Teendők listája

- [ ] Azért azt erősnek érzem, hogy az SVN elavultnak számítana, már csak azért is, mert egyes felmérések szerint még mindig ez a legnépszerűbb verziókezelő rendszer. A "git alapú" megfogalmazás sem szerencsés, mert az alapokat nem a Git fektette le, csak az lett népszerű; nagy részben a GitHub miatt. Javaslom inkább a centrális és elosztott elvű verziókezelési fogalmakat használni, a konkrét termékek (Git, SVN) pedig inkább csak példák legyenek. 3
- [ ] Vannak azért raszteres bináris állományok is. Fontos lenne hangsúlyozni, hogy vektoros állományok esetén az állapot alapú verziókezelés elveszti a módosítás szemantikai információját, hogy pontosan milyen transzformációs művelet került végrehajtásra. Ez a szöveges vektoros állományokra is igaz, pl. SVG. Bináris állományok esetén pedig a tárolás sem túl hatékony természetesen. 3
- [ ] Rövid bevezető a verziókezelésről. 5
- [ ] Szerintem írható az, hogy a 3.x verzió támogatott, tesztelve a 3.6-os verzióval lett. 6
- [ ] Konkrétan .NET 4.0-ra épül a projekt, így az vagy újabb frissebb változat kell. Ez Windows alatt amúgy már az OS része. 6
- [ ] Bár végül nem a .NET Core változat lett felhasználva, de Mono révén így is használható Linux és MacOS alatt is. 6
- [ ] Új projekt létrehozásakor is jó lenne ezt támogatni. 7
- [ ] A ∼ a nem tördelt szóköz. 7
- [ ] Ez így nagyon körülményes, jobb lenne megnézni nem támogatható-e új projekt létrehozásakor is a verziókezelés. 9
- [ ] Angol szavakat érdemes tipográfiailag kiemelni, pl. dőlten szedni. 9

- [ ] Az ablak tartalmát kicsit rendezd. Ne legyenek fölöslegesen nagy térközök, a Commit és a Cancel gomb lehetne egy vonalban, ilyenek. 10
- [ ] Itt számít, hogy konkrétan Shapefile vagy csak annyi, hogy fájlba mentett? 10
- [ ] Ezt a megzavaró működést nehéz lenne megkerülni valamilyen módon? . . 10
- [ ] Ha a projekt fájl mellé kerülne mentésre a tartalom, akkor mindez nem lenne ilyen körülményes. 11
- [ ] Útvonalakat szedj monospace betűtípussal. 11
- [ ] A fejlesztői dokumentációba kerülhetne leírás a projekt fordításáról, mert nem volt triviális ugyebár. 12
- [ ] Ezt inkább úgy kellene megfogalmazni, hogy van egy backend (AEGIS), ami a verziókezelésért, az adatok és a transzformációk tárolásáért felel; valamint egy frontend (QGIS plugin), ami egyrészt a GUI-t nyújtja, másrészt felel az architekturális és adatreprezentációs különbségek áthidalásáért (pl. layerek). Továbbá a Python.NET nyújtja a .NET szoftverkönyvtár eljárásainak meghívási lehetőségét Python kódból.

  Ezt ennek megfelelően kellene végigvezetni a teljes dolgozaton. Valójában nem a technológiák közötti áthidalást kellett megoldani (arra ott volt a Python.NET), hanem reprezentációs különbségek voltak. 12

Figure: Komponens diagram 12

- [ ] Az SFA-ról kicsit írni kellene. Egy osztálydiagram is elférne, generálj egyet VS-ben vagy tudok küldeni. 13
- [ ] Ez nem volt megoldható végül a pontosság csökkentésével? Sokkal szebb megoldás lenne.

  Ha nem is megoldható, ez így akkor se legyen leírva, hanem az szerepeljen, hogy az AEGIS minden geometriához egyedi azonosítót rendel, amit felhasználtál, stb. 15
- [ ] A típus és metódusneveket is szedd mindenhol monospace betűtípussal. . . 17
- [ ] Ezt inkább úgy, hogy a szükséges műveleteket implementálni kellett itt és itt 17
- [ ] Valóban jobb lenne ezek helyére pseudokód vagy algoritmus. 20
- [ ] Erre azért van jól bevetett magyar szó: réteg 21
- [ ] signal -> szignál mindenhol 22

☐ Ez szerintem elhagyható. 25
☐ Amennyiben ez egyfajta projektszerkezet lenne, akkor inkább a fejezet elejére kerüljön. 26
☐ Ilyet le ne írj, hacsak nem kifejezetten kéred, hogy pontozzák le a dolgozatod. 27
☐ Ehhez igazából csak le kell származtatni a IRevisionControl interfészből egy megfelelő megvalósítást. A WCF-es projektekben van is erre konkrét példa, ld. RevisionControlService osztályt például. Szóval ilyen van már amúgy. 28
☐ Teljesen biztos vagy benne, hogy a nyelvi beállítás okozta a hibát? Mert ilyet csak akkor írj le, ha ez 100%. 29
☐ A definíciókat nem igazán ilyen célra szokás használni. Lehetne esetleg külön nomenklatúra (jelölés-/elnevezésjegyzék) függelék, de most egy egyszerű description lista is elég lesz szerintem. 29
☐ Ezt "hossz" helyett inkább komponensek számának vagy hasonlónak hívhatnánk, szerintem félrevezető ez a "hossz". 30

Eötvös Loránd Tudományegyetem

Informatikai Kar

Programozáselmélet és Szoftvertechnológiai

Tanszék

Téradatok verziókövetésének megvalósítása Quantum GIS-ben

Témavezető:

Cserép Máté

egyetemi tanársegéd

Szerző:

Dorogi Benjámin

programtervező informatikus BSc

Budapest, 2019

Tartalomjegyzék

1. Bevezetés **3**

2. Felhasználói dokumentáció **5**

2.1. A téradatok verziókezelésének problémája 5

2.2. A modul működése 5

2.3. A beépülő modul telepítése 6

2.3.1. Rendszerkövetelmények 6

2.3.2. Telepítés 6

2.4. A verziókezelő használata 7

2.4.1. A kezelőfelület 7

2.4.2. Új követett projekt létrehozása 8

2.4.3. Módosítások mentése a verziókezelőbe 9

2.4.4. Tetszőleges verzió betöltése 9

2.4.5. Verziókezelt projekt betöltése 10

2.5. A modul eltávolítása 11

3. Fejlesztői dokumentáció **12**

3.1. A feladat specifikálása 12

3.1.1. Technológiák összekötése 12

3.1.2. A geometriaadatok kezelése 13

3.1.3. A QGIS reprezentáció kezelése 14

3.2. C# folyamatok használata Python kódban, a Python.NET modul . . 15

3.2.1. A Python .NET használata 16

3.3. Geometriai adatok változásának kezelése 17

3.3.1. Az AEGIS WorkingCopy 17

3.3.2. Az AEGIS WorkingCopyWrapper 17

3.3.3. A transzformációk . . . 18
3.3.4. Az IGeometryTransformation interfész . . . 18
3.3.5. A transzformációk generálása . . . 19
3.4. A QgisWorkingCopy . . . 20
3.4.1. A hiányzó információk biztosítása . . . 20
3.4.2. A QGIS munkapéldány funkciói . . . 21
3.5. A QAEGIS osztály . . . 22
3.5.1. Signalok és feldolgozásuk . . . 22
3.5.2. A kezelőfelület és eseményei . . . 24
3.6. Adatátvitel Python és C# közt . . . 25
3.7. Egyéb segédeljárások . . . 25
3.8. A program szerkezete . . . 26
3.8.1. A nyers programkód . . . 26
3.8.2. A működéshez szükséges binárisok . . . 26
3.9. A működésben lévő beépülő modul könyvtárszerkezete . . . 27
3.10. Tesztelés . . . 27
3.11. Továbbfejlesztési lehetőségek . . . 28
3.11.1. Az AEGIS továbbfejlesztési lehetőségei . . . 28
3.11.2. A Python modul továbbfejlesztése . . . 28
3.12. Megjegyzések a fejlesztés közben felmerült problémákat illetően . . . 29
3.12.1. Külső könyvtárak használata . . . 29
3.12.2. A Python.NET fordítási nehézségei . . . 29
3.13. Fogalomtár . . . 29

4. Összegzés **31**

Irodalomjegyzék **32**

1. fejezet

Bevezetés

Az informatikai projekteknek a munkafolyamatok követése, a különböző állapotok megtekintésének és visszaállításának lehetősége kulcsfontosságú részét képezi az agilis projektmenedzsment, valamint a kooperatív és elosztott munkavégzés elősegítése érdekében. Napjainkban a szöveges alapú munkák – például szoftverkód, szöveges dokumentumok, táblázatok – esetében már számtalan eszköz áll rendelkezésünkre ezen funkcionalitás eléréséhez, a már elavultnak számító – de ennek ellenére még mindig használt – CVS és SVN szoftverektől, a jelenleg alapvetőnek számító git alapú rendszerekig.

Azért azt erősnek érzem, hogy az SVN elavultnak számítana, már csak azért is, mert egyes felmérések szerint még mindig ez a legnépszerűbb verziókezelő rendszer. A "git alapú" megfogalmazás sem szerencsés, mert az alapokat nem a Git fektette le, csak az lett népszerű; nagy részben a GitHub miatt. Javaslom inkább a centrális és elosztott elvű verziókezelési fogalmakat használni, a konkrét termékek (Git, SVN) pedig inkább csak példák legyenek.

Az említett verziókezelők közös tulajdonsága, hogy mind úgynevezett állapot alapú revíziókezelést valósítanak meg, azaz minden egyes módosításkor az állományok nyers változásait, sorok törlését, hozzáadását tárolják, és ezek alapján állítanak elő újabb vagy régebbi revíziókat.

Az állapot alapú verziókezelés egyik hiányossága, hogy az adatokat binárisan tároló folyamatok esetében – amik jellemzően valamilyen grafikus információt tárolnak vektorosan –, nagy tárigényű és jóval kevésbé hatékony.

Vannak azért raszteres bináris állományok is. Fontos lenne hangsúlyozni, hogy vektoros állományok esetén az állapot alapú verziókezelés elveszti a módosítás szemantikai információját, hogy pontosan milyen transzformációs művelet került végrehajtásra. Ez a szöveges vektoros állományokra is igaz, pl. SVG. Bináris állományok esetén pedig a tárolás sem túl hatékony természetesen.

Az ilyen téradatok esetében a művelet alapú revíziókezelés sokkal célravezetőbb, mivel ebben az esetben a változások (vagy más szóval *delták*) az adott objektumokon végzett műveletek, melyek újbóli végrehajtásával megkaphatjuk a frissített változatát a módosított adatnak, az invertált műveletek alkalmazásával pedig korábbi állapotokat érhetünk el. A téradatok hatékony verziókezelésére való igényre válaszként született az Eötvös Loránd Tudományegyetemen fejlesztett *AEGIS* geoinformatikai programcsomag [1, 2] revíziókezelő modulja [3, 4], ami interfészt biztosít az összes szükséges folyamathoz, viszont ezen szoftvernek még nem született valós, ipari környezetben is felhasználható implementációja.

Szakdolgozatom motivációja az előbbi hiány megszüntetése, a széleskörűen elterjedt és használt nyílt forráskódú *QGIS* térinformatikai programhoz [5] írt verziókezelő beépülő modul megvalósításával, mely a *Q-Aegis* nevet kapta.

A modul feladata a QGIS alapvető geometriai műveleteinek naplózása, tetszőleges verzióállapotok betöltése az AEGIS könyvtár funkcióinak felhasználásával

2. fejezet

Felhasználói dokumentáció

Rövid bevezető a verziókezelésről.

2.1. A téradatok verziókezelésének problémája

Téradatokkal – vagy bármilyen kép alapú adatokkal – egyre több szakterület dolgozik, és általánosan ezek a folyamatok kifejezetten magas számú felhasználók által végzett módosítással járnak. Az információk tárolása vagy binárisan, vagy valamilyen összetett geodéziai adatbázis állománnyal történik, ezek verziókövetésére pedig nem született még általánosan alkalmazható megoldás. A QGIS egy széles körben elterjedt, főleg térképészeti adatok létrehozását, módosítását és tárolását támogató szoftver, ami még nem rendelkezik verziókezelést megvalósító kiegészítéssel, a Q-Aegis modul ezen hiány betöltésére született.

2.2. A modul működése

A Q-Aegis plugin az AEGIS nevű téradatkezelő keretrendszer felhasználásával biztosítja a QGIS-ben létrehozott projektek műveletalapú verziókezelését. A verziókezelő és a szerkesztőprogram közti technikai különbségeket a nyílt forráskódú *Python.Net* [6] modul használatával hidalja át. A program követi a rétegek és a rajtuk lévő objektumok (*feature*ök) változásait, a geometriák alakulását reverzibilis transzformációkká alakítja és ezeket tárolja. A korábbi vagy újabb verziók betöltésekor így nem szükséges forrásfájlokat cserélni, a rétegek új állapotait beállítani,

mindez automatikusan megtörténik. Ezen kívül a megvalósítása miatt a Q-Aegis plugin lehetővé teszi, hogy a QGIS-ben végzett munka során csupán az eredetileg nem menthető memóriában tárolt rétegekkel dolgozzon a felhasználó.

2.3. A beépülő modul telepítése

2.3.1. Rendszerkövetelmények

Mivel a program egy beépülő modul (*plugin*) a QGIS térinformatikai programhoz, ezért használatához szükséges legalább a szoftver 3.6-os verzió jelenléte. Korábbi verziók esetén nem ismert viselkedések léphetnek fel, viszont a beépülő potenciálisan használható korábbi 3.x-es verziókkal is (nem tesztelt). Az AEGIS csomag használata miatt, amennyiben nincs még jelen a rendszerben, telepíteni kell a Microsoft .NET keretrendszert a legfrissebb verzióig . Az előbbi dependencia miatt a program csak Microsoft Windows operációs rendszer alatt használható.

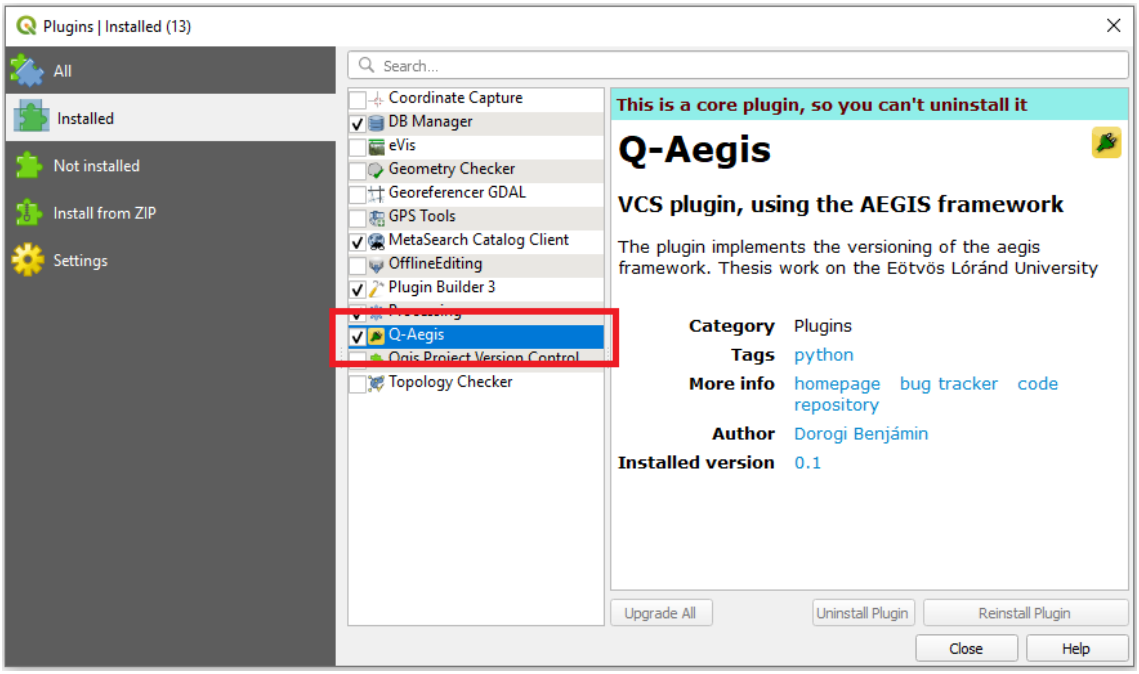
Szerintem írható az, hogy a 3.x verzió támogatott, tesztelve a 3.6-os verzióval lett.

Konkrétan .NET 4.0-ra épül a projekt, így az vagy újabb frissebb változat kell. Ez Windows alatt amúgy már az OS része.

Bár végül nem a .NET Core változat lett felhasználva, de Mono révén így is használható Linux és MacOS alatt is.

2.3.2. Telepítés

A beépülő telepítése kifejezetten egyszerű, csak ki kell csomagolni a mellékelt tömörített állomány tartalmát a helyi QGIS verzió `python/plugins` könyvtárába, amely telepítési könyvtár a `apps/qgis/python/plugins` útvonalán érhető el. Ezután a QGIS-t elindítva a *Plugins / Manage and install plugins* menüponton keresztül megnyitott plugin kezelőben be kell kapcsolni a Q-Aegis plugint, hogy az ábrán látható állapotot kapjuk :



2.1. ábra. Az aktiválás utáni állapot

Ezután a verziókezelő modul használatra kész.

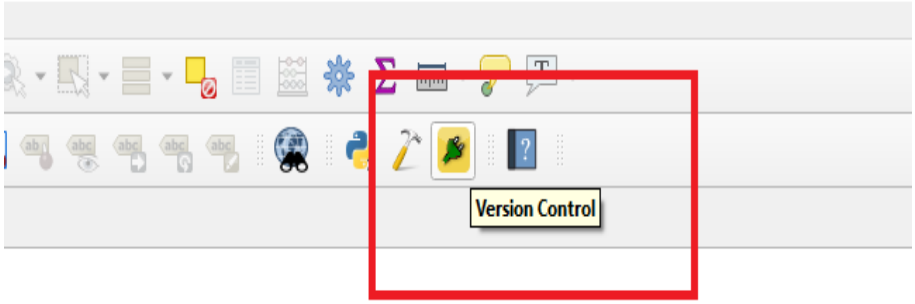
2.4. A verziókezelő használata

2.4.1. A kezelőfelület

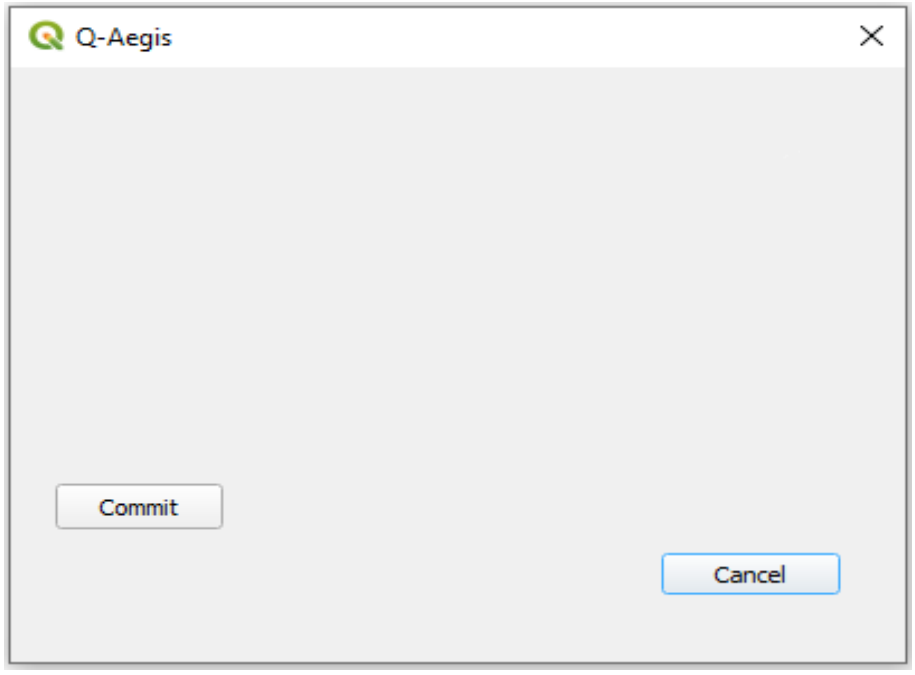
A beépülő modul a QGIS-en belül létrehozott projektekkel van összekötve, így amíg nem nyitunk meg egy projektet, nem érjük el az új funkciókat. Projekt megnyitása után a modul kezelőfelülete a QGIS beépülő modulok eszköztárán található ikonra (2.2. ábra) kattintva nyitható meg, és a 2.3. ábrán látható módon néz ki alapértelmezett állapotban.

Új projekt létrehozásakor is jó lenne ezt támogatni.

A ~ a nem törtdelt szóköz.



2.2. ábra. Az eszköztáron található ikon



2.3. ábra. A kezelőfelület alapállapota

Az ablak a *Cancel* bezárás gombbal zárható be, a többi gomb funkciójára, valamint a felület esetleges változásaira a használatot magyarázó részekben térünk ki.

2.4.2. Új követett projekt létrehozása

Amennyiben teljesen üres projekttel szeretnénk elkezdeni a munkát, technikai okokból a projektet létre kel hozni, elmenteni, majd újra megnyitni. Ekkor létrejön

Ez így nagyon körülményes, jobb lenne megnézni nem támogatható-e új projekt létrehozásakor is a verziókezelés.

a projekthez tartozó tároló, a verziókezelés használható innentől. Ha már meglévő projektet szeretnénk verziókezelés alá vonni, akkor a projektfájl megnyitása után a kezelőfelületen található *Commit* beküldés gombra kattintva a projekt teljes állapota bekerül a rendszerbe és a további módosítások már követhetőek lesznek.

2.4.3. Módosítások mentése a verziókezelőbe

Angol szavakat érdemes tipográfiailag kiemelni, pl. dőlten szedni.

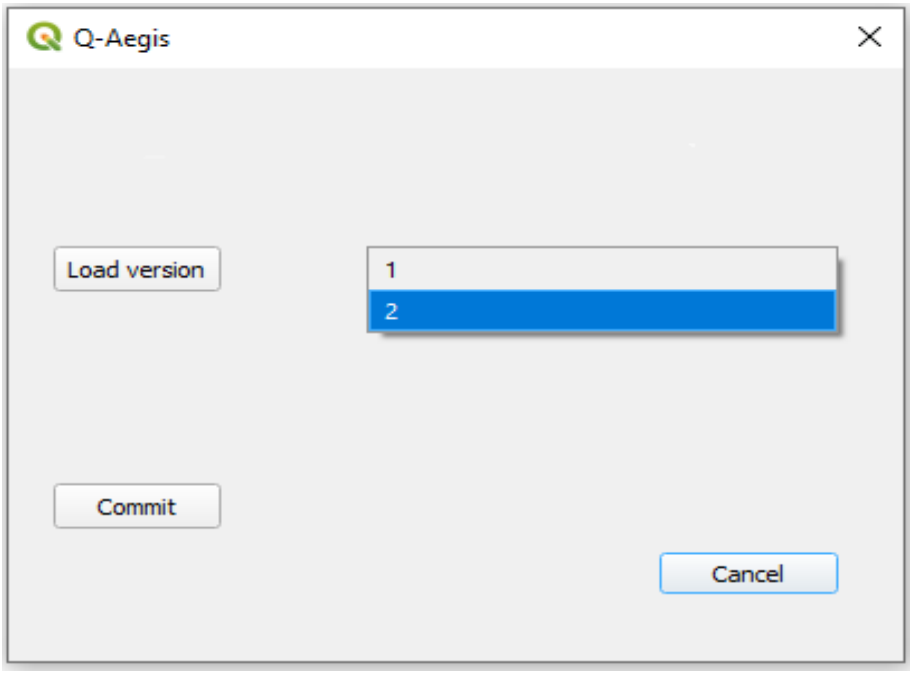
Miután módosításokat végeztünk a projekten, a *Commit* gomb újbóli megnyomásával az új állapot bekerül a revíziókezelő rendszerbe. A QGIS szerkesztési funkciói azok széles terjedelme miatt (egyelőre) nem teljes körűen támogatottak Q-Aegisben, hanem a következő alapvető módosítások kezeltek:

- Új réteg hozzáadása
- Létező réteg eltávolítása
- Geometria hozzáadása réteghez
- Geometria eltávolítása rétegről
- Geometria módosítása:
  - Tetszőleges geometria részének vagy egészének eltolása
  - Vonal típusú geometriákba új pont felvétele, pontok eltávolítása vonalakból
  - Poligon típusú geometriákhoz "lyukak" hozzáadása vagy eltávolítása
  - Multigeometriákhoz részek hozzáadása vagy eltávolítása

2.4.4. Tetszőleges verzió betöltése

Amennyiben már van a projekthez mentett verzió, a kezelőfelület némileg megváltozik, megjelenik rajta egy lista az elérhető verziókról (2.4). A listából egy tetszőleges számú verziót kiválasztva, majd a "Load Version" gombra kattintva a projektbe betöltődik a kiválasztott verzió. Amennyiben a betöltés előtt a projektben vannak olyan módosítások, amelyek még nem kerültek be a rendszerbe, a program egy felugró ablakkal figyelmezteti a felhasználót, és felajánlja neki a lehetőséget, hogy commitolja a módosításait, mielőtt betöltené az új verziót.

Ha korábbi állapot van betöltve, és módosításokat végzünk rajta, akkor ha menteni próbálunk, a rendszer hibaüzenettel jelzi, hogy előbb a legfrissebb verzióra kell állni és utána lehet módosítani.



2.4. ábra. A verzióválasztó lista

Az ablak tartalmát kicsit rendezd. Ne legyenek fölöslegesen nagy térközök, a Commit és a Cancel gomb lehetne egy vonalban, ilyenek.

Megjegyzés. *A QGIS több opcióval is rendelkezik arra, hogy milyen módon tároljuk a rétegeinket. A Q-Aegis csak a vektoros rétegtípusokon végzett műveleteket támogatja. Szerkesztés közben nem számít, hogy memóriában tárolt ideiglenes réteggel, vagy Shapefile-ban tárolt réteggel dolgozunk , viszont verzió betöltésekor az összes réteg törlődik, és a betöltendő verziónak megfelelő rétegek jönnek létre, amik mind memóriában tároltak. Mivel alapvetően ezeket a QGIS nem tárolja, kilépéskor figyelmezteti a felhasználót, hogy ezek tartalma el fog veszni, ez viszont nem érinti azokat a projekteket, amik a Q-Aegisben vannak, hiszen az abban tárolt állapotok alapján tölti vissza az adatokat.*

Itt számít, hogy konkrétan Shapefile vagy csak annyi, hogy fájlba mentett?

Ezt a megszokott működést nehéz lenne megkerülni valamilyen módon?

2.4.5. Verziókezelt projekt betöltése

A Q-Aegis jelenleg még nem ad lehetőséget arra, hogy létező tárolóból hozzunk létre egy új QGIS projektet, de az alábbi lépésekkel ez megvalósítható:

1. Hozzunk létre egy új projektet, mentsük el
2. Nyissuk meg, majd zárjuk be, így létrehozva egy új, üres tárolót
3. Lépjünk be a telepítésnél bemásolt könyvtárban található `repositories` mappába (`plugins/qaegis/repositories`)
4. A másolni kívánt projekt nevével megegyező nevű könyvtár tartalmát másoljuk ki
5. Szúrjuk be a kimásolt állományokat és alkönyvtárakat az új projekt nevével megegyező nevű mappába, felülírva tartalmát
6. Az új projektet megnyitva rendelkezésre áll a projekt másolata

Ha a projekt fájl mellé kerülne mentésre a tartalom, akkor mindez nem lenne ilyen körülményes.

2.5. A modul eltávolítása

A Q-Aegis modul kikapcsolható a QGIS beépülő modulokat kezelő felületén, ugyanúgy ahogy hozzáadásra került. A teljes eltávolításhoz töröljük ki a `qaegis` mappát a `plugins` könyvtárból.

Útvonalakat szedj monospace betűtípussal.

3. fejezet

Fejlesztői dokumentáció

A fejlesztői dokumentációba kerülhetne leírás a projekt fordításáról, mert nem volt triviális ugyebár.

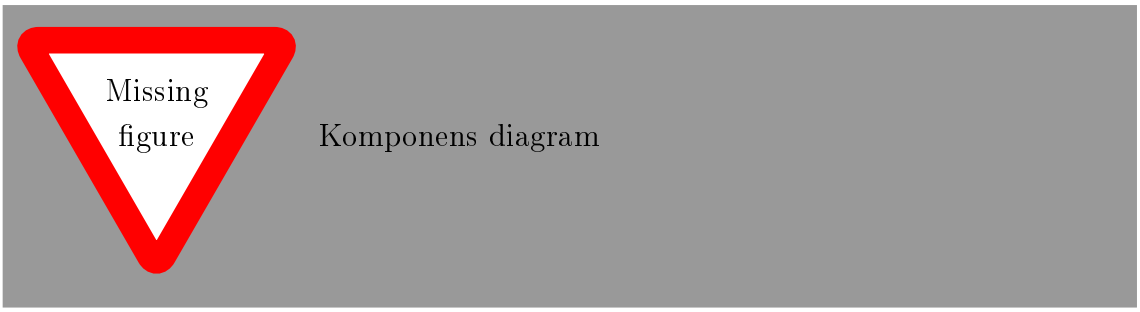
3.1. A feladat specifikálása

A beépülő modul célja a téradatok verziókezelésének megvalósítása a QGIS-en belül az AEGIS keretrendszer segítségével. Ez szétválasztható a QGIS és az AEGIS közti technológiai különbségek áthidalására, a tényleges geometriamódosítások kezelésére, valamint ezek összekötésére a QGIS-es reprezentációjukkal.

Ezt inkább úgy kellene megfogalmazni, hogy van egy backend (AEGIS), ami a verziókezelésért, az adatok és a transzformációk tárolásáért felel; valamint egy frontend (QGIS plugin), ami egyrészt a GUI-t nyújtja, másrészt felel az architekturális és adatreprezentációs különbségek áthidalásáért (pl. layerek). Továbbá a Python.NET nyújtja a .NET szoftverkönyvtár eljárásainak meghívási lehetőségét Python kódból.
Ezt ennek megfelelően kellene végigvezetni a teljes dolgozaton. Valójában nem a technológiák közötti áthidalást kellett megoldani (arra ott volt a Python.NET), hanem reprezentációs különbségek voltak.

3.1.1. Technológiák összekötése

A QGIS Python nyelvet használó interfészt biztosít az extra funkciók megvalósítására, az AEGIS keretrendszer pedig a C# alapú Microsoft.NET keretrendszerben íródott, ezen két technológiát kell valahogy egyesíteni.



Komponens diagram

3.1.2. A geometriaadatok kezelése

A geometriák tárolása, betöltése, valamint eltávolításuk kezelése mind az AEGIS létező funkciói voltak a projekt kezdeténél, így a tényleges feladat, a transzformációk definiálása, és létrehozása. Az átalakítások tervezése közben az a fő szempont, hogy a lehető legáltalánosabbak legyenek, és minden lehetséges módosítás leírható legyen véges alkalmazásukkal. Ezt a törekvést segíti az, hogy az AEGIS geometriatípusai megfelelnek a *Simple Feature Access* (röviden *SFA*) standardnak [7] , ezáltal egy adott hierarchiával rendelkeznek: A pontok különálló típust képeznek, A vonalak koordináták sorozatai, amik hasonlóan viselkednek a pontokhoz, a sokszögek pedig gyűrűkkel vannak leírva, amik a vonalak speciális változatai, így az "alacsonyabb szintű" geometriák transzformációi felhasználhatóak a bonyolultabb térobjektumok átalakításainak leírásához is. Az alakzatok lehetséges változásainak megfelelő rekreálásához a következő transzformációkat szükséges definiálni:

Az SFA-ról kicsit írni kellene. Egy osztálydiagram is elférne, generálj egyet VS-ben vagy tudok küldeni.

- Pont eltolása
- Vonal részleges vagy teljes eltolása
- Pont felvétele a vonalba
- Pont eltávolítása a vonalból
- Sokszög héjának módosítása
- Sokszögben lévő lyuk módosítása
- Lyuk felvétele sokszögbe
- Lyuk eltávolítása sokszögből

Ezeken kívül az összes többi alakzatot egybefogó, úgynevezett multigeometriák változásainak leírásához szükségesek a következő műveletek:

- Multigeometria egy vagy több részének módosítása a megfelelő egyedi geometria transzformációkkal
- Geometria hozzáadása multigeometriához
- Geometria eltávolítása multigeometriából

Ezen transzformációk segítségével bármilyen alapvető geometriaváltozást le lehet írni.

A transzformációk generálása komplex feladat, melynek során elemeznünk kell a kiindulási geometriát, és a végeredményt, ez alapján pedig létrehozni azt a transzformációgyűjteményt amit végrehajtva az eredetin, megkapjuk az újat. Vonalláncokra és poligonokra a következő algoritmussal adható meg:

1. Tekintsük a ez eredeti és a módosult geometria "hosszát" (ld. 3.13. szakasz)
2. A két geometria megegyező hosszú részein -ami megegyezik a "hosszaik" minimumával- hozzuk létre azokat a transzformációkat, amik az eredeti részeit az új részeknek megfelelő részekre alakítják
3. Amennyiben az új geometria hosszabb, a régiben még nem létező részeket adjuk hozzá
4. Amennyiben a régi geometria hosszabb, az újban már nem létező részeket töröljük ki

Mivel pont geometriáknál csak az eltolás művelete létezik, nem szükséges részekre bontani a transzformációgenerálást, a koordináták különbségeiből kiszámolható az összes szükséges paraméter.

Ezen részek implementálásával a geometriák alakulásának követése megvalósítható.

3.1.3. A QGIS reprezentáció kezelése

A fő különbség az AEGIS és a QGIS geometriákat kezelő része között az, hogy az előbbi csak a téradatot kezeli, az utóbbi viszont ezeket még rétegekre osztva

jeleníti meg. Ezzel önmagában nem lenne probléma, viszont a QGIS nem rendel egyedi azonosítót a geometriákhoz, így előfordulhat, hogy két különböző rétegen azonos geometria szerepel, és módosításukkor nem tudjuk megállapítani, hogy az AEGIS-ben tárolt adatok közül melyik módosult. Ez alapján a beépülő modulon belül megvalósítandó plusz funkciók a következőek:

- A QGIS objektumok (feature-ök) és az AEGIS geometriák kapcsolatának nyilvántartása
- A QGIS műveleteinek megfeleltetése az AEGIS műveleteknek
- Az AEGIS-ben tárolt állapotok megjelenítése
- A verziókezelés lépéseihez grafikus kezelőfelület biztosítása

Ezeken kívül sajnos láthatóvá kell tenni az AEGIS-hez képest külső modulnak számító plugin számára a geometriák belső azonosítóit, ami szembe megy a verziókezelő tervezési elképzeléseivel de sajnos nem elkerülhető. Ezen kívül valamilyen módon kompatibilissá kell tenni a QGIS geometriáit az AEGIS reprezentációikkal.

Ez nem volt megoldható végül a pontosság csökkentésével? Sokkal szebb megoldás lenne.
Ha nem is megoldható, ez így akkor se legyen leírva, hanem az szerepeljen, hogy az AEGIS minden geometriához egyedi azonosítót rendel, amit felhasználtál, stb.

3.2. C# folyamatok használata Python kódban, a Python.NET modul

Ahogy a specifikáció elején említésre került, a szoftver amihez a verziókezelő beépülő modul és a verziókezelést biztosító könyvtár két teljesen különböző technológiát használ. Mivel a Python programozási nyelv széleskörűen elterjedt különböző extra funkciók hozzáadására más környezetekhez, ezért több opció volt az eltérő megoldások összekötésére. Az egyik az IronPython nevű kiegészítő modul a Pythonhoz, viszon ez csak a nyelv 2.7-es verzióját támogatja, aminek hamarosan megszűnik a támogatottsága, így más utat kellett keresni. Az előbbihez hasonlóan nyílt forráskódú Python.NET modul azonos funkcionalitással rendelkezik, és a legújabb verziók-

kal is kompatibilis, így kézenfekvő megoldásnak bizonyult. A Python.NET projekt integrálása a QGIS pluginba egyedül azért okozott nehézséget, mivel a nyelvi összeköttetést biztosító szoftver fordításához számos dependenciának jelen kell lennie a rendszerben, így végül már fordított bináris formájában került a beépülő modulba. Ez sajnos kizárja annak lehetőségét, hogy a könyvtár esetleges frissülésével a plugin is dinamikusan megkapja a legújabb állományokat, viszont a program telepítését meglehetősen megkönnyíti.

3.2.1. A Python .NET használata

A Python.NET az úgynevezett *Common Language Runtime* (*CLR*) modult biztosítja a Python programozási nyelvhez. Amennyiben a fordított források a Python belső elérési útvonalain szerepelnek, importálható a CLR modul. Ezután a szintén classpathhoz adott dinamikusan linkelhető szoftverkönyvtárba (DLL) fordított C# könyvtárakat is hozzáadhatjuk a Python kódhoz. A tényleges implementációban a beágyazás az alábbi módon néz ki :

```
1 pluginpath = "{}/../python/plugins/qaegis".format(QgsApplication.
     pluginPath())
2 sys.path.insert(0,pluginpath+"/pythonnet")
3 sys.path.insert(0,pluginpath+"/aegis")
4
5 import clr
6 clr.AddReference('AEGIS.Core')
7 clr.AddReference('AEGIS.IO')
8 clr.AddReference('AEGIS.Versioning')
9 clr.AddReference('AEGIS.Processing')
```

3.1. forráskód. A CLR modul bekötése, és a dll-ek importálása a Q-aegis kódjában

Az importálás után pedig például ilyen módon hozhatunk létre egy String kulcsokkal egész számokat tároló C# szótárat:

```
1 dictionary =  Dictionary[String,int]()
```

3.2. forráskód. C# objektum használata python kódban

3.3. Geometriai adatok változásának kezelése

3.3.1. Az AEGIS WorkingCopy

Az AEGIS verziókezelő moduljának külső interfésze az úgynevezett Working Copy, avagy munkapéldány. Ezen kereszült a következő funkciók érhetőek el:

1. Teljes revíziókezelő, és hozzá tartozó tároló létrehozása vagy betöltése, attól függően, hogy van-e már jelen egy
2. Geometriák hozzáadása az aktuális verzióhoz
3. Geometriák eltávolítása az aktuális verzióból
4. Geometria módosítása az aktuális verzióban, az IGeometryTransformation interfészt megvalósító transzformációpéldánnyal
5. Aktuális verzió mentése a verziókezelőbe
6. Tetszőleges verzió betöltése és megjelenítése

A WorkingCopy két hiányossággal rendelkezik a beépülő modulban való használathoz. Transzformációk előállítását nem támogatja, valamint az aktuális állapotát kifelé csak geometriák halmazaként biztosítja, ami a QGIS réteges megjelenítésében problémákat okozhat.

3.3.2. Az AEGIS WorkingCopyWrapper

A munkapéldány említett hiányosságainak pótlására jött létre a `WorkingCopyWrapper` osztály , amely a WorkingCopyhoz képest az alábbi többletfunkciókkal bír:

A típus és metódusneveket is szedd mindenhol monospace betűtípussal.

1. Eredeti és módosult geometria alapján transzformációk generálása, majd ezzel a workingcopy módosító függvényének használata
2. Egy egyedi azonosítókkal ellátott geometrialista biztosítása a lokális állapotról

A rendeltetésszerű működéshez ez viszont nem elegendő, ugyanis a specifikációban felsorolt szükséges transzformációknak nem volt implementációja az AEGIS-ben, ebből kifolyólag generálásukra se volt lehetőség. Ezt az AEGIS Processing könyvtárának kiegészítésével orvosoltam.

Ezt inkább úgy, hogy a szükséges műveleteket implementálni

3.3.3. A transzformációk

A geometriák átalakító műveletek kialakítása az alábbi elven alapul:

- A pontok transzformációja leírható egy egyszerű vektor hozzáadással
- Az összes többi geometria kisebb részekből áll (ld. 3.13. szakasz)
- Egy geometria módosítása leírható a részeire vonatkozó kisebb transzformációkkal

Ez alapján három típusú transzformáció alakult ki: *i*) a rész módosítása, ami a módosítandó részek indexeit és a részeken elvégzendő résztranszformációit tartalmazza; *ii*) a rész hozzáadása amely az új rész geometriáját és a beszúrás indexét tartalmazza; *iii*) valamint a rész eltávolítása, amely az eltávolítandó rész indexét tárolja, valamint az eltávolított rész geometriáját, erre az invertálhatóság megvalósításához van szükség.

3.3.4. Az IGeometryTransformation interfész

Ahogy korábban is említésre került, az AEGIS a transzformációkat az IGeometryTransformation implementációjaként várja. Ezen interfész legfontosabb metódusai:

1. Execute(IGeometry geometry) : a transzformáció végrehajtása az IGeometry interfésznek megfelelő téradaton, a módosított alakzat visszaadása. A forrásgeometriának meg kell egyeznie a transzformáció bemeneti típusával.
2. Invert(): A transzformáció inverzének előállítása, mellyel helyreállítható az eredeti geometria. Hozzáadás típusú művelet inverze ugyanazon az indexen történő eltávolítás és fordítva, részmódosítás esetén a résztranszformációk inverzeinek végrehajtása ugyanazokon a részeken.

Az összes módosítást leíró osztály rendelkezik ezen metódusok egyedi implementációival, melyek megfelelnek a saját geometriatípusaiknak.

3.3.5. A transzformációk generálása

: Az eredeti és módosított geometriákból transzformációk generálását az AEGIS Processing modulban található TransformationFactory statikus osztályok végzik. A transzformációk létrehozása a specifikációban leírt algoritmus alapján működik, ez legkönnyebben a multipont transzformáció generálásának kódjával szemléltethető:

```
1  public static List<IGeometryTransformation<MultiPoint, MultiPoint>>
       Create(MultiPoint oldMultiPoint, MultiPoint newMultiPoint)
2  {
3  List<IGeometryTransformation<MultiPoint, MultiPoint>> result = new
       List<IGeometryTransformation<MultiPoint, MultiPoint>>();
4  var pointTranslates = new Dictionary<int,
       PointTranslateTransformation<Point>>();
5  for (int i = 0; i< Math.Min(oldMultiPoint.Count, newMultiPoint.
       Count);i++)
6  {
7  pointTranslates.Add(i, PointTranslateTransformation<Point>.Create(
       oldMultiPoint[i], newMultiPoint[i]));
8  }
9  result.Add(new MultiPointTranslateTransformation<MultiPoint>(
       pointTranslates));
10 if (oldMultiPoint.Count < newMultiPoint.Count)
11 {
12 for (int i = 0; i < newMultiPoint.Count; i++)
13 {
14 if (i > oldMultiPoint.Count -1)
15 {
16 result.Add(new MultiPointAddPointTransformation<MultiPoint>(
       newMultiPoint[i], i));
17 }
18 }
19 } else if (oldMultiPoint.Count > newMultiPoint.Count)
20 {
21 for(int i = 0; i< oldMultiPoint.Count; i++)
22 {
23 if (i > newMultiPoint.Count -1)
24 {
```

```
25 result.Add(new MultiPointRemovePointTransformation<MultiPoint>(
       oldMultiPoint[i], i));
26 }
27 }
28
29 }
30
31 return result;
32 }
```

3.3. forráskód. A MultiPointTransformationFactory Create metódusa

Valóban jobb lenne ezek helyére pseudokód vagy algoritmus.

Mint látható, a generáló függvény először létrehoz annyi résztranszformációt, amekkora részen még egyezik a két geometria hossza, majd a még nem, vagy már nem jelen lévő részgeometriák létrehozásának illetve eltávolításának műveleteit állítja elő. Az összes transzformációgenerátor hasonló módon működik.

3.4. A QgisWorkingCopy

3.4.1. A hiányzó információk biztosítása

Az AEGISben létrehozott kibővített munkapéldány önmagában még nem elég ahhoz hogy a QGIS projektek verziókezelésére alkalmas legyen, mivel nem képes a rétegek adatainak és változásainak követésére, valamint nincsen megoldása arra se, hogy a különböző featureökhöz rendelje a benne tárolt geometriákat. Ezeket a funkciókat a QgisWorkingCopy Python osztály biztosítja az úgynevezett QWCData állomány segítségével. Ebben egy extraData névre keresztelt objektum található amely az alábbi formátumot követi: Az egész objektum egy Python szótár, amelyben a kulcsok a verziószámok, a hozzájuk tartozó objektumok pedig a projekt adott verzióhoz tartozó állapotát írják le. Ebben az állapotleíróban a keyDict a featureök "Qgis id (ld. 3.13. szakasz)"-jához rendelve tárolja az Aegisben tárolt geometriák kulcsait, a layers pedig a rétegek betöltéséhez szükséges adatokat tartalmazza: az azonosítót, a geometriatípust és a referenciarendszert.

3.4.2. A QGIS munkapéldány funkciói

Extra adatok nyilvántartása

a QgisWorkingCopy tükrözi az AEGIS munkapéldány wrapper funkcionalitását, kiegészítve azokat az extra adatok karbantartásával. Példányosításakor ellenőrzi, hogy létezik-e a verziókezelő állományainak tárolására mappa, ha nem létrehozza, majd példányosítja a wrappert, szimpla fájl alapú tárolóval és kétirányú deltákat használó revízió kontrollerrel. Ezek után betölti az azonos mappában tárolt QWCData fájlból a szükséges további adatokat. A geometriák hozzáadása, módosítása és törlése egyszerűen a WorkingCopyWrapper azonos metódusait hívja meg, azonban a commit függvény hívásakor el kell tárolnia az extra adatokat. Ezen információk a Q-Aegis futása közben gyűlnek össze mely folyamat később lesz részletezve. A legtöbb adat kezelése nyílvánvaló, viszont a feature id-k karbantartása, valamint a verziók betöltése érdemel némi külön figyelmet.

A feature id probléma

A QGIS a rétegeken lévő objektumok azonosítóit a réteg tárolásától függően különböző módon kezeli (amire a dokumentációja nem sok említést tesz). Amennyiben memóriában tárolt a layer , a featureök indexelése 1-től kezdődik, és a műveletektől függetlenül nem változik. Shapefileban tárolt adatok esetén viszont 0-tól indul az indexelés, és objektum eltávolítása esetén az összes olyan azonosító, ami nagyobb volt nála, csökken eggyel, hogy egy folyamatos id állapot maradjon. Ezért amikor feature eltávolításokat kezelünk, és a réteg tárolója shapefile volt, minden törlés után az összes eltárolt nagyobb azonosítót dekrementálni kell eggyel.

Erre azért van jól bevetett magyar szó: réteg

Verzió betöltése, rétegek kezelése

Egy kiválasztott verzió betöltése a munkapéldányon keresztül történik, először a workingcopyt a megjeleníteni kívánt verzióra állítjuk, majd ebből lekérjük az aktuális geometria állapotot, amit a geometriák egyedi azonosítóiból és magukból a geometriákból álló párok formájában kapunk meg, C# szótár formájában. Ezt az adott verzióhoz tartozó extra adat objektum segítségével feldolgozzuk olyan módon, hogy rendelkezésünkre álljon az összes megjeleníteni kívánt réteg létrehozásához szüksé-

ges adat, valamint összepárosítjuk a keyDict segítségével a geometriákat a layereken kirajzolandó featureökkel. Ezt követően a rétegadatok alapján létrehozzuk a valódi QGIS layereket, feltöltjük őket az adott verzióban lévő állapotukkal, majd ezeket jelenítjük meg a QGIS grafikus felületén.

3.5. A QAEGIS osztály

A tényleges beépülő modul maga a QAEGIS Python osztály, melynek váza a QGIS-hez írt plugin builder beépülővel lett generálva. Ez biztosítja, hogy az alap szerkezete megfelelő legyen az alapprogramba való integráláshoz, és minden szükséges metaadattal rendelkezzen. Ebben az osztályban férünk hozzá a QGIS interfészéhez, amin belül az aktuális projekt példánnyal dolgozik a modul. Mivel a QGIS grafikus felülete a Qt szoftverkönyvtáron alapul, ezért az események kezelésére az úgynevezett *szignálokat* használjuk fel. A signalokat a felhasználó tevékenységei váltják ki, és egy adott információhalmazt adnak, amire rákötethetünk saját implementálású feldolgozófüggvényeket. A signalok és kezelésük áttekintésével könnyen megérthető a beépülő modul működése.

signal -> szignál mindenhol

3.5.1. Signalok és feldolgozásuk

A QGIS interfész projekt megnyitási signal

Ezen jelzést használja a modul a munkapéldány megnyitására, amely folyamat vagy létrehoz egy új üres revíziókezelőt, vagy ha létezik a megnyitott projekthez egy, betölti azt. Ezen signal kezelése során jönnek létre azok a listák is amik követik az aktuális szerkestésekben a rétegek és featureök létrehozását és törlését.

A projekt példány réteg signaljai

Az aktuálisan megnyitott projekt réteg létrehozásakor vagy törlésekor egy jelzés formájában ad információkat a végzett műveletről. Ezeket kezelve kerülnek a bufferlistákba az elvégzett műveletek. A bufferekre azért van szükség, mert a projekt mentéséig nem tekinthető véglegesnek a réteg jelenléte vagy eltávolítása, ezért amíg nem mentett állapotban vannak, nem kerülnek be a revíziókezelésbe. A réteg törléséhez tartozó signal kezelésekor nem csak a rétegadatok eltávolítása kerül a bufferbe,

hanem az összes rajta lévő objektum törlése is, így ha a layer már nincs jelen, a tartalma se marad felesleges adatként a rendszerben.

A projekt mentésének signalja

A projekt mentésekor tekinthető ténylegesen elvégzettnek egy layer törlése vagy felvétele, ezért ezen jelzés kiváltásakor kerülnek ezek a módosítások a munkapéldányba. Új rétegek esetén a rajtuk végzett geometriaműveletek mentésig nem relevánsak -hiszen ha elvetnénk a mentésüket, az összes rajtuk lévő objektum is törlésre kerülne-, így mentés után kerül be minden feature is új geometriaként az AEGIS rendszerbe.

Réteg mentésekor kiváltott jelzések

Fontos megjegyezni, hogy a rétegek rendelkeznek az egyes műveletek végrehajtásakor megjelenő signalokkal is, viszont ezek nem biztos hogy mentésre is kerülnek, így hasonlóan a rétegműveletek bufferéléséhez, ezek is egy köztes állapotban tárolódnak a véglegesítésükig. Szerencsére ezt a funkciót maga a QGIS biztosítja, így a függőben lévő műveletek tárolását nem szükséges újraimplementálni.A rétegen végzett műveletek mentésekor három különböző jelzés kerül feldolgozásra.

- **Hozzáadott featureök mentése**: Ezen signal szolgáltatja az adatokat az újonnan létrehozott geometriákról, így ezen keresztül kerülnek be az új téradatok a verziókezelésbe. Érdemes kiemelni itt az alábbi kódrészletet :

```
1  geomToAdd = feature.geometry()
2  for layer in self.iface.mapCanvas().layers():
3    if layer.id() == layerId:
4      layerType = layer.wkbType()
5  if layerType in range(4,7) :
6    geomToAdd.convertToMultiType()
```

3.4. forráskód. A multigeometria réteg probléma kezelése

Erre azért van szükség, mert amennyiben a réteg multigeometriákkal dolgozik, új geometria létrehozásakor -valamilyen ismeretlen okból- még szimpla téradatként kezeli azokat, viszont bármilyen módosítást végzünk rajtuk, átalakítja az egyszerű alakzatokat egyelemű többszörös változataikká. Ez komoly

problémákat tud okozni a módosítások kezelésénél, hiszen követetlenül változik meg a geometria típusa. Ezért ellenőrzi a program, hogy a réteg geometriatípusa többszörös-e, amit a 4,5,6 kódokkal jelöl, és ha igen, akkor még az új adat bekerülése előtt konvertálja azt multigeometria formába.

• **Eltávolított featureök mentése**: A geometriák eltávolításakor problémás, hogy az újonnan létrehozott adatokat mentésükig ideiglenes azonosítóval kezeli a QGIS, és mentéskor ezeket az ideiglenes változatokat törli, és felveszi véglekesként. Mivel az ideiglenes verzióik nem szerepelnek a verziókövetésben, így ha az eltávolításukat próbálnánk feldolgozni, hibára futnánk. Ezt kiküszöbölendő kihasználjuk az ideiglenes featureök egy tulajdonságát. Amíg a már véglegesített objektumok tárolás módjától függően 0-tól vagy 1-től kezdve kapnak azonosítót, ideiglenes verzióik a negatív irányba kapnak ID-t, ezért ha egy feature id-ja 0 vagy 1 alatti, ideiglenesnek tekinthető. Ennek felhasználásával megfelelő módon eltávolíthatóak a ténylegesen a felhasználó által törölt téradatok.

• **Módosított geometriák mentése**: A módosított geometriák kezelése elég egyértelmű mivel a valódi változáskezelést az AEGIS kiegészítései biztosítják, az egyedüli plusz ellenőrzés ami belekerült azt vizsgálja, hogy a módosított geometria mentetlen rétegen foglal-e helyet, és ha igen, nem tekintjük módosításnak, hiszen még nem került be a rendszerbe.

3.5.2. A kezelőfelület és eseményei

A kezelőfelület alapja a plugin generátor által létrehozott Qt ablak, amely Qt designer segítségével lett végleges formájára alakítva. A felület tervezésekor a fő szempont az volt hogy annyira egyszerű és intuitív legyen amennyire az csak lehetséges, így végül mindössze két elemmel vezérelhetővé vált a teljes verziókezelés

1. **A commit gomb**: Az aktuális verzió mentésére szolgáló gomb kettős funkciót tölt be. Az általánosabb, hogy ha már kezelt projekttel dolgozuk, megnyomásakor változtatásaink véglegesítésre kerülnek a revíziókezelő modulban, és létrejön egy újabb verzió. Másik funckióját abban az esetben látja el, ha egy létező, de még nem verziókezelt projekt van megnyitva. Ekkor megnyomásával

a projekt teljes szerkezete, azaz rétegei és azok tartalma bekerülnek a friss munkapéldányba mint kiindulási verzió.

2. **A verzióválasztó lista, és a hozzá tartozó betöltés gomb**: A verzióválasztó lista nem jelenik meg a kezelőfelületen addig, amíg a munkapéldány nem rendelkezikbetölthető verzióval. Amennyiben már használható, a kívánt verziószámot kiválasztva és a betöltőgombra kattintva elindul a kiválasztott revízió megjelenítése. Mivel a verziómegjelenítés implementációjából fakadóan elveti az összes aktuális módosítást amit esetlegesen még nem mentett a felhasználó, ezért amennyiben vannak ilyen függőben lévő műveletek, a program felajánlja azok mentését egy új verzióba a betöltés megkezdése előtt.

3.6. Adatátvitel Python és C# közt

A geometriák ugyan az AEGIS és a QGIS rendszerekben is szabványosan vannak kezelve, viszont a megvalósításaik különbözőek. Az ebből adódó problémák elkerülése érdekében a geometriák a legtöbb helyen a WKT reprezentációjukban, szövegesen kerülnek átadásra. Ezen kívül az AEGISben a geometriákhoz rendelt egyedi azonosító is használva van néhány helyen az objektumok kiválasztására. Erre azért van szükség, mert az AEGIS a geometriák közti ellenőrzésekor sokkal pontosabb adatokkal dolgozik mint a QGIS és ezek az eltérések sokszor nem várt viselkedéseket eredményeztek, például geometriák módosításakor a verziókezelő modul ellenőrzi, hogy van-e a módosított geometriával egyező a tárolóban, és ha nem talál ilyet, akkor szimplán felveszi újként, ami a pontatlanságok miatt lehetetlenné tette a módosítások hatékony követését, ezért módosításkor az eredeti állapotot közvetlenül kulcs alapján szedi ki a rendszer a tárolóból, ahelyett hogy a QGIS reprezentációval egyező geometriát keresne.

3.7. Egyéb segédeljárások

Ez szerintem elhagyható.

A szoftver rendelkezik néhány adatátalakító eljárással, ezek a következőek:

1. Aegis geometria reprezentációt QGIS reprezentációvá alakító függvény és inverze, a geometriák Well Known Text reprezentációját használják.

2. C# szótárat

3.8. A program szerkezete

Amennyiben ez egyfajta projektszerkezet lenne, akkor inkább a fejezet elejére kerüljön.

3.8.1. A nyers programkód

A létrehozott programkód szerkezete szétválasztható a C#-ban valamint a Pythonban írt részekre.
A C# nyelven implementált kód az alábbi struktúrát követi:

1. Az AEGIS egyes funckinalitásai a megfelelő könyvtárakban található állományokban vannak

2. A geometriatranszformációk, valamint a transzformációgenerátorok osztályai az AEGIS Processing könyvtárának QgisTransformation alkönyvtárában kerültek mentésre

3. A munkapéldány wrapper osztálya az AEGIS Versioning WorkingCopy könyvtárában lett létrehozva, az általa használt eredeti WorkingCopy mellett

A Python kód a qaegis mappában található, az alábbi szerkezetel:

1. A plugin magját adó QAEGIS osztály definícióját, valamint a QgisWorkingCopy osztályt a qaegis.py állomány tartalmazza

2. A szükséges konverziók implementációi a converters.py fájlban találhatóak

3. A kezelőfelület leírása a qaegis_dialog.ui állományba, a működtetéséhez szükséges kód az azonos nevű de Python kiterjesztésű fájlba került

3.8.2. A működéshez szükséges binárisok

A beépülő modul működéséhez szükséges az AEGIS csomag dllekké fordított verziója a plugin könyvtáron belüli aegis mappában kapott helyet, a lefordított Python.NET CLR modul pedig az azonos szülőmappájú `pythonnet` könyvtárban található.

3.9. A működésben lévő beépülő modul könyvtárszerkezete

A használatban lévő plugin a QGIS Python beépülőket tartalmazó mappájában található, ebben a könyvtárban van minden szükséges állomány. A tényleges verziókezelést biztosító tárolók a repositories alkönyvtárban kerülnek létrehozásra, ezen belül pedig a kezelt projekt nevével megegyező mappákban találhatóak a szükséges fájlok. Ezen belül a changesets mappa verziószámokkal elnevezett forrásaiban találhatóak a revíziókhoz tartozó változáslisták, a snapshots mappa tartalmazza az esetlegesen mentett verzióállapotokat, a revisioncontrol fájl pedig az összes többi használt adatnak ad helyet. A QWCData állomány tartalma az összes AEGIS-en kívüli követendő adat szerializált formája.

3.10. Tesztelés

Ilyet le ne írj, hacsak nem kifejezetten kéred, hogy pontozzák le a dolgozatod.

Sajnos a beépülő modul tesztelése korántsem volt alapos , csupán felhasználói tesztek lettek végrehajtva, azaz a program működése a használatával lett ellenőrizve. Ezek alapján az alábbi eredmények születtek.

1. A modul hozzáadása, betöltése hiba nélkül működik
2. A projektek verziókezelésének inicializálása üres, és meglévő munka esetén is végrehajtható
3. A rétegek létrehozása, törlése verziókezelt
4. Tetszőleges korábbi vagy új verzió betölthető, megfelelő rétegállapotok jönnek létre
5. A geometriák létrehozása és törlése működik
6. A geometriák módosításának megfelelő transzformációk kerülnek létrehozásra és tárolásra

3.11. Továbbfejlesztési lehetőségek

3.11.1. Az AEGIS továbbfejlesztési lehetőségei

Az AEGIS szerkezetéből adódóan számos lehetőséget biztosít további funkciók létrehozására:

1. A geometriákhoz tartozó egyéb metaadatok módosítására létre lehetne még hozni új transzformációs osztályokat, ezáltal ezek is kezelhetőek lennének

2. A repository osztályt ki lehetne bővíteni olyan módon, hogy ne nyers fájlokban tárolja az adatait, hanem valamilyen hálózati úton, így eszközök közötti kommunikáció is megvalósítható lenne.

   Ehhez igazából csak le kell származtatni a IRevisionControl interfészből egy megfelelő megvalósítást. A WCF-es projektekben van is erre konkrét példa, ld. RevisionControlService osztályt például. Szóval ilyen van már amúgy.

3. A módosítások kezelését tovább lehetne fejleszteni olyan adatok hozzáadásával, mint a szerkesző felhasználó azonosítója, ezzel alkalmasabbá téve a verziózást több ember által végzett projektek esetén is.

4. Esetlegesen a rétegrendszer implementálható lenne az AEGIS-en belül, mivel a legtöbb vektorgrafikus szoftver hasonló elven működik, így nem az alkalmazás részben kéne ezeket az információkat kezelni.

3.11.2. A Python modul továbbfejlesztése

1. A QAEGIS és QgisWorkingCopy osztályok erőteljesen az AEGIS keretrendszerre támaszkodnak, ezért annak bármilyen kiegészítése implementálható lenne ezen beépülő modulban való alkalmazásra is.

2. A kezelőfelület bővíthető lenne egy, a verziókezelők exportálására és importálására alkalmas funkcióval, így nem kéne az állományok másolását alkalmazni projekt betöltésére.

3. Mivel a QGIS a geometriák megjelenítésének vizuális részleteit, például a színüket, a rétegekhez kötve szabályozza, a rétegek stílusának követésével pontosabb állapotbetöltésre lenne lehetőség

3.12. Megjegyzések a fejlesztés közben felmerült problémákat illetően

3.12.1. Külső könyvtárak használata

Számomra az egyik legnagyobb nehézséget az jelentette, hogy a jellegéből adódóan az általam írt alkalmazásnak idomulnia kellett mind a QGIS, az AEGIS és a Python.NET sajátosságaihoz is. Ez a felhasznált algoritmusok tervezésekor sokszor nehézséget jelentett. Ezen kívül problémás volt a QGIS API dokumentációja is, sok technikai részletről nem tesz említést, így az ezek által okozott problémák csak tesztelés során váltak láthatóvá, valamint sok helyen nem pontosan úgy viselkedtek a QGIS funkciói, ahogyan azok dokumentálva voltak.

3.12.2. A Python.NET fordítási nehézségei

Teljesen biztos vagy benne, hogy a nyelvi beállítás okozta a hibát? Mert ilyet csak akkor írj le, ha ez 100%.

A Python.NET CLR moduljának használatához le kellett azt fordítani, ami sokkal nehezebb feladatnak bizonyult mint elsőre számítottam volna.A fordítója számos dependenciával dolgozik, és ezek nem várt hibákkal rendelkeztek. A legmeglepőbb talán az volt, amikor kiderült, hogy a modul nem fordul le olyan operációs rendszeren aminek a rendszernyelve nem angol. Mivel a Windows csak a megjelenítési nyelv változtatását támogatja, a fejlesztés érdekében az egész operációs rendszeremet le kellett cserélni egy angol nyelvű verzióra.

3.13. Fogalomtár

A definíciókat nem igazán ilyen célra szokás használni. Lehetne esetleg külön nomenklatúra (jelölés-/elnevezésjegyzék) függelék, de most egy egyszerű description lista is elég lesz szerintem.

Feature : A QGIS geometriai egysége, a layeren tárolt geometria és az összes hozzá tartozó egyéb adat.

Geometriatípus : A szabványos geometrialeírás típusainak egyike, lehetséges értékei Point, MultiPoint, LineString, MultiLineString,Polygon,MultiPolygon

Geometria "hossza" : A geometria közvetlen részgeometriáinak száma. Vonal esetén a pontjainak száma, sokszög esetén 1 (azaz a héj számossága) + a hozzáadott lyukak száma stb.

Ezt "hossz" helyett inkább komponensek számának vagy hasonlónak hívhatnánk, szerintem félrevezető ez a "hossz".

Geometria részgeometriái : Pont geometriának nincs részgeometriája. Vonal geometria részei a pontok amelyek összekötésével megkapjuk a vonalat. Sokszög esetén az alakzat külső héja, valamint a síkidomon lévő lyukak, ezek vonalakkal írhatóak le. Multigeometriák részgeometriái az egyes különálló részeik.

Layer :A QGIS megjelenítési rétege, kötött geometriatípussal, projekten belül tetszőleges darabszámú létezhet

MultiGeometria : Olyan geometria, amely több különálló geometriát tartalmaz, de egy egységként van kezelve

Project : A QGIS munkapéldánya, rétegeket lehet hozzáadni, és azon geometriákkal lehet műveleteket végezni

WKB,WKT : Well Known Binary, Well Known Text, geometriák leírására használt szabványos bináris és szövegformátum

Working Copy : A munkapéldány a verziókezelésen belül, egy adott verzió állapotát tartalmazza, a legfrissebb verzión szerkeszthető

Qgis id : A featureök egyedi beazonosítására használt karakterlánc, a réteg azonosítójából és a feature id-jából áll elő

Szótár adatszerkezet : Kulcs-érték párokat tartalmazó lista. C#-ban kötött a kulcsok és értékek típusa, Pythonban akár elementként eltérhet.

4. fejezet

Összegzés

Meglátásom szerint a kitűzött feladatot sikerült megvalósítani. Az alkalmazás képes a QGIS projektek geometriai részének majdnem teljeskörű verziózására, a beépülő modul könnyen telepíthető és használható. Az AEGIS keretrendszer használata számos továbbfejlesztési lehetőséget biztosít, de a pythonban megírt részek is bővíthetőek a nagyobb funkcionalitás elérésének érdekében.

Irodalomjegyzék

[1] Roberto Giachetta. "AEGIS - A state-of-the art component based spatio-temporal framework for education and research". In: *Free and Open Source Software for Geospatial (FOSS4G) Conference Proceedings*. Vol. 13. 2013.

[2] *AEGIS geospatial framework - Origin edition*. `https://github.com/robertogiachetta/aegis-origin`. Utolsó elérés: 2019-05-17.

[3] Máté Cserép. "Téradatok verziókezelésének vizsgálata". MSc diplomamunka. Eötvös Loránd Tudományegyetem, Informatikai Kar, May 2013.

[4] Máté Cserép and Roberto Giachetta. "Operation-based revision control for geospatial data sets". In: *Geomatics Workbooks* 12 (July 2015), pp. 139–152.

[5] *QGIS - A Free and Open Source Geographic Information System*. `https://www.qgis.org/`. Utolsó elérés: 2019-05-17.

[6] *Python for .NET*. `https://github.com/pythonnet/pythonnet`. Utolsó elérés: 2019-05-17.

[7] John R. Herring. *OpenGIS Implementation Standard for Geographic information - Simple Feature Access - Part 1: Common architecture*. Tech. rep. Reference Number: OGC 06-103r4. Open Geospatial Consortium, May 2011.