Teszt generálás webes alkalmazásokhoz

Írásos összefoglaló Pan Liu, Huaikou Miao, Hongwei Zeng és Linzhi Cai An Approach to *Test Generation for Web Applications* [1] c. munkájáról.

Készítette: Doktor Tibor

Budapest, 2013. január

1 Bevezetés

Napjaink szoftvereinek egyre növekvő hányadát adják a webes alkalmazások, melyek helyes működésének ellenőrzése azonban új kihívások elé állítja a hagyományos szoftver tesztelési módszereket, aminek oka az ilyen alkalmazások velejáró komplexitása.

Jellemzően a szoftver-minőségbiztosítási csapat felelős a webes alkalmazások teszteléséért, melyet manuálisan hajtanak végre. Ez azonban a webes alkalmazások összetettségének növekedésével egyre nehezebben vagy egyáltalán nem megvalósítható. Egyrészt adódik ez abból, hogy a hibák reprodukálása egyre nehezebbé válik a tesztelő számára a végrehajtott műveletek számának növekedésével. Másrészt a manuális tesztelés időigényes és fárasztó feladat a tesztelést végző személy számára, főleg ha azt a regressziós tesztek végrehajtásakor meg is kell ismételni.

Emiatt, a manuális tesztek hiányosságainak megszüntetése érdekében az automatikus teszt generálás egyre fontosabb területté válik a kutatásokban. A modell alapú tesztelés hatékony automatizált teszt generálási módszer a web alkalmazások számára, ahol az alapötlet, a szoftver tervezése során definiált modellek felhasználása a tesztelés vezérlésére, különösen a teszt esetek automatizált generálása során. A hibák detektálása a modell és a szoftver működése között megfigyelt inkonzisztencia felderítésével történik. Emiatt szükséges hogy a modell helyes és a belőle származtatott teszt utak is helyesek és hatékonyak legyenek.

Web alkalmazásoknál gyakran alkalmazott módszer modell alapú tesztelés esetén a teszt fa segítségével generált tesztekkel történő verifikáció. Teszt fa felhasználásával tesztelhető a web alkalmazás használhatósága az egyes elemek elérhetőségének aspektusából. Ezekben a fákban a gyökerétől a levelekig terjedő teszt sorozatok definiálják a teszt utakat a web alkalmazások számára. Azonban a módszer egyik hiányossága, hogy pl. ha a web alkalmazásban különböző, eltérő jogosultságokkal rendelkező felhasználói szerepkörök is jelen vannak, a fából felesleges és nem biztonságos teszt utak is származtathatók, amely akár egy adott szerepkörhöz nem hozzáférhető funkció tesztelését is elvárja.

Az összefoglaló alapjául szolgáló dokumentum ez utóbbi problémára ad egy lehetséges megoldást, amely felhasználva modell ellenőrzési technikákat, a web alkalmazások navigációs modelljének dekompozíciójával szűri ki a szükségtelen, felesleges teszt utakat.

A szerzők által ajánlott módszer alapján ehhez először Kripke struktúrával megalkotásra kerül az adott web alkalmazás navigációs modellje, majd a modell particionálásából előálló al-modellekből hozza létre a teszt generáláshoz szükséges teszt fákat.

2 A navigációs modell

Mint az a bevezetésben is említésre kerül a jelen dokumentumban bemutatott tesztelési megközelítés a web alkalmazások navigációjának viselkedésére helyezi a hangsúlyt.

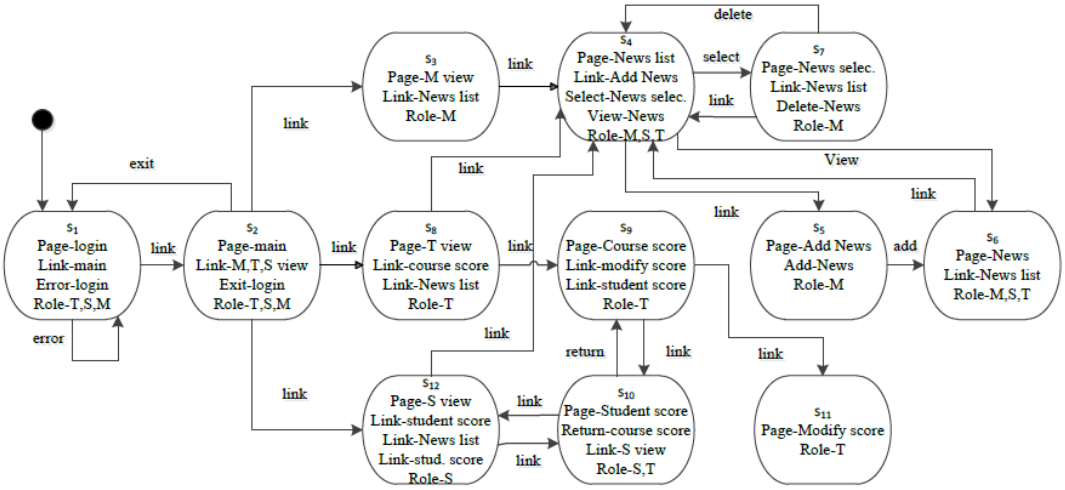
Egy web alkalmazáson belül a navigáció egy klasszikus állapot átmeneti gráfnak tekinthető, ahol a pontok az elérhető állapotok (oldalak) az élek pedig az azok közötti átmeneteket (navigációt) jelentik.

A szerzők által ajánlott navigációs modell definíciója a következő:

1. Definició (Navigációs modell). A web alkalmazások navigációs modellje felírható $PN = (S, Init, AP, R, R, L)$ alakban, ahol S az állapotok véges halmaza, amely a web alkalmazás oldalait jelöli, $Init \subseteq S$ a kezdő állapotok halmaza, AP atomi kijelentések halmaza, $R \subseteq S \times S \times AP$ egy állapotátmeneti reláció oly módon, hogy minden $s \in S$ állapothoz létezik egy $t \in S$ állapot és egy $p \in AP$ atomi kijelentés amely kielégíti a $(s, t, p) \in R$ relációt, $L : S \rightarrow 2^{AP}$ pedig egy állapot függvény amely minden állapotot megcímkéz egy atomi kijelentés halmazzal.

A definícióban definiált S halmaz a web alkalmazás minden weboldalát tartalmazza, függetlenül attól, hogy az egy statikus vagy dinamikusan generált lapot jelent-e. A bejelentkezési oldalakat a szerzők egyedi kezdő állapotnak tekintik. Az atomi kijelentések az egyes oldalak tulajdonságait, a szerepköröket és a web alkalmazásnak küldött kéréseket írják le a modellben. A szerzők több ilyen atomi kijelentést definiáltak munkájukban, az oldalakat reprezentáló „Page<page name>”, webalkalmazás felhasználói szerepköreit reprezentáló „Role<role_name>”, a linkeket jelentő „Link<page_name>”, a hibás bejelentkezéskor megjelenő „Error<page_name>”, és a „Return<page _name>” kijelentést, mely a kérés esetén visszatérő weboldalt írja le.

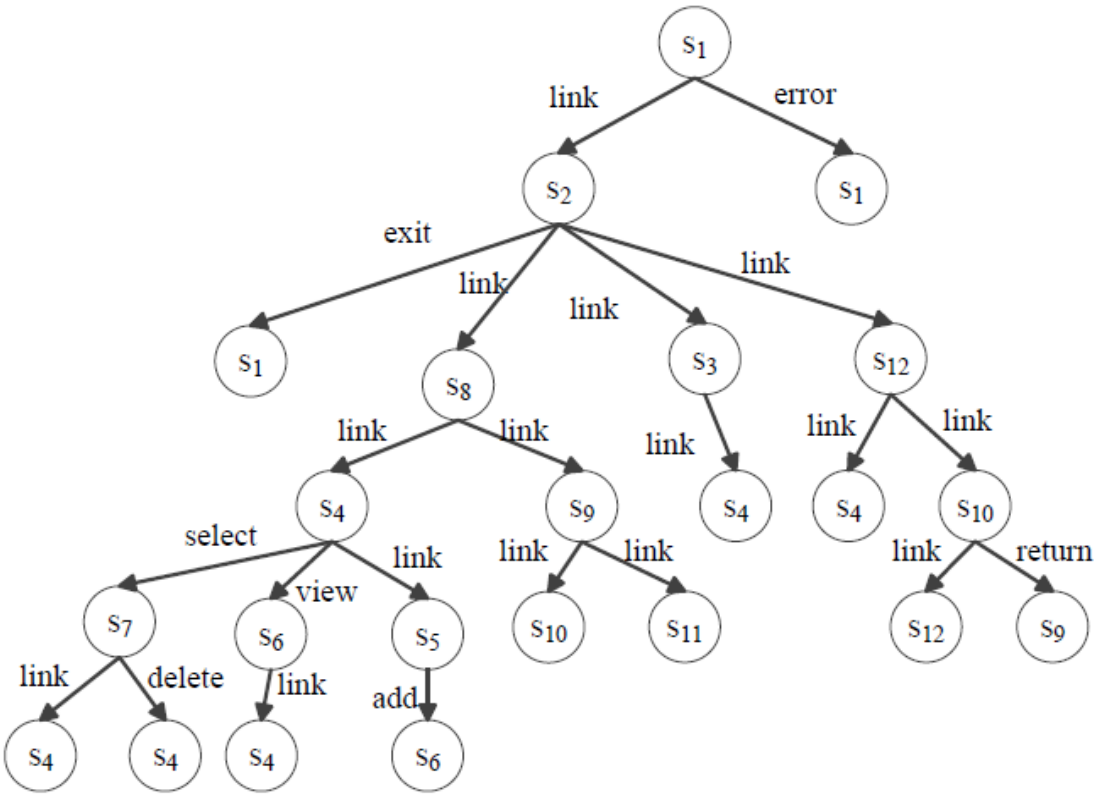
A jelen összefoglalóban bemutatott publikáció alkotói a modell használatát egy hallgatói kurzusokat menedzselő web alkalmazás példáján keresztül mutatják be. A példa alkalmazás navigációs modelljét az 1. Ábra mutatja be.



1. Ábra: A hallgatói kurzus menedzselő rendszer navigációs modellje

Az alkalmazás három különböző felhasználói szerepkört definiál, a diák, tanár és menedzser szerepköröket. A tanár szerepkör rögzíthet, módosíthat és ellenőrizhet hallgatókhoz tartozó eredményeket az adott kurzushoz. A hallgatói szerepkörhöz ezzel szemben csupán a hallgató saját eredményének megtekintéséhez tartozó jogosultság tartozik. A menedzser szerepkör pedig lehetővé teszi hírek felvételét, törlését és böngészését a rendszerben.

A szerzők ezután bemutatják a teszt fa konstruálásának ebben a fázisban történő hátrányát. A 2. Ábra által bemutatott tesztfából generált teszt utak olyan teszt utakat is tartalmaznak, amelyek hatástalanok vagy nem végrehajtatók, mert az adott felhasználói szerepkörnek nincs jogosultsága az adott művelet végrehajtására. Erre egy példa a $\langle s_1, s_2, s_{12}, s_{10}, s_9 \rangle$ útvonal, mivel a hallgatóknak nincs jogosultságuk a rendszerben az eredmények módosítására.



2. Ábra: A navigációs modellből létrehozott teszt fa

A szerzők a fenti helytelen navigációs utak megkülönböztetésére munkájukban bevezették a hatékony navigációs viselkedés definícióját:

2. Definíció (hatékony navigációs viselkedés). Legyen $F:U\rightarrow 2^{A}$, ahol $U\left(U\neq\emptyset\right)$ jelöli a felhasználói szerepköröket a web alkalmazásokban, $A\left(A\neq\emptyset\right)$ pedig a szerepköri navigációs műveletek halmazát. $\forall u\in U$ esetén *F(u)* jelöli az *u* szerepkörhöz tartozó hatékony navigációs viselkedések halmazát.

A fenti definícióban bemutatott hatékony navigációs viselkedések halmazának előállítására a szerzők a modell ellenőrzés technikáját alkalmazták. Ehhez a CTL temporális logika segítségével formalizálták a szerepkörök tulajdonságait az alkotók, majd ezt a navigációs modellhez csatolva adták bemenetként a NuSMV eszköznek a modell ellenőrzés végrehajtásához. Ha az eszköz ellenpéldát talált, a modellből eliminálták a nem biztonságos viselkedéseket az adott szerepkörhöz, amelynek eredményeként előálltak a hatékony navigációs viselkedést tartalmazó al-modellek.

3 Az alkalmazott CTL formula

A szerzők az előző fejezetben hivatkozott modell ellenőrzéshez alkalmazott CTL kifejezéseket Backus-Naur formában definiálták a következő definíció segítségével:

3. Definíció (Computation Tree Logic, CTL). Esetén a CTL formula definiálható Backus-Naur

formában:

$$\phi ::= p \mid \neg\phi \mid \phi \wedge \varphi \mid \phi \vee \varphi \mid \phi \rightarrow \varphi \mid \mathbf{AX}\phi \mid \mathbf{EX}\phi \mid$$
$$\mathbf{AF}\phi \mid \mathbf{EF}\phi \mid \mathbf{AG}\phi \mid \mathbf{EG}\phi \mid \mathbf{A}[\phi\,\mathbf{U}\,\varphi] \mid \mathbf{E}[\phi\,\mathbf{U}\,\varphi]$$

ahol p az atomi kifejezések tartománya φ és ϕ pedig CTL formulák.

Néhány példa a szerzők által a példa alkalmazás modell ellenőrzésére használt CTL kifejezésekből:

- A hallgatók jogosultak a hírek oldal megtekintésére, de nem érhetik el a hírek hozzáadására szolgáló oldalt.

$$AG\left(\text{Role}-\text{S} \rightarrow \neg\left(\text{link} \wedge \text{EF Page}-\text{add News}\right)\right)$$

- Az oktatók nem érhetik el a hallgatói adatokat tartalmazó oldalakat, hogy megtekintsék a személyes adatokat.

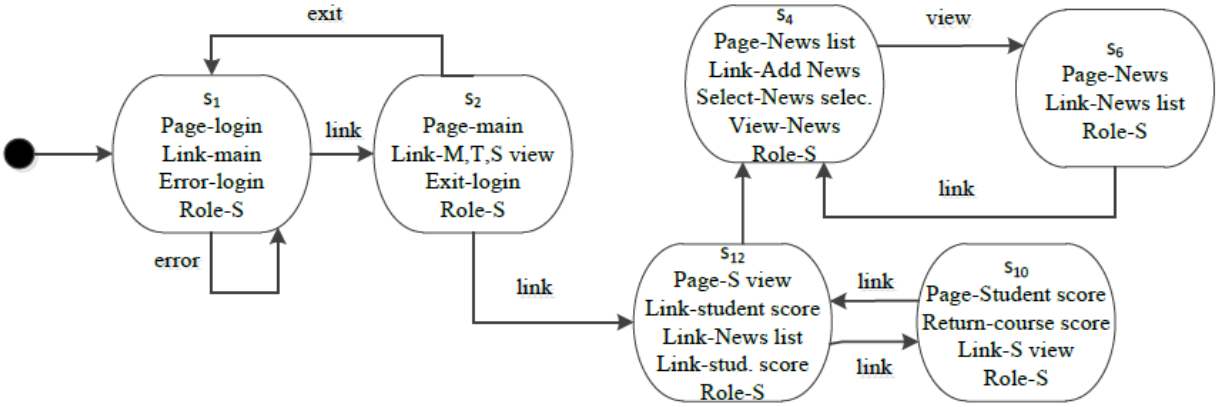
$$AG\left(\text{Role}-\text{T} \rightarrow \neg\left(\text{link} \wedge \text{EF Page}-\text{S view}\right)\right)$$

4 Modell particionálás

A szerzők által ajánlott módszer alapját a Kripke struktúrában előállt navigációs modell dekompozíciója jelenti. Ez a modell nem hatékony navigációs elemeinek eliminálását jelenti mindaddig míg a modell ellenőrzés során ellenpéldát kapunk eredményül. A partícionálás részletes lépései a következők:

1. A modell ellenőrző futtatása a felhasználói szerepkörhöz tartozó kifejezések és a navigációs modellel bemenetként.
2. Ha ellenpélda jelenik meg kimenetként, egy szükségtelen navigáció viselkedés eltávolításra kerül a modellből.
3. A 2. lépés ismétlése mindaddig a kimeneten ellenpélda jelenik meg.
4. Manuálisan minden olyan állapot és ahhoz kapcsolódó tranzíció törlése a modellből, ami a kezdő állapotból nem elérhető, ezek után pedig elő áll az adott u felhasználói szerepkörhöz tartozó al-modell.

A 3. Ábra a hallgatói szerepkörhöz tartozó al-modellt mutatja be.



3. Ábra: A hallgatói szerepkörhöz tartozó navigációs al-modell

5 Teszt generálás és a redundancia csökkentése

A szerzők a különböző szerepkörökhöz tartozó al-modellek bejárásához a branch-first algoritmust alkalmazták. Azonban az így előálló teszt fák tartalmazhatnak redundáns utakat, amik szükségessé teszik a redukciós technikák alkalmazását. Ezt a szerzők az állapot lefedettség vizsgálatával valósították meg.

6 Konklúzió

Az itt bemutatott módszer egy formális megközelítést alkalmaz webes alkalmazások tesztelésére, annak navigációs viselkedésére koncentrálva. A módszer három fázisra osztható. Első lépésben a web alkalmazás navigációs modellje kerül megalkotásra, Kripke struktúrák segítségével. A második lépésben a modell partíciókra bontása valósul meg a modell ellenőrzési technika segítségével. A módszer befejező fázisban az előző fázisban elő állt al-modellekben még jelenlévő redundáns teszt utak eltávolítása történik meg.

7 Irodalom

(1) P. Liu, H. Miao, H. Zeng and Lizhi Cai, „An Approach to Test Generation for Web Applications” International Journal of u- and e- Service, Sience and Technology, Vol. 6., No. 1. , pages 61-76, february 2013