

ETO jelzetszerkesztési szabályok algoritmizálása

2012.12.30.

Tartalomjegyzék

1.	Bevezetés	3
2.	A szoftver felépítése	4
3.	Adatbázis	5
4.	A szoftver működés közben	6
4.1.	Főosztály kiválasztása.....	7
4.2.	Főtáblázati szám kiválasztása (böngészés vagy keresés).....	7
	Keresés.....	8
	Böngészés	10
4.3.	Alosztályok kiválasztása (böngészés vagy keresés)	10
4.4.	Jelzet szerkesztése	11
5.	A program alkalmazási területei.....	12
6.	Hasonló próbálkozások	13
	AUDACIOUS - Automatic Direct Access to Information with the Online UDC system	13
	UDC summary online	13
7.	Automatizálás kérdése	14
8.	Összefoglalás	15
9.	Irodalomjegyzék	16
10.	Ábra és képjegyzék.....	16

1. Bevezetés

Dolgozatomban azt a programot szeretném bemutatni, ami köré egész kutatásom felépült. Az alap programot egy hallgatótársamtól készen kaptam: képességeiben azonban jóval szerényebb volt, viszont sok lehetőséget rejtegetett magában, melynek kiaknázására jelenleg is törekszem. Néhány társammal a programot jelenleg is fejlesztjük.

Az ETO jelzetszerkesztő program fejlesztése során nem kerültem meg a felhasználói döntést, ugyanis, ha elfordultam volna az automatizálás irányába, akkor annak problémái meghaladták volna munkám tartalmi határait, valamint olyan távlatokba vezetett volna el, ahonnan már nem láthattam volna a fő problémákat.

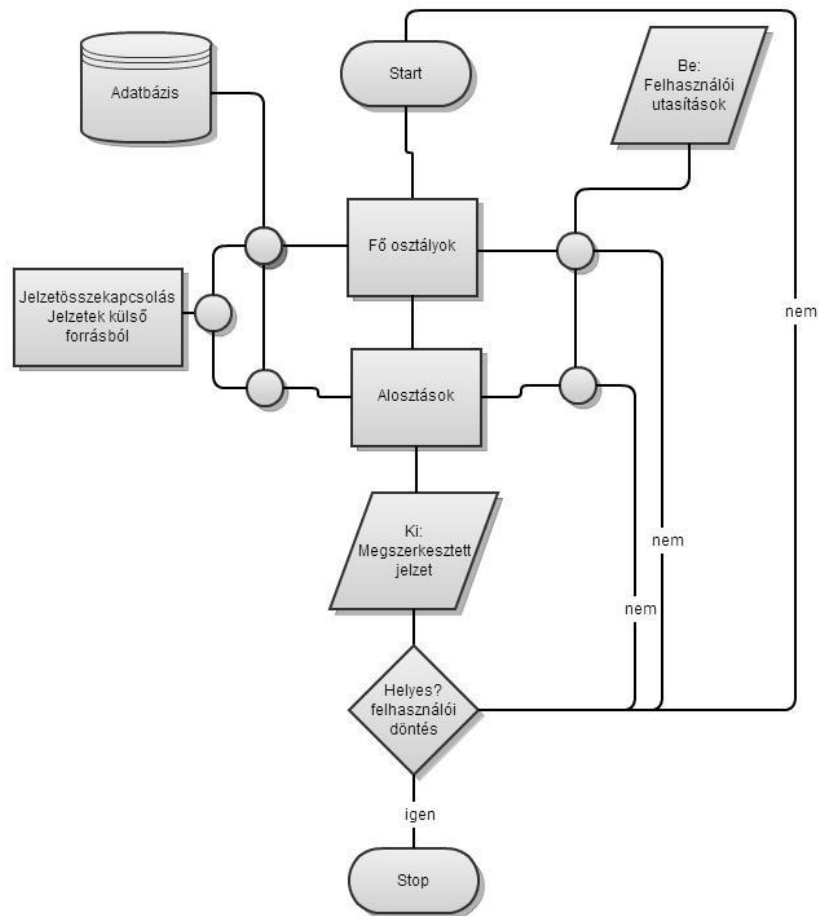
Viszont a tárgyalás során foglalkozom az automatizálás lehetőségeivel is. Nem lehet ezt a problémát, irányt szó nélkül hagyni, hiszen a mai tendenciákat tekintve az informatika terén a véső cél ennek a megvalósítása lenne.

Említést teszek még az első számítógépes és automatizálási próbálkozásokról, valamint a UDC kezdeményezéséről is.

Munkám célja az ETO szabályainak leírása a számítástechnika logikájával és ezek segítségével egy jelzetszerkesztő program fejlesztése, ezáltal megnyitva a kaput a további osztályozásra támaszkodó fejlesztéseknek. Remélem, hogy ez felkelti az érdeklődést a terület iránt, ezáltal modernizálhatjuk és felgyorsíthatjuk a dokumentumok tartalmi feltárását.

2. A szoftver felépítése

A program két részből áll: a szoftverből, mely JAVA nyelven íródott és az adatbázisból. Az adatbázis SQL típusú, annak is a „lite” verziója, amely majd megkönnyítheti a mobil applikációk fejlesztését. Az alkalmazás az adatbázisból nyeri ki a megfelelő jelzeteket.



1. ábra Általános folyamatábra

A fenti folyamatábra a legáltalánosabb. A főosztály és alosztályok sorrendje ebben az esetben nem rögzített, csupán a legideálisabb formát hivatott tükrözni. Az ábrán látható az adatbázis, mely arra hivatott utalni, hogy a jelzeteket egy külső, előre definiált és rögzített tárolóból ajánlott meghívni, ezáltal egyértelműen elkülöníthető a felhasználói döntésektől, utasításoktól.

Jól láthatóan a folyamat során két alapvető dologra hagyatkozunk a felhasználóra, valamint a jelzetekre az adatbázisból. A szerkesztés során végighaladunk az adott szinteken (elkészítjük a jelzetet). Kiválasztjuk a megfelelő főosztály és alosztásokat. A jelzetekhez kapcsolódhatnak névalosztások és össze is kapcsolhatjuk őket, külső forrásból is emelhetünk be jelzetet. A felhasználó eldönti, hogy számára megfelelő-e a kész jelzet? Ha igen, akkor kész vagyunk, ha nem akkor bármely szinten elvégezhetjük az újraserkesztést, javítást.

3. Adatbázis

rowid	jelzet	jelentes
1	1	FILOZÓFIA. PSZICHOLÓGIA
2	101	A filozófia lényege és feladatai
3	101.1	A filozófia jellege. A filozófia mint tudomány. A filozó
4	101.2	A filozófia lehetséges volta. Lehetséges-e filozofálni?
5	101.3	A filozófia tere és korlátai
6	101.8	A filozofálás módszerei
7	101.9	A filozófus személye és hivatása
8	11	A lét filozófiai problémái. Metafizika
9	111	Általános metafizika. Ontológia
10	111.1	A lét fogalma. Lételmélet. A primer szubsztancia. fog
11	111.11	Egzisztencia. Létezés. Valóság. Különbség az esszenci
12	111.12	Esszencia. Lényeg
13	111.125	Lehetőség. Belső lehetőség. Isteni lényeg és lehetőség
14	111.3	Szubsztancia. Magánvaló (monász). Egyediség
15	111.31	A szubsztancia és az akcidencia közötti különbség. A
16	111.32	A primer szubsztancia, az egyedi szubsztancia (proté
17	111.33	A szekunder szubsztancia, az absztrakt esszencia (de
18	111.4	Akcidencia. Járulékok
19	111.5	Vonatkozás. Viszony. Reláció
20	111.6	Aktus. Tevékenység. Működés
21	111.61	Lehetőség szerinti és valóságos lét

2. ábra Adatbázis felépítése

Az adatbázis felépítése egyszerű és ezt szeretném meg is tartani. Minden főosztálynak és alosztályoknak külön táblája lesz az adatbázisban, három oszlopot találunk minden táblában. Balról az első oszlop elsődleges kulcs és egy sorszámot tartalmaz, amely minden sort külön azonosít. A visszakeresésnél az azonosító kapcsolja össze a jelzetet annak jelentésével. A második a jelzetet, a harmadik a jelentését tartalmazza. Jelenleg a 3. főosztállyal bezárólag tartalmaz táblákat, valamint a formai, földrajzi és nyelvi alosztályokat digitalizáltuk be.

```

public void megnyit(){
    try {
        Class.forName("org.sqlite.JDBC");
        connection=DriverManager.getConnection("jdbc:sqlite:eto.db");
    }
    catch(ClassNotFoundException e) {
        HuOptionPane.showConfirmDialog(this,"Nem sikerült a JDBC meghajtó betöltése.", "Hiba!",HuOptionPane.ERROR_MESSAGE);
    }
    catch(Exception e) {
        HuOptionPane.showConfirmDialog(this,"Hiba történt!\n"+e.toString(), "Hiba!",HuOptionPane.ERROR_MESSAGE);
    }
}

```

3. ábra *Megnyit* eljárás

A jelzetek olvasása során fontos volt a különböző hibák kezelése, a JAVA tapasztalataim szerint jól együtt tud dolgozni az SQL adatbázissal. Ezt a későbbiekben a lekérdezések-nél is jól láthatjuk.

A jövőben az adatbázisban további kapcsolatok kiépítésére lesz szükség. Ki kell építeni a kereszthivatkozások rendszerét.

4. A szoftver működés közben

Mint azt már fentebb említettem a program Java nyelven íródott. Az alap szoftvert egy hallgatótársam készítette, az alap program kevesebb funkcióval rendelkezett, valamint a hozzá társított adatbázis csupán néhány soros volt.

Formailag folyamatosan változik a program, de minél átláthatóbb és használhatóbb felület kialakítása a cél.

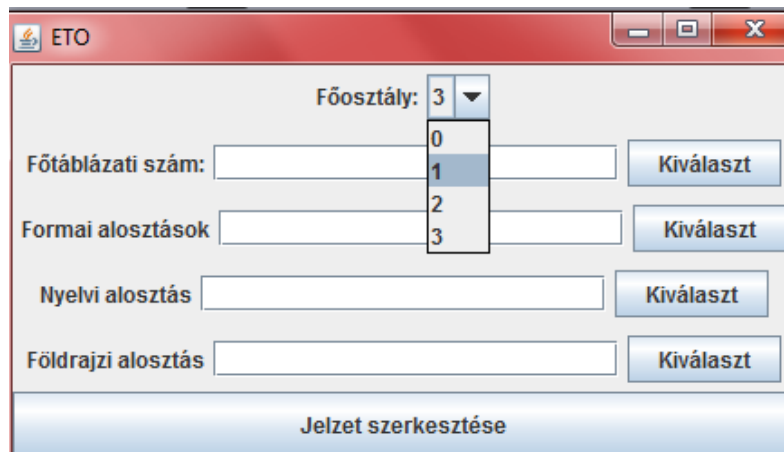
A szoftver használata során az alábbi lépéseken kell végigmennünk:

- 1) Főosztály kiválasztása (tábla az adatbázisban)
- 2) Főtáblázati szám kiválasztása (böngészés vagy keresés)
- 3) Alosztályok kiválasztása (böngészés vagy keresés)
- 4) Jelzet szerkesztése gombra kattintás
- 5) Új ablakban megjelenik a kész jelzet

A fentebbi pontokon egyesével végighaladva szeretném bemutatni a szoftver működését a felhasználói felület és programozás szintjén. A példa mellyel a lépéseken végig kívánok haladni, az egy középiskolás filozófiai szöveggyűjtemény lesz. Tehát a tartalmi feltárást ezen fogom végrehajtani.

4.1. Főosztály kiválasztása

Első lépésünk a megfelelő főosztály (tábla) kiválasztása. A táblákra való szétbontásra azért volt szükség, mert egy táblába zsúfolt jelzetek megnehezítették volna a böngészést és a keresést. A kiválasztás legördülő listából történik. A filozófiát az egyes főosztályban fogjuk megtalálni.



The screenshot shows a window titled "ETO" with a red header bar. Below the header, there is a dropdown menu labeled "Főosztály:" with the value "3" selected. To the right of the dropdown is a list of numbers: 0, 1, 2, and 3. Below this, there are four rows of input fields and buttons:

Field Label	Field Value	Button Label
Főtáblázati szám:		Kiválaszt
Formai alosztások		Kiválaszt
Nyelvi alosztás		Kiválaszt
Földrajzi alosztás		Kiválaszt

At the bottom of the window, there is a blue bar with the text "Jelzet szerkesztése".

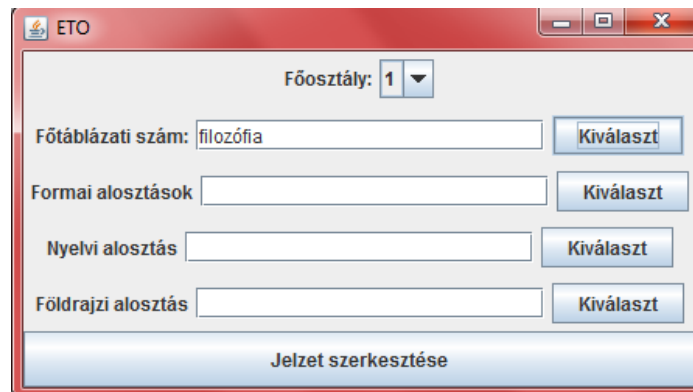
4. ábra Főosztály kiválasztása (programkép)

A program írásánál ez néhány új változó bevezetésével valósult meg. Ezekben kellett tárolni a megfelelő tábla számát. A kiíratáskor a fix értékű változó helyett az újat adtam meg.

4.2. Főtáblázati szám kiválasztása (böngészés vagy keresés)

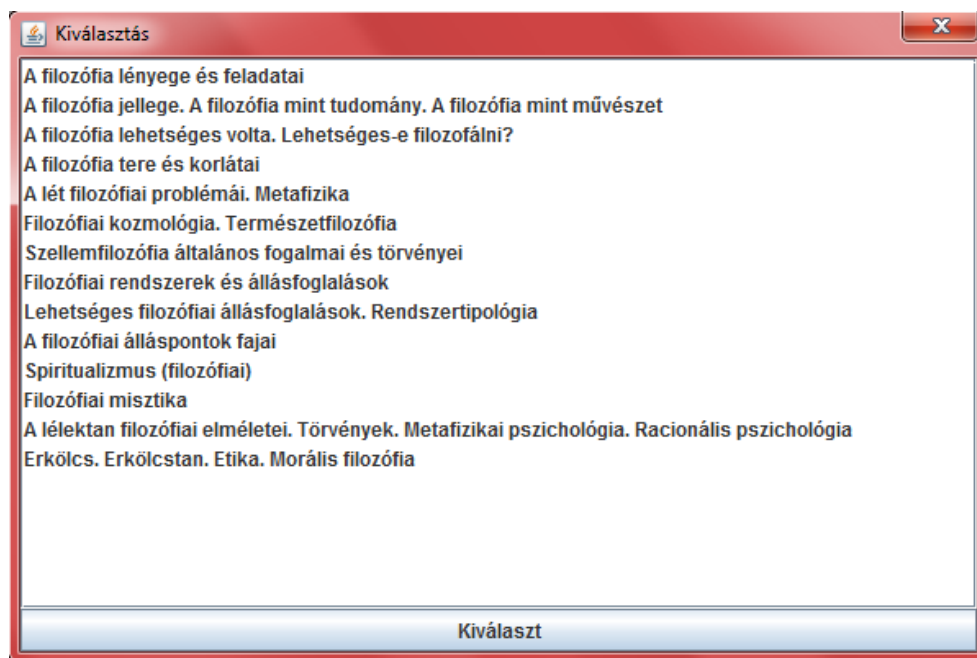
A böngészés és keresés között különbséget a program úgy tesz, hogy a szövegbeviteli mező ki van-e töltve vagy sem. Maga a megkeresési és kiíratási folyamat ugyanúgy megy végbe.

Keresés



5. ábra Keresés (programkép)

A kereséshez, mint azt már említettem a mezőben meg kell adni a feltételt. A példánk miatt mi most a „filozófia” kifejezésre fogunk keresni. A kiválasztás gombra kattintva az alábbi új, felugró ablak jelenik meg.



6. ábra Böngésző ablak (programkép)

Itt láthatjuk az összes megtalált tételt, mely tartalmazta a keresési feltételt. Innen nekünk a „*A filozófia jellege. A filozófia mint tudomány. A filozófia mint művészet*” lesz a legmegfelelőbb. Kiválasztani a sorra való kattintással lehet, döntésünket pedig a kiválaszt gombra való kattintással erősítjük meg.

```

public void actionPerformed(ActionEvent e) {
    foName = (String)foList.getSelectedItem();

    if(e.getSource()==targy){
        keresd=targyszó.getText();
        new Listazo(this, foName);
    }
    if(e.getSource()==forma){
        keresd=formai.getText();
        new Listazo(this, "formai");
    }
    if(e.getSource()==nyelv){
        keresd=nyelvi.getText();
        new Listazo(this, "nyelvi");
    }
    if(e.getSource()==foldrajz){
        keresd=foldrajzi.getText();
        new Listazo(this, "foldrajzi");
    }
    if(e.getSource()==szerkesztes){
        new Jelzet(this, megalkot());
    }
}

```

7. ábra Kattintás kezelés

A megerősítés után az ablak eltűnik és a kiválasztott sor a kereső mezőbe íródik be a fő ablakban.

A programnak először is el kell döntenie, hogy mely parancsot hajtsa végre a menüben leadott utasításra. Ezt a baloldalon látható ábrán lévő módon teszi. A *listazo* eljárást, milyen paraméterekkel futtassa le. Itt dől el, az is, hogy milyen típusú jelzetet fogunk megalkotni (mely táblával kell dolgoznia a programnak).

A *listazo* eljárásban található a *megnyit* eljárás:

amint már említettünk, ez kapcsolja az adatbázist a szoftverhez. Tartalmazza a *listaz* nevű metódust, ami megkapja a *listazo* második paraméterét. Keresésnél ez határozza meg, hogy mely táblában kell keresnünk.

```

public void listaz(String mit){
    try {
        Statement stmt=connection.createStatement();
        ResultSet rs = stmt.executeQuery("SELECT jelentes FROM '"+mit+"' WHERE jelentes LIKE '"+Eto.keresd+"%'");
        while(rs.next()) {
            String nev=rs.getString("jelentes");
            model.addElement(nev);
        }
        rs.close();
        stmt.close();
    }
    catch(SQLException e) {
        HuOptionPane.showMessageDialog(this, "SQL hiba lépett fel!", "Hiba!", HuOptionPane.ERROR_MESSAGE);
    }
    catch(Exception e) {
        HuOptionPane.showMessageDialog(this, "Hiba történt!\nOka:" + e.toString(), "Hiba!", HuOptionPane.ERROR_MESSAGE);
    }
}

```

8. ábra Listaz eljárás

Jól látható az SQL-ben megírt SELECT utasítás, ami az adatbázisból a lekérdezést végrehajtja. Megkapja, azt, hogy hol kell keresni és a kereső kifejezést (*ETO.keresd* hivatkozással). Végül kiírja a jelentés oszlopban található sorokat, ezt listázza a program az új ablakban, ebből választunk magunknak megfelelő jelzetet. Végül az itt előforduló hibákat is kezeljük.

Böngészés

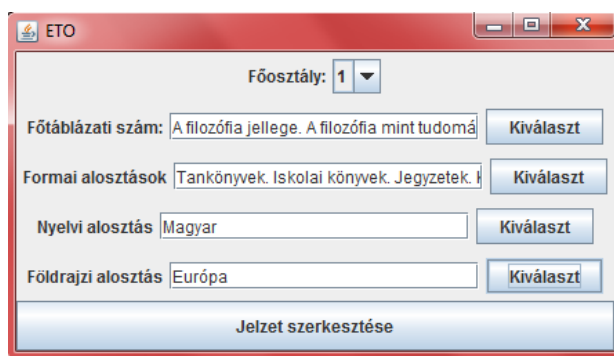
A böngészés elkezdéséhez nem adunk meg keresendő kifejezést. A programozási megvalósítás hasonló a kereséshez, csak a kereső kifejezést tartalmazó paraméterek üresek maradnak, így az adatbázison lefuttatott lekérdezés eredménye az adott tábla összes sora lesz. A hatékony böngészéshez, majd a későbbiekben megfelelő böngészési hierarchiát kell kiépíteni. Jó ötletet adott ehhez a UDC Consortium honlapján található ingyenes részleges ETO jelzet adatbázis.

4.3. Alosztályok kiválasztása (böngészés vagy keresés)

A felhasználónak ugyanazon a lépéseken és felületeken kell végigmennie, mint az előző pontban. A programozási megvalósítás is ugyanaz, hiszen csak a paraméterek és változók módosulnak a döntési folyamatot taglaló rész szerint. A példánkban végzett tartalmi feltárás során az alábbi döntéseket kell meghoznunk:

- Formai alosztás: *Tankönyvek. Iskolai könyvek. Jegyzetek. Képzéshez szükséges szövegek*
- Nyelvi alosztás: *Magyar*
- Formai alosztás: *Európa*

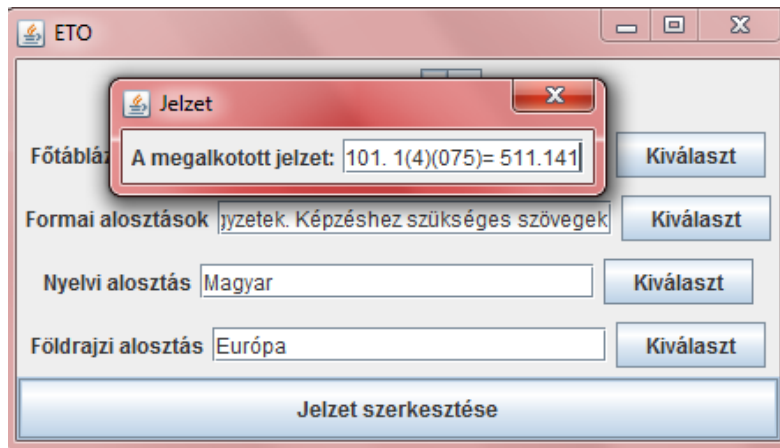
Mindezek után a főablak így fog kinézni.



9. ábra Főablak jelzet megszerkesztése előtt (programkép)

Nincs más hátra, mint a „Jelzet szerkesztése” feliratú gombra kattintás!

4.4. Jelzet szerkesztése



10. ábra Megszerkesztett jelzet (programkép)

Újonnan megjelenő ablakban megkapjuk a megalkotott jelzetet, ami az alábbiakból tevődik össze:

- 101. 1 - *A filozófia jellege. A filozófia mint tudomány. A filozófia mint művészet*
- (4) - *Tankönyvek. Iskolai könyvek. Jegyzetek. Képzéshez szükséges szövegek*
- (075) - *Magyar (nyelvű)*
- = 511.141 - *Európa (területéhez kötődő)*

De mi is történik valójában a programban?

```
public String megalkot(){
    StringBuilder jelzet = new StringBuilder();
    if((targyszso.getText()==null) || (targyszso.getText().equals(""))){
        HuOptionPane.showMessageDialog(this, "Nem választott fő táblázati számot!", "Hiba!", HuOptionPane.ERROR_MESSAGE);
    }
    else{
        jelzet.append(keres(foName, targyszso.getText()));
    }
    if((!(foldrajzi.getText()==null)) && !(foldrajzi.getText().equals(""))){
        jelzet.append(keres("foldrajzi", foldrajzi.getText()));
    }
    if((!(formai.getText()==null)) && !(formai.getText().equals(""))){
        jelzet.append(keres("formai", formai.getText()));
    }
    if((!(nyelvi.getText()==null)) && !(nyelvi.getText().equals(""))){
        jelzet.append(keres("nyelvi", nyelvvi.getText()));
    }
    return jelzet.toString();
}
```

11. ábra Megalkot eljárás

A „Jelzet szerkesztése” gombra történő kattintáskor a *megalkot* nevű eljárás fut le. A metódus megvizsgálja, hogy adtunk-e meg fő táblázati számot. Ha nem hibát ír ki, ugyanis egyelőre csak teljes jelzet szerkesztését feltételezem.

Ezek után az eljárás megvizsgálja, hogy a többi esetben választottunk-e alosztásokat, és ha nem, akkor helyettük semmit nem ír. Mindegyik esetben a már említett *keres* eljárást futtatja le. Végül a különböző gombok kezelésekor meghívjuk a megfelelő gombhoz hozzá-

rendelt *jelzet* eljárásban a *megalkot* metódust.

```
public class Jelzet extends JDialog{
    private JTextField jtf;
    private Container cp;

    public Jelzet(JFrame owner, String jelzet){
        super(owner, "Jelzet", true);
        setLocation(owner.getX()+60, owner.getY()+40);
        cp = this.getContentPane();
        cp.setLayout(new FlowLayout());
        this.add(new JLabel("A megalkotott jelzet:"));
        jtf = new JTextField(jelzet);
        this.add(jtf);
        this.pack();
        this.setVisible(true);
    }
}
```

A *jelzet* valójában a megszerkesztett ETO jelzet kiíratását végzi el egy új ablakban. Paraméterében a kész jelzetet kapja.

12. ábra *Jelzet* eljárás

5. A program alkalmazási területei

A kész program sok könyvtári területen lesz alkalmazható. Természetesen nagyban gyorsítaná a tartalmi feltárást (főleg egy automatizált változat). A leendő könyvtárosok tanulásának segítésére is alkalmas lesz, azáltal, hogy a szoftver online és mobil alkalmazásként is elérhető lenne.

Nagy segítséget jelenthetne, azokban a könyvtárakban is, ahol tartalmi feltárást kell végezni, de kevés a munkaerő. Később a program más nyelvű adatbázisokon olyan országokban is alkalmazható lenne, ahol nem elég fejlett a könyvtári rendszer és a könyvtárosok képzése. Ez jelentős erőforrásokat szabadítana fel és megkönnyítené a könyvtári automatizálás folyamatait is, különösen akkor, hogyha mobil alkalmazásként is elérhető lenne.

6. Hasonló próbálkozások

Ebben a fejezetben a hasonló elgondolású vagy hasonló célokat kitűző fejlesztéseket gyűjtöm össze. Itt főleg azokra a fejlesztésekre gondolok, melyek az UDC osztályozását veszik alapul.

AUDACIOUS - Automatic Direct Access to Information with the Online UDC system

Pauline Atherton (Cochrane) és Robert R. Freeman az American Institute of Physics kutatói a 60-as években foglalkoztak a UDC automatizálásának kérdésével. Sok kutatási eredményt jelentettek meg ezzel kapcsolatban az évek folyamán és az 1965-ös FID konferencián is bemutatták eredményeiket.

Arra jutottak, hogy a két legnagyobb probléma az általam is említett nyelvi, nyelvtani kérdés és a tezaurusz és a UDC összekapcsolása. Úgy vélték, hogy mindezek kiegészülve a decimális számrendszer problémájával és a UDC szabványtalanságaival már erős gátat szabnak az automatizálásnak.

A későbbi próbálkozások is kudarcba fulladtak nagyrészt a fentebb említett problémákra hivatkozva. (Rigby, M. (1981). *Automation and the UDC 1948–80* (2nd ed.). (FID 565.) The Hague: FID.)

UDC summary online

A UDC consortium által fejlesztett online jegyzék logikai hierarchiában tartalmazza a jelzeteket. Mindezt 48 nyelven teszi meg, és a jelzetek le is tölthetők az összes nyelven xml-ben vagy linked data-ként.

Nagy hiányossága, hogy a jelzeteknek csak töredékét tartalmazza (2400 db). Viszont megtartja a hierarchikus sorrendet, amely az alkalmazásban is követendő példa lesz a böngészési felületen. Az útmutató és a hozzászólás funkciók még fejlesztés alatt vannak.

Az alkalmazás ebben az állapotában még csak oktatási célra alkalmas, de én úgy látom, hogy a végső céljuk is ez, hiszen a könyv formátumért ugyanúgy fizetni kell.

7. Automatizálás kérdése

Annak, hogy miért nem mentem el ebbe az irányba, több oka is van. Ha kizárjuk a felhasználót a döntési folyamatból, akkor bizony a feltárás folyamata is a programra hárul.

Az automatikus feltárás problémáját jelenleg csak statisztikai módon tudom elképzelni, mely során a különböző kifejezések gyakoriságának vizsgálatával (az egész dokumentumban) próbálnánk meghatározni a mű témáját, hozzávéve a cím, valamint az esetleges szurrogátum tartalmát. Természetesen ez nem fogja elérni egy ember teljesítőképességét. Először is nem digitális dokumentum esetén el kéne végezni a digitalizálást. Szükségünk lenne olyan nyelvi programra is, mely képes a szavakat nyelvtanilag is kezelni, például lemmatizálni. Előfordulási gyakoriság alapján sorrendet is fel kellene állítanunk. Az így kapott szóhalmazt rendszerezni kellene, valamint egy tezaurusz segítségével meghatározni, hogy mely tudományterületekről származnak. Azt is meg kell állapítanunk, hogy milyen típusú alosztásokhoz tudjuk ezeket hozzárendelni. A programnak magának kellene meghoznia döntéseit. Az ehhez szükséges algoritmusok és azokat szemléltető ábrák bonyolultsága azonban többszörösen meghaladná az itt bemutatottakét.

Tehát mire is lenne szükségünk ahhoz, hogy az automatizálás megoldható legyen?

- A magyar nyelvtant jól használó nyelvi szoftverre
- Folyamatosan frissülő, minden tudományterületet felölelő tezauruszra
- Tanulékony és önálló működésre képes jelzetszerkesztő szoftverre (MI)

Az utolsó fejlesztéséhez, elengedhetetlen az első kettő megléte. A szépirodalmi művek még így is nagy fejtörést okoznának a számítógépnek. Viszont a többi műfaj nem okozhatna gondot a szoftvereknek. Természetesen a jelzetszerkesztő folyamatos felügyeletet követel meg, főleg a kritikus kezdeti időszakban, mikor a legtöbb hiba jöhet elő.

Mindezek után azt mondhatjuk, hogy az automatizálás kérdése még megoldatlan probléma, de még ha megoldható lenne, akkor se lenne képes olyan munkát végezni, mint egy könyvtáros, ezért nem zárható ki és ezért nem is zártam ki a felhasználói döntéseket a jelzetszerkesztés folyamatából.

8. Összefoglalás

A program még nincs kész és jelenleg is fejlesztés alatt áll. Folyamatosan bővitem újabb funkciókkal. Jelenleg elsőrendű célom a teljes ETO digitalizálása és adatbázisba feltöltése. Csak ezek után lehetne a többi alosztályt is hozzákapcsolni a programhoz. Mindezek után lehetne foglalkozni a komplikáltabb jelzetszerkesztési és kiíratási problémákkal.

A jövőben, ha a szoftver fejlesztési állapota már azt megengedi, akkor érdemes lenne azt könyvtári környezetben tesztelni, ugyanis a Library 2.012 konferencián ezt több alkalommal is felvetette a számomra a külföldi hallgatóság. Ehhez viszont elengedhetetlen a teljes digitalizált ETO feltöltése az adatbázisba.

Végül az egyik tanárom tanácsára az ábraszerkesztés helyett a formalizálás irányában szeretném vezetni elméleti munkámat. Ebben a formában fogom megírni az elméleti és logikai alapjait fejlesztésemnek. Az ábraszerkesztéssel történő szemléltetéseim több alkalommal is kudarcot vallottak, főleg azok bonyolultsága és átláthatatlansága miatt.

Biztosan maradtak még megoldatlan problémák, amiket még nem tárgyaltam. Ezek vagy a későbbi kutatásaim során vagy a projektünk programozási fázisában elő fognak kerülni és azzal munkámat kiegészíthetem.

9. Irodalomjegyzék

- 1) Barátné Hajdú Ágnes (2005). *Egyetemes Tizedes Osztályozás*. Budapest: KI.
- 2) Gliffy. Online Diagram Software and Flowchart Software.
<http://www.gliffy.com>
(Letöltve: 2012. november 2.)
- 3) C. McIlwaine (1997). The Universal Decimal Classification: Some Factors Concerning Its Origins, Development, and Influence. *Journal of the american society for information science*, April, 331-339.
http://polaris.gseis.ucla.edu/gleazer/462_readings/McIlwaine.pdf
(Letöltve: 2012. november 2.)

10. Ábra és képjegyzék

1. ábra Általános folyamatábra	4
2. ábra Adatbázis felépítése	5
3. ábra <i>Megnyit</i> eljárás	6
4. ábra Főosztály kiválasztása (programkép).....	7
5. ábra Keresés (programkép)	8
6. ábra Böngésző ablak (programkép)	8
7. ábra Kattintás kezelés	9
8. ábra <i>Listaz</i> eljárás	9
9. ábra Főablak jelzet megszerkesztése előtt (programkép)	10
10. ábra Megszerkesztett jelzet (programkép).....	11
11. ábra <i>Megalkot</i> eljárás	11
12. ábra <i>Jelzet</i> eljárás	12