Budapesti Műszaki és Gazdaságtudományi Egyetem
Villamosmérnöki és Informatikai Kar
Méréstechnika és Információs Rendszerek Tanszék

Effektív hozzáférési szabályok a kollaboratív modellezésben

Önálló laboratórium beszámoló

Készítette
Balogh Tímea

Konzulens
Debreceni Csaba

2017

TARTALOMJEGYZÉK

1. Bevezetés

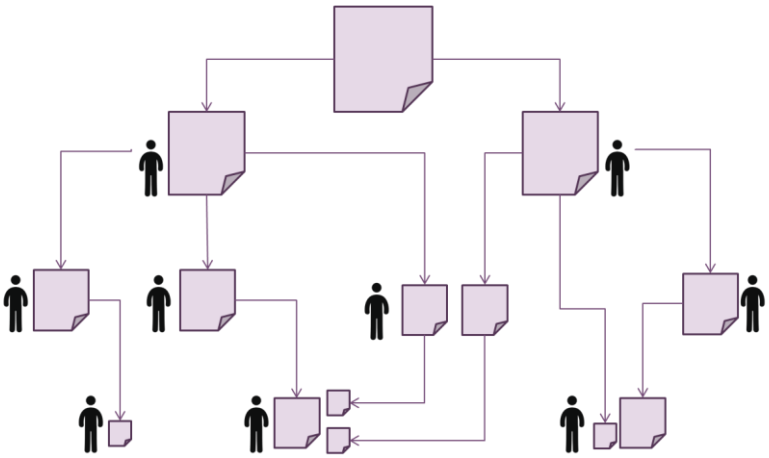
A szoftverfejlesztés terén egyre nagyobb szerepet kap a modellvezérelt tervezés. A módszer lényege, hogy a rendszert magas absztrakciós szintű modellekkel írjuk le, később ezeket folyamatosan finomítjuk, majd futtatható forráskódot generálunk belőlük, ezzel megkönnyítve és meggyorsítva a fejlesztés folyamatát. Modellek alkalmazásával komplex rendszerek is sokkal átláthatóbbá válhatnak.

Ezeken a projekteken általában egyszerre többen is dolgoznak, ami felveti a biztonság kérdéskörét is. A modellnek lehetnek olyan privát részei, amelyekhez csak bizonyos felhasználók férhetnek hozzá, kritikus pontjai, amelyeket csak a megfelelő szakértelemmel rendelkező fejlesztők módosíthatnak. A kollaborációban részt vevő fejlesztők, csapatok gyakran különböző szaktudással rendelkeznek, a rendszer más és más részeinek elkészüléséért felelősek, amihez elég, ha a modell releváns részéhez férnek csak hozzá. Azt, hogy a különböző felhasználók milyen modell elemeket írhatnak és olvashatnak, a hozzáférés-szabályozás mondja meg.

1.1. Hozzáférés-szabályozás lehetőségei

1.1.1. Fájl szintű hozzáférés-szabályozás

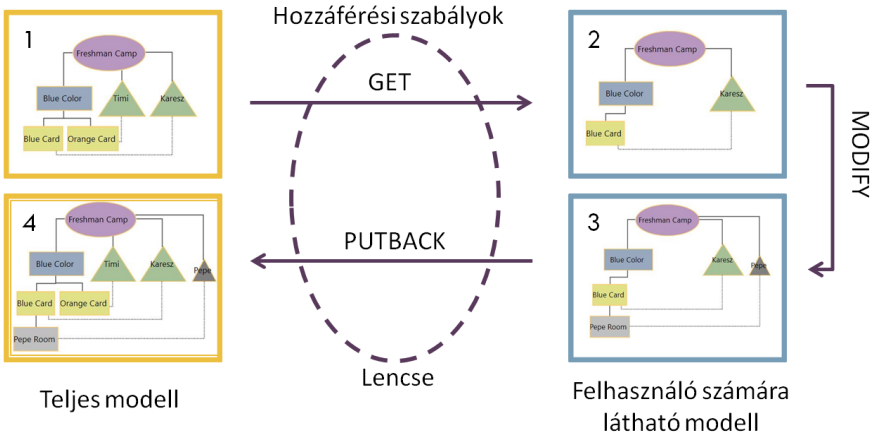
Fájl szintű hozzáférés szabályozás esetén, ha egy felhasználónak csak a modell egy bizonyos fragmenséhez szeretnénk jogosultságot biztosítani, akkor a modellnek ezt a részét le kell választanunk és külön kell eltárolnunk, a fájl tulajdonságai között pedig beállítani, hogy a felhasználó számára publikus legyen. Újabb felhasználók és szabályok hozzáadásával azonban egyre inkább felaprózódhat a modell, ahogy azt a lenti ábra szemlélteti. Összetettebb, több felhasználós rendszerek esetén ez megnehezíti a fejlesztés folyamatát.



1. ábra: Fájl szintű hozzáférés-szabályozás hátránya

1.1.2. Szabály alapú hozzáférés-szabályozás[1]

Az előbbinél hatékonyabb lehet egy szabály alapú hozzáférés-szabályozás, amely során modell szinten, szabályokban fogalmazhatjuk meg, hogy ki milyen olvasási/írási jogosultsággal rendelkezzen a különböző modell elemek felett. Ezen szabályok betartásáért egy ún. lencse felel, ez egy kétirányú modell transzformáció, aminek műveletei a GET és a PUTBACK. Előbbi minden felhasználó számára egy egyedi nézetet készít a modellből, amelyen csak azok a modell elemek (objektumok, attribútumok, referenciák) szerepelnek, amiket a szabályok szerint láthat. A módosítások a PUTBACK művelettel érvényesíthetők, ami előbb ellenőrzi az írási jogosultságokat, és amennyiben azok teljesülnek, végrehajtja a változást, ha nem, akkor visszautasítja a műveletet.



2. ábra: A szabályok betartásáért felelős lencse működése

[1] Gabor Bergmann, Csaba Debreceni, Istvan Rath, and Daniel Varro: Query-based Access Control for Secure Collaborative Modeling using Bidirectional Transformations, 2016.

1.2. MONDO Collaboration Framework

A MONDO kutatási projekt keretei közt készült egy kollaborációs keretrendszer, amely a másodikként említett szabály alapú hozzáférés-szabályozást alkalmazza. Ennek hiányossága, hogy nem veszi figyelembe az alapértelmezettként megadott hozzáférési szabályokat, valamint az olvasási és írási függőségeket. Az alapértelmezett szabályokat akkor vesszük figyelembe, ha egy modell elemre nincs definiálva egyéb specifikus szabály. Olvasási vagy írási függőségről pedig például akkor beszélünk, amikor egy modell elem látható/módosítható, de az őt tartalmazó elem nem, vagy amikor egy modell elem nem látható, de módosítható. A célom egy olyan, elméletben már létező algoritmus[2] implementálása, amelynek célja, hogy ezeket a konfliktusokat feloldja.

[2] Csaba Debreceni, Gábor Bergmann, István Ráth and Dániel Varró: Deriving Effective Permissions for Modeling Artifacts from Fine-grained Access Control Rules, 2016

2. Megismert technológiák

A következők közül az Eclipse Modeling Framework és Xtext már nem volt számomra teljesen ismeretlen, a témalaboratórium keretei közt foglalkoztam velük.

2.1. Eclipse Modeling Framework (EMF)

Az EMF egy Eclipse pluginekből álló modellező és kódgeneráló keretrendszer. Megkülönbözteti a metamodellt (Ecore modell) a tényleges modelltől, előbbi a modell struktúráját írja le, utóbbi pedig a metamodell konkrét példánya. Az Ecore modell gyakorlatilag osztályokat (EClass), valamint azok attribútumait (EAttribute) és a közöttük lévő referenciákat (EReference) tartalmaza. Az eszköz segítségével Java kódot is generálhatunk a modellünkhöz.

2.2. Xtext

A modellek leírásához modellezési nyelveket használunk. Ezeknek részei az absztrakt szintaxis és a konkrét szintaxisok. Előbbi azt határozza meg, hogy a nyelvnek milyen típusú elemei vannak és ezek milyen kapcsolatban állnak egymással, vagyis ez maga a metamodell. Ehhez több konkrét szintaxis is megadható, ezek olyan szöveges vagy grafikus megjelenítést biztosítanak a modellhez, amiktől olvashatóvá és szerkeszthetővé válik a modell leírása. Az Xtext keretrendszer segítségével szöveges konkrét szintaxis készíthető.

2.3. VIATRA Query

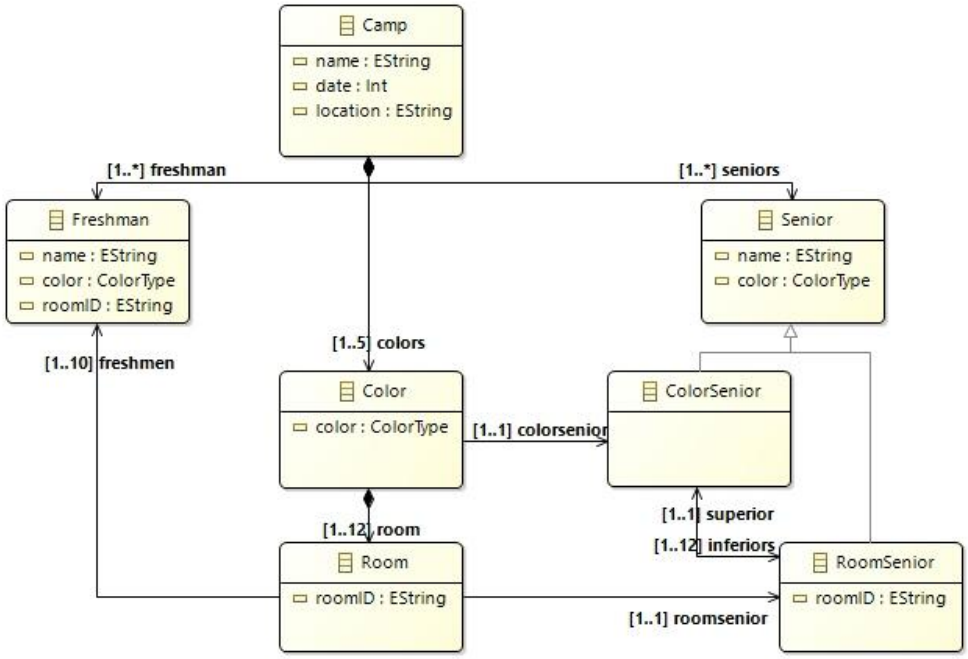
A VIATRA Query egy deklaratív lekérdezési nyelvvel rendelkező modelltranszformációs eszköz. A nyelv segítségével a lekérdezéshez gráfmintákat fogalmazhatunk meg a metamodell osztályaival, attribútumaival, referenciáival, a rendszer pedig azokat a modell elemeket adja vissza, amelyek illeszkednek a megadott mintáraa.

3. Motivációs példa

A félévben végzett munkám reprezentálására egy EMF-ben készített, gólyatábor témájú modellt használtam.

3.1. Metamodell

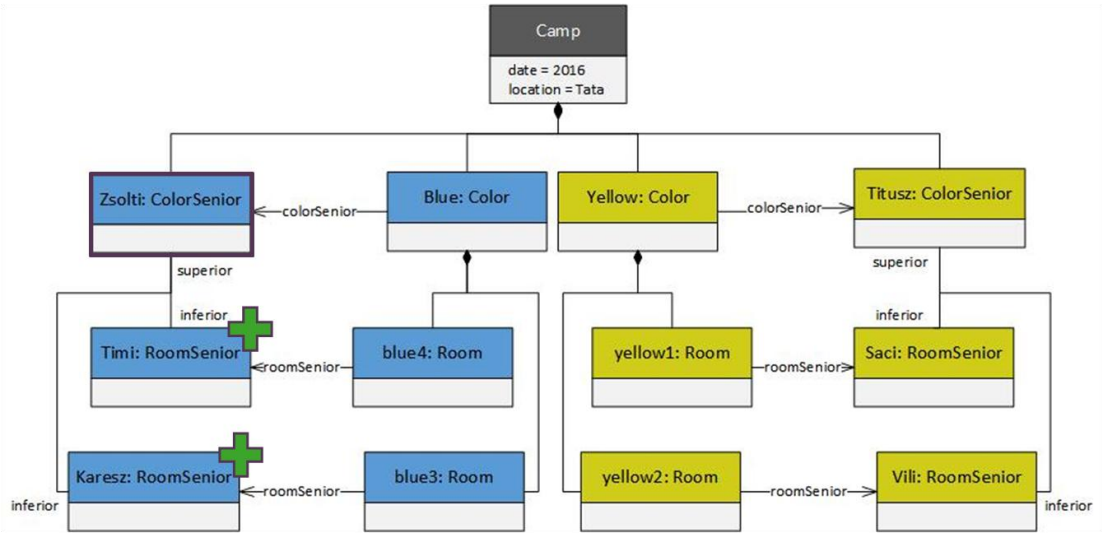
A modell gyökérosztálya maga a tábor, ami tartalmazza a gólya és a senior osztályokat, valamint egymásba ágyazva a „szín” és a „szoba” osztályokat, ezekbe a csoportokba tartoznak a résztvevők. Minden csoport tartalmaz referenciát egy-egy megfelelő típusú seniorra, aki azt irányítja. A seniorok között kétirányú referencia található, a „színseniorok” és a „kártyaseniorok” hierarchiájának ábrázolására.



3. ábra: A gólyatábor metamodellje

3.2. Példánymodell

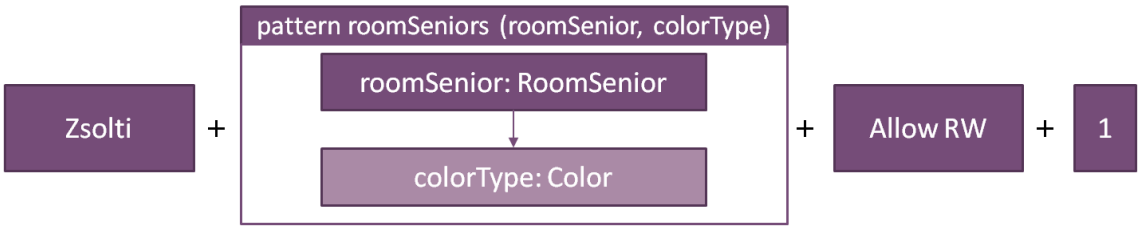
Az előbbi metamodellnek egy lehetséges példánymodelljében egy kék és egy sárga szín van, mindegyik két-két szobával, valamint a megfelelő típusú senior vezetővel. Ezen a modellen egy hozzáférési szabály lehet a következő: mivel a Zsolti nevű kék színsenior a saját színén belül a szobaseniorok főnöke, ezért azt szerettem volna meghatározni, hogy ő - mint felhasználó - ezekhez az objektumokhoz férjen hozzá és módosíthassa őket.



4. ábra: A gólyatábor példánymodellje az érvényesíteni kívánt szabállyal

4. Hozzáférési szabályok kiértékelése

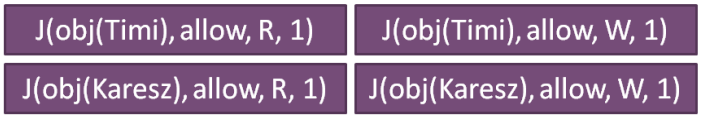
A hozzáférési szabályok több elemből állnak. A modell azon részeit, amelyekre a szabály vonatkozik, *gráf mintákkal* írjuk le. Ezek adják vissza az ún. asseteket, ezek lehetnek objektum (obj), attribútum (attr) vagy referencia (ref) modell elemek, ezekre adhatunk *írási vagy olvasási jogosultságok*at egyéni *felhasználók*nak vagy felhasználók egy csoportjának. Ezen kívül pedig *prioritás*t is rendelhetünk a szabályokhoz, ezzel egy fontossági sorrendet felállítva közöttük.



5. ábra: Egy szabály felépítése

Vagyis a már említett szabályhoz a következőket kell meghatározunk: A Zsolti nevű felhasználónak az adott színű szobasenior objektumokra írási és olvasási engedélyt adunk, 1-es prioritással. Ezzel a gráfmintával egyelőre csak az összes szobaseniort tudjuk lekérdezni a színükkel együtt, később ezt a kört szűkíteni kell még, hogy csak a megfelelő színűek legyenek benne.

A szabályok érvényesítéséhez azokból először ún. judgementeket definiálunk. Egy judgement egy bizonyos asset elérhetőségére vagy módosítására tartalmaz engedélyt vagy tiltást, valamint ennek a prioritását. A hozzáférés-szabályozás ezen judgementek kiértékelése, összevetése alapján dönti el, hogy mely szabályok érvényesülnek. A lenti képen az előbb példaként felhozott szabály alapján létrejövő judgementek láthatók: a Timi és a Karesz nevű objektumok írhatók és olvashatók.



6. ábra: A létrejövő judgementek

A már említett algoritmus - melynek célja a szabályok közötti ütközések feloldása - a megadott szabályok alapján definiált judgementek halmazát alakítja úgy, hogy a közöttük lévő konfliktusokat elemi konfliktusokra vezeti vissza, vagyis egy bizonyos assetre az egyik judgement engedélyez egy bizonyos műveletet, a másik tiltja azt. Ezek feloldásához először a judgementek prioritását veszi figyelembe, ha az megegyezik, akkor pedig az adott szabálynak az engedélyező vagy korlátozó tulajdonságát, amik az allow-t vagy a deny-t teszik dominánsabbá. Továbbá az olvasási/írási függőségek feloldására kibővíti az érvényre jutó judgementek listáját.

5. Nyelvtan

A szabályok definiálásához először egy Xtext nyelvtant készítettem.

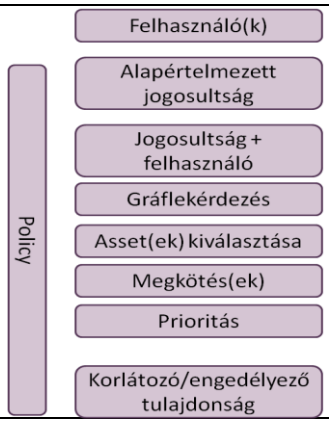
Az alábbi ábrán ennek a szöveges szerkesztője látható, amelyben az előbb példaként felhozott szabályt írtam le. Az elején adhatjuk meg a felhasználókat, vagy egy halmazukat, akikre majd a szabályok vonatkoznak. A szabályokat vagyis rule-okat policyba csoportosíthatjuk, erre megszabhatunk egy alapértelmezett jogosultságot (itt az írás/olvasás tiltott), valamint korlátozó vagy engedélyező tulajdonságot, ami megszabja, hogy az azonos prioritású szabályok között a tiltó vagy az engedélyező a dominánsabb. (Itt pl. mivel korlátozó tulajdonságot szabtunk meg, ha lenne még egy szabály, ami a meglévő engedélyezővel szemben a Zsolti nevű felhasználónak megtiltja a hozzáférést ezekhez az objektumokhoz, akkor az a szabály érvényesülne.) A rule elején megadjuk, hogy melyik felhasználóra vonatkozik, és milyen jogokat ad/tilt. (Itt engedélyezzük Zsoltinak az olvasást/írást.) A végén adhatjuk meg hozzá a már említett prioritást. A rule-on belül egy gráflekérdezésre hivatkozunk. (Az én példám szerint ez adja vissza a szobaseniorokat a színükkel együtt.) Ebből választhatunk ki objektum, attribútum vagy referencia asseteket (itt a szobasenior objektumokat), amelyek halmazát különböző megkötésekkel tovább specifikálhatjuk. (A szobaseniorok közül a kékek kellenek).

```
user Zsolti


policy camp deny RW by default {

	rule blueRoomSeniors allow RW to Zsolti {
		from query "roomSeniors"
		select obj(roomSenior)
		where colorType is bound to "BlueColor"
	} with 1 priority


} with restrictive resolution
```



A hozzáférési szabályok definiálását két extra funkció is kényelmesebbé teszi a felhasználó számára. Az egyik az automatikus formázás, a Ctrl+Shift+F billentyűkombinációt lenyomva az alább látható formára alakul a helyesen beírt szöveg. Ezen kívül szintén Java nyelven ScopeProvider-t is írtam, ami pedig Ctrl+space kombinációt a megfelelő helyeken lenyomva egy listából választhatunk az alternatívák közül. Konkrétan a felhasználók, a létező lekérdezések, és a lekérdezés által visszaadott assetek közül válogathatunk.

6. Összefoglalás

A félév során megismerkedtem a MONDO Collaboration Framework által is megvalósított szabály alapú hozzáférés-szabályozással, annak hiányosságaival, és azt ezt kiegészítő algoritmussal. Megtettem az algoritmus implementálásához szükséges első lépéseket: definiáltam egy Xtext nyelvtant, amellyel meghatározhatók a hozzáférési szabályok, ezt integráltam a meglévő rendszerbe a VIATRA Query technológiával frissítve. A későbbiek folyamán munkám középpontjába az algoritmus implementálását szeretném helyezni.