rendszerről alkotott képével vannak szinkronban (1. Ábra). Az UBR a következő lépések egymás utáni elvégzését jelenti:

Taxi: bejelentkezés

Direktíva: A vezető bejelentkezik a rendszerbe, bekerül a nyilvántartásba és képes lesz rendelések fogadására.

Feladatok:

1. Az autó *offline* állapotban van.
2. A vezető beteszi azonosító kártyáját az autó termináljába.
3. A terminál elküldi a vezető adatait a központhoz.
4. A központ konfirmálja a bejelentkezést.
5. A terminál elküldi az autó pozícióját a központnak.
6. A központ elkezd továbbítani ezeket az információkat az összes zónának
7. Az autó *hozzáférhető* állapotban van

Variánsok:

- 1b. Az autó nincsen *offline* állapotban (például az előző vezető nem jelentkezett ki). Az autót *offline* állapotba kell hozni.
- 4b. Az azonosító kártya nem érvényes. A bejelentkezés nem történik meg, az autó *offline* állapotban marad.

1. Az egyes S_i scenáriók p_i prioritásának meghatározása.
2. A szoftverdokumentum teljes vizsgálatára szánt T idő definiálása.
3. A $T_i = p_i * T$ részüidők hozzárendelése az egyes S_i scenáriókhoz.
4. A szoftverdokumentum

1. Ábra: feladatjelzéssel elkészített adatfolyam: taxi bejelentkezés [6].

vizsgálata az S_i scenáriók alapján, esetenként T_i ideig: helyes-e a vizsgált dokumentumrész S_i -nek megfelelően? A fentiekből látható, hogy a gyakorlat-orientált olvasás, ellentétben több más olvasási technikával, feladatspecifikus. Minden egyes szoftver esetében a vizsgált dokumentumokhoz más- és más scenáriók szükségesek, ellentétben például a CBR-rel, amely általános szempontok (egységesség, konzisztencia, helyesség, teljesség) alapján segíti a szoftvertesztelőket az orientálódásban. A feladatspecifikus használati eset-diagramok kevésbé költség-hatékonyak, és egyes szerzők szerint az UBR átlagos performanciája nem jobb (sőt esetenként rosszabb) más olvasási technikák teljesítményénél. A vélemények csupán abban egységesek, hogy a kivitelezendő projekt jellege illetve a vizsgált szoftverdokumentum típusa nagymértékben leszűkíti a használható olvasási technikák halmazát¹.

Gyakorlat-orientált olvasás az Evolúciós vagy Boehm-féle fejlesztési modellek alapján fejlesztett szoftverben

Regnell at al. *Combining Scenario-based Requirements with Static Verification and Dynamic Testing* [4] című munkájában javaslatokat tesz a scenárióknak a szoftvertesztelésben történő újrafelhasználására. Szerinte könnyűszerrel lehet scenáriókat készíteni már a követelménydokumentumokból (*requirement documents*); ezeket a scenáriókat pedig úgy a statikus verifikációban, mint a dinamikus tesztelésben használni lehet transzformáció, illetve extenzió révén. Regnell fenntartja, hogy az esetek többségében a használati eset-diagramok megfelelőek a tesztesetek generálására, mivel mindkettő egy külső nézőpontból (a felhasználó szemszögéből) tekint a rendszerre. Ehhez azonban a következő feltételeknek kell teljesülniük: 1). a RE folyamatában készült, megfelelően dokumentált scenáriók el vannak mentve tekintettel a fejlesztés későbbi fázisaira. 2). a scenáriókat folyamatosan bővítik, változtatják, kiegészítik stb. a fejlesztés során. Ezek a követelmények alapvetőek arra nézve, hogy a scenáriók ne legyenek elavultak, amidőn a tesztelési

2. Ábra: Evolúciós modell

szakaszban újrafelhasználnák őket. A használati eset-diagramok folyamatos átírása, bővítése azonban rendkívül nehézkes lehet olyan esetekben, amikor a RE és a V&V között sok idő telik el. Inkább olyan fejlesztési modellekben lehet eredményes ez a technika, ahol a fejlesztés fázisai között viszonylag kevés az időkülönbség, vagyis az

¹ UBR-re készített scenáriókat nem érdemes UBR-rel vizsgálni, megfelelőbb ilyen esetben a CBR használata. Az UBR viszont kiválóan alkalmas design-dokumentumok (főként *specification and description language* – SDL-ben megírt) ellenőrzésére, esetenként akár forráskód vizsgálatára is.

3. Ábra: Boehm-féle spirál

Evolúciós modellben (2. Ábra) és a Boehm-féle spirál modellben (3. Ábra). Az ezekre a modellekre alapuló szoftverfejlesztésben a scenáriók alapjául szolgáló követelmény dokumentumokat minden fejlesztési fázisban frissíteni kell a továbblépés érdekében, míg a különböző fejlesztési szakaszok közötti időkülönbség meglehetősen kicsi – a modellek természetéből adandóan.

Következtetések, távolabbi perspektívák

Észrevételeim szerint az Evolúciós, illetve Boehm-modelleken alapuló szoftverfejlesztés esetében költség-hatékony a gyakorlat orientált olvasás, mivel 1). a követelmény dokumentumokat folyamatosan frissítik – a modellek természetéből adandóan 2). a követelmény dokumentumokból nyert scenáriók újrafelhasználhatóak a tesztelési fázisban (a folyamatos frissítés következtében mindig aktualizáltak maradnak). Megfontolásaim alapul szolgálhatnak a gyakorlat-orientált olvasás hatékonyságát vizsgáló kísérleteknek.

Hivatkozások

- [1] V.R. Basili, S. Green, O. Laitenberger, F. Lanubile, F. Shull, S. Sørunga°rd, and M.V. Zelkowitz, "The Empirical Investigation of Perspective-Based Reading," *Empirical Software Eng.: An Int'l J.*, vol. 1, no. 2, pp. 133-164, 1996.
- [2] M.E. Fagan, "Design and Code Inspections to Reduce Errors in Program Development," *IBM System J.*, vol. 15, no. 3, pp. 182-211, 1976.
- [3] A. Porter, L. Votta, and V.R. Basili, "Comparing Detection Methods for Software Requirements Inspection: A Replicated Experiment," *IEEE Trans. Software Eng.*, vol. 21, no. 6, pp. 563-575, June 1995.
- [4] B. Regnell, and P. Runeson, "Combining Scenario-Based Requirements with Static Verification and Dynamic Testing," *Fourth International Workshop on Requirements Engineering: Foundations of Software Quality*, June 1998.
- [5] F. Shull, R. Ioana, and V.R. Basili, "How Perspective-Based Reading Can Improve Requirements Inspections," *Computer*, vol. 33, no. 7, pp. 73-79, July 2000.
- [6] T. Thelin, P. Runeson, and Claes Wohlin, "An Experimental Comparison of Usage-Based and Checklist-Based Reading," *IEEE Transactions on Software Engineering*, vol. 29, no. 8, pp. 687-704, 2003.
- [7] G. Travassos, F. Shull, M. Fredericks, and V.R. Basili, "Detecting Defects in Object-Oriented Designs: Using Reading Techniques to Increase Software Quality," *Proc. Int'l Conf. Object-Oriented Programming Systems, Languages & Applications*, 1999.

Görög Levente-Károly / egyetemista III. év
 Sapientia egyetem / Iceland 68, MS 547218
 Tel., email: 0742606039, goroglev@yahoo.com