.NET alapú programok minőségének és biztonságának növelése

Doktori értekezés tézisei

Pócza Krisztián

Témavezető:
Dr. Porkoláb Zoltán

Eötvös Loránd Tudományegyetem
Informatika Doktori Iskola
Az informatika alapjai és módszertana doktori program
Iskola- és programvezető: Dr. Demetrovics János

Budapest, 2010

1 Bevezetés

A szabványos Microsoft .NET technológia a 2002-es megjelenése óta dinamikus fejlődésen ment keresztül. Fontos megemlíteni, hogy mivel egy átgondolt és jól megtervezett keretrendszerről beszélünk, ezért a legelső verzióban napvilágot látott koncepciók még ma is megállják a helyüket. Mind a keretrendszer, mind pedig a nyelvi támogatás szempontjából a folyamatos és dinamikus bővülés jellemezte.

A .NET keretrendszer már önmagában is segíti azt, hogy magas minőségi és biztonsági paraméterekkel rendelkező programokat állíthassunk elő. Ilyen eszközök például a felügyelt hivatkozások kezelése, puffer túlcsordulások elleni védelem, valamint a futás idejű típusvizsgálat. Azonban vannak olyan területek, amelyeket ez a rendszer sem fed le keretrendszer, alaprendszer mivoltából adódóan. Azt is figyelembe kell venni, hogy konkrét programok (pl. elosztott alkalmazások) alkalmazásszintű biztonságának megőrzése, minőségének növelése még mindig az alkalmazás tervezőjének illetve kivitelezőjének a feladata. Ez az a terület, amely tudományos szempontból is rendelkezik kutatási lehetőségekkel.

Az általában tágan értelmezett biztonság és minőség fogalmának egy olyan leszűkítését vizsgálom, amely a szoftverfejlesztés fázisaira koncentrál. Felvázolom, csoportosítom azokat a módszereket, amelyek a programok minőségi és biztonsági jellemzőit javítják.

Az elosztott alkalmazások publikus szolgáltatásinak hozzáférés-vezérlése széles körben vizsgált terület. Ennek a kutatásának az egyik korai eredménye az Eiffel nyelv szofisztikált hozzáférés vezérlési mechanizmusaihoz köthető. Ugyanakkor nincs olyan ismert általános megoldás, amely kiterjesztené az elosztott alkalmazások kontextusába az Eiffel nyelvben megismert hozzáférés-vezérlő mechanizmusokat. További gond az, hogy az elosztott rendszerekben futó munkafolyamatok nem kapcsolódnak szorosan össze biztonsági és vezérlési szempontból azokkal a szolgáltatásinterfészekkel, amelyek a munkafolyamatok publikus homlokzatát adják. Deklaratív eszközökkel akár egy metódus szintű szerepkör-szabályhierarchia alapú felhasználói szintű jogosultsági rendszer, hálózati szegmenseket korlátozó mechanizmus is kialakítható. Általános probléma az elérhető keretrendszerek tekintetében ezen felül az, hogy a jogosultságok korlátozása nem megfelelő granularitási szinten, azaz nem metódus szinten kerül megvalósításra. Ahhoz,

hogy ezek a problémák általánosan kezelhetők legyenek, egy formális modellre van szükség, amely biztosítja a platformfüggetlenséget. Ezzel foglalkozik az első tézis.

A formális modell alapján implementáció is készíthető. Azt már a modell kialakítása során figyelembe kell venni, hogy a formális megfogalmazásból az implementáció könnyen elkészíthető legyen. Cél az, hogy a formális modellből egy bijektív átírási rendszer segítségével platformfüggő megoldás legyen elkészíthető. A .NET keretrendszert választottam az átírás implementációs célplatformjának. Ezzel a második tézis témája.

A .NET keretrendszer esetében nem létezik olyan naplózó mechanizmus, amely utasítás vagy akár változó használat szintű naplóbejegyzéseket tudna létrehozni. Egy ilyen naplózó megoldás a dinamikus programszeletelés bemeneteként, valamint a fejlesztési, illetve tesztelési fázisban is felhasználható lehet. A dinamikus szeletelő algoritmusok egy olyan naplót igényelnek, amelyek a végrehajtott utasításokon kívül célszerűen az olvasott, valamint a módosított változókat is tartalmazza. Másik fontos használati eset az, amikor a naplót a program lefutása után elemezzük. A napló rengeteg olyan információt tud nyújtani, amely a Debugging során nem vagy csak nehezen lenne megszerezhető. Ez a megoldás leginkább a többszálú programok esetében lehet érdekes, ott szolgálhat több előremutató eredménnyel. Ezzel foglalkozik a harmadik tézis.

2 A dolgozat céljai és módszerei

Kutatásom során olyan módszereket dolgoztam ki, amelyek az önálló alkalmazások, illetve az elosztott alkalmazások minőségi és biztonsági jellemzőit egyaránt javítják. A módszerek konkrét implementációit a .NET keretrendszerre készítettem el, amelyek kis módosítással, de az általánosított koncepciókat megtartva ültethetők át más platformokra is.

Az eredményeimet mindig iteratív módszerekkel állítottam elő. Ez azt jelenti, hogy a kutatott területet több fázisban, a témában mindig mélyebbre merülve ismertem meg. Ez tette lehetővé egyre komolyabb eredmények elérését, újítások létrehozását.

Az első iterációban mindig arra törekedtem, hogy az adott területet a lehető legnagyobb mértékben megismerjem, megtaláljam azokat a pontokat, ahol eredményessé válhat a kutatás. Ez mindig olyan témaköröket jelent, amelyet mások még nem fedeztek fel, vagy bár több kutató foglalkozik a témával, de gondolkodásmódjuk illetve vizsgálati módszereik különböznek.

A következő iterációkban folyamatosan publikáltam eredményeimet. A dolgozatban ezeket az eredményeket foglaltam össze sokszor az iterációs szemléletet szem előtt tartva.

3 A dolgozat eredményei

3.1 Elosztott alkalmazások biztonsági és minőségi kérdései

Egy olyan módszert hoztam létre, amely más kutatók által készített munkák egy továbbfejlesztése, illetve a már meglévő, de eddig egymástól függetlenül kezelt biztonsági megszorítások felhasználását rendszerezi.

Bevált gyakorlat szerint a homlokzat (facade) tervezési minta segítségével egy interfészt adunk a külvilág számára, amely egy rendszer szolgáltatásait képes publikálni a külvilág számára.

Ezen szolgáltatások semmilyen megszorításokat nem definiálnak a szerződés szintjén, amelyet egy megoldandó problémának látok. Ezért dolgoztam ki egy olyan módszert, amelynek segítségével megszorításokkal láthatók el a kliens és a szerver kommunikációja során meghatározott szerződések.

Ezt a megoldást az üzleti szolgáltatások és a mögöttük üzemelő munkafolyamatok, a hívó felhasználóra vonatkozó dinamikus szabályok, a futás idejű hozzáférés-vezérlés kiterjesztése, hálózati korlátozások ötvözése jellemzi. Az eddig egymástól függetlenül, nehézkesen integrálható módon létező üzleti szolgáltatásokat, munkafolyamatokat, hozzáférés-vezérlést, szabály alapú jogosultságkezelést kötöttem össze. A klasszikus hozzáférés-vezérlés egy kiterjesztését definiáltam az elosztott alkalmazások kontextusába.

Ebből kifolyólag egy olyan formálisan is leírható, illetve definiálható üzleti szolgáltatásokra vonatkozó platform független megszorításkészletet, illetve ezeket ellenőrző mechanizmusokat alakítottam ki, amely gyakorlatban felmerült problémákra ad megoldást.

A témával kapcsolatos eredményeket az [1] [2] [3] alatt publikáltam.

1. Tézis. Megmutattam a jelenlegi elosztott alkalmazások esetében használatos hozzáférés-vezérlő mechanizmusokban található korlátokat. Egy olyan formális modellt definiáltam, amely megválaszolja az elosztott hozzáférés-vezérlés legfontosabb kérdéseit, összekapcsolja a szolgáltatások és munkafolyamatok jogosultságkezelését, kiterjeszti a szerepkör alapú valamint a hálózati szegmensekhez tartozó jogosultságkezelést, valamint absztraktságából adódóan garantálja a platform- és implementációfüggetlenséget. A formális modell alkalmazhatóságát több ipari esettanulmányon keresztül is validáltam.

3.2 A formális modell megvalósítása

A bemutatott formális modell haszontalan lenne akkor, ha az a gyakorlatban nem segítené a programozók munkáját. Ezért a következő iterációban egy konkrét implementációt mutattam be a formális modellre. Ismertettem az elérni kívánt célokat valamint azokat a programozási paradigmákat, amelyek elvei segítségemre lehetnek egy formális rendszer konkrét programozási rendszerre való transzformálása során.

A transzformáció egy projekció, átírási rendszer a formalizált világgal és a vele barátságban álló deklaratív paradigmát sugalló C# nyelvi attribútumok között.

A dolgozatban formalizált esettanulmányokat áttranszformáltam egy C# nyelvű megvalósítássá, amelyből egyértelműen látható a módszer működőképessége is. A teljes rendszer architektúráját vázoltam, majd meghatároztam, hogy melyik építőelem melyik komponensben kell, hogy helyet foglaljon. A megvalósítás során több, .NET technológiát felhasználtam, ezek közül a legfontosabb a kommunikációért felelős WCF (Windows Communication Foundation) valamint a munkafolyamatok kezelésért felelős WF (Workflow Foundation).

Az ismertetett megoldás egy korábbi változatát több állami és nagyvállalati ügyfélnek készített alkalmazásban felhasználtam.

A témával kapcsolatos eredményeket [3] alatt ismertettem.

2. Tézis. A formális modellt felhasználva egy keretrendszert terveztem, amely alapján működő implementációt készítettem szabványos C# nyelven a .NET platformra. Megmutattam egy átírási módszert, amellyel a formális modell által definiált megszorítások egyértelműen átírhatók C# nyelvre. A keretrendszer segítségével megvalósítottam a formálisan is definiált esettanulmányokat.

3.3 Futás idejű napló létrehozása

Megvizsgáltam a .NET-alapú programfejlesztés során felhasználható eszközöket. Ebből az elemzésből nyilvánvalóvá vált, hogy nem áll rendelkezésre egy részletes, nyelv független, futás idejű napló, amely akár a dinamikus programszeletelés bemeneteként is

felhasználható. Célul tűztem ki, hogy azonosítom egy dinamikus szeletelés esetében is helytálló futás idejű naplókészítő eljárással kapcsolatos követelményeket.

A megoldás megtervezése és kivitelezése során iteratív módszert választottam. Először a szekvencia pontok által határolt utasítások szintjén készítettem naplót. Miután ez a megoldás beváltotta a hozzá fűzött reményeket, a változó szintű napló létrehozására tértem át. Ebben az esetben már meghatározható az, hogy melyik utasításban milyen változók olvasására, illetve írására kerül sor a program futása során. A módszer abban is egyedülálló a többi megközelítéssel szemben, hogy non-intrusive, azaz nem igényli sem manuálisan, sem automatikus eszközzel a program eredeti forráskódjának módosítását ahhoz, hogy részletes naplót tudjon generálni a futó programok esetében. A naplókészítő metódusok a JIT-fordítás során kerülnek behelyezésre az naplózott alkalmazás IL kódjába.

A módszert több, különböző futási karakterisztikával rendelkező alkalmazás esetében is teszteltem.

A témával kapcsolatos kutatások eredményét [4] [5] [6] alatt ismertettem.

3. Tézis. Megmutattam egy részletes, futási idejű naplózó eljárás szükségességét a szabványos .NET platform felett. Definiáltam az eljárással kapcsolatos követelményeket, valamint egy olyan programozási nyelv független megvalósítást hoztam létre, amely nem igényli az eredeti forráskód módosítását. A megoldás párhuzamos környezetben is megfelelő minőséggel és teljesítménnyel üzemel.

4 További kutatási lehetőségek

Amikor egy eredmény napvilágot lát, akkor fontos az, hogy a meglévő megoldásokkal könnyen integrálható legyen, könnyen tovább lehessen fejleszteni, illetve későbbi megoldásokkal összekapcsolható, kiterjeszthető legyen. Nincs olyan kutatási eredmény, amelyet elsőre a tökéletesség jellemezne. Minden eredmény egy evolúción, fejlődésen megy keresztül, amely során vagy általánosabbá és általánosabbá válik, vagy pedig egyre specifikusabb lesz.

A két legfontosabb és legrészletesebben tárgyalt kutatási területemet kívánom a továbbiakban is vizsgálni. Ezek az elosztott .NET alkalmazások minőségének és biztonságának javítása, valamint a futás idejű naplókészítés a magasabb minőségű .NET-programok létrehozásához.

Ami az első és a második tézisben felvázolt megoldásból kitűnik az az, hogy két

szolgáltatás egymástól teljesen független, azaz az egyik szolgáltatás állapota alapján nem lehetséges megszorításokat tenni egy másik szolgáltatás metódusainak elérhetőségére, meghívhatóságára. Amennyiben több szolgáltatásra egy közös megszorításcsomagot akarunk értelmezni, akkor a kontextusfüggő, munkamenetfüggő állapotaikat egy közös tárban kell nyilvántartani.

Az elosztott alkalmazások terén elért eredményeimet fel kívánom használni az ipari alkalmazások tervezése, fejlesztése során. Az iparban felmerülő problémák specifikusabb irányokat jelölhetnek ki a további kutatási területek felé.

Az általam adott naplózó megoldás részletességéből adódóan komplexebb forrásnyelvi kifejezéseket tartalmazó .NET programok magas részletességű dinamikus szeletelését is lehetővé teszi.

Amennyiben a programunkat előfeltételekkel, utófeltételekkel, illetve invariánsokkal látjuk el, ily módon specifikációt adunk meg, akkor a specifikáció segítségével ellenőrizhetjük a program helyességét. A napló segítségével a program teljes lefutása után bejárt program-utat, illetve változó értékeket validáljuk a specifikáció segítségével.

A folyamatosan fejlődő .NET keretrendszer újabb és újabb szolgáltatások hozzáadását indukálja a naplózó megoldásomhoz. A modernebb keretrendszer szintű funkcionalitások kezelése mellett, a naplózás teljesítményének javítása is fontos terület lehet.

A dolgozatban említett modern tervezési paradigmák (pl. Domain Driven Design) filozófiájának illetve a napjainkban divatos generatív programozás integrációjának segítségével olyan módszerek felé mozdulok el a jövőben, amelyek magas minőségű, biztonságos és lazán csatolt elosztott alkalmazások hatékony létrehozását hivatottak támogatni.

Célom az, hogy a gyakorlatban felhasznált objektum orientált és komponens orientált módszerek mellett olyan paradigmák használatát segítsem elő az elosztott alkalmazások készítése során, amelyek bizonyos kontextusokban a meglévő objektum- és komponenselvű módszerekkel integrálva segítik a rendszerek tervezését és fejlesztését. Többek között ide tartozik a deklaratív, az aspektus orientált, a funkcionális illetve a generatív programozási paradigma.

Hivatkozások

1. Biczó, M., Pócza, K., Porkoláb, Z.: Runtime access control in C# 3.0 using extension methods. Annales Universitatis Scientiarum Budapestinensis de Rolando Eötvös Nominatae, Sectio Computatorica 30, 41-60 (2009)

2. Pócza, K., Biczó, M., Porkoláb, Z.: Runtime Access Control in C#. In : Proceedings of the 7th International Conference on Applied Informatics (ICAI), Eger (Hungary), pp.28-31 (2007)

3. Pócza, K., Biczó, M., Porkoláb, Z.: Securing Distributed.NET Applications Using Advanced Runtime Access Control. Frentiu et al ed.: Studia Universitatis Babes-Bolyai Informatica LIII(2008/2), 39-54 (2008)

4. Pócza, K., Biczó, M., Porkoláb, Z.: Cross-language Program Slicing in the.NET Framework. In : Conference proceedings of.NET Technologies 2005, Plzen (Czech Republic), pp.141-150 (2005)

5. Pócza, K., Biczó, M., Porkoláb, Z.: Towards Effective Runtime Trace Generation Techniques in the.NET Framework. In : Short communication papers proceedings of.NET Technologies 2006, Plzen (Czech Republic), pp.9-16 (2006)

6. Pócza, K., Biczó, M., Porkoláb, Z.: Towards detailed trace generation using the profiler in the.NET Framework. Annales Universitatis Scientiarum Budapestinensis de Rolando Eötvös Nominatae, Sectio Computatorica 30, 21-40 (2009)

7. Biczó, M., Pócza, K., Forgács, I., Porkoláb, Z.: A New Concept of Effective Regression Test Generation in a C++ Specific Environment. Acta Cybernetica 18(3), 481-501 (2008)

8. Biczó, M., Pócza, K., Porkoláb, Z.: A Cache-Based Interprocedural Static Slicing Algorithm. In : Proceedings of the 7th International Conference on Applied Informatics (ICAI), Eger (Hungary), pp.207-218 (2007)

9. Biczó, M., Pócza, K.: Generating Functional Implementations of Finite State Automata in C# 3.0. Electronic Notes in Theoretical Computer Science (ENTCS) 238(2), 3-12 (2009)

10. Pócza, K., Biczó, M., Porkoláb, Z.: docx2tex: Word 2007 to TeX. TUGBoat, TUG 2008 Conference Proceedings 29(3), 392-400 (2008)

11. Pócza, K., Pataki, N.: An Improvement on the Access Control Features of C#. In : Proceedings of Sixteenth Electrotechnical and Computer Science Conference (ERK 2007), Portoroz, vol. B, pp.33-41 (2007)

12. Pócza, K., Biczó, M., Porkoláb, Z.: FC#: Designing an Internal Functional DSL to C# 3.0. In : Proceedings of the Implementation and Application of Functional Languages 20th International Symposium, IFL 2008, Hatfield, Hertfordshire (UK), vol. Technical Report no. 474 (2008), pp.299-314 (2008)