Szegedi Tudományegyetem

Informatikai Intézet

SZAKDOLGOZAT

Godó Viktor

2016

Szegedi Tudományegyetem
Informatikai Intézet

Adatbázis-kezelő modul fejlesztése 2D műszaki rajzolóprogramhoz

Szakdolgozat

Készítette:
Godó Viktor
programtervező informatikus BSc szakos hallgató

Témavezető:
Dr. Németh Gábor
egyetemi adjunktus

Szeged
2016

Feladatkiírás

A témát kiíró oktató neve: Németh Gábor

Meghirdető egység: Képfeldolgozás és Számítógépes Grafika

Témakör: adatbázis, szoftverfejlesztés

Típus: Szakdolgozat

A feladat megnevezés: Adatbázis-kezelő modul fejlesztése 2D műszaki rajzolóprogramhoz

A feladat angol megnevezése: Development of database modul for 2D CAD application

Hány fő jeletkezhet: 2

Milyen szakos hallgatók számára: Mérnökinformatikus BSc, Programtervező informatikus BSc

Feladat rövid leírása:

A műszaki tervezőprogramok kapcsán elsősorban térinformatikai vagy épületgépészeti alkalmazásoknál van arra szükség, hogy rajzelemekhez plusz információt rendeljünk. A hallgató feladata egy olyan adatbázis-modul készítése egy 2D műszaki rajzolóprogramhoz, amely lehetőséget nyújt a felhasználó számára, hogy az egyes rajzelemekhez plusz információt rendeljen. Ilyen információ lehet például csővezetékek anyaga, áramköri rajzoknál a típus megnevezése, műszaki paraméterek megadása és az alkatrész jelölése. A feladat egy olyan adatbázis-modul elkészítése, amely lehetőséget nyújt a felhasználónak arra, hogy meghatározza, hogy adott rajzelemhez milyen jellemzőket szeretne hozzávenni, a rajzelemeket és jellemzőit egy relációs adatbázisban kezelje, a rajzelemek jellemzőit meg lehessen jeleníteni a rajzolóprogramon belül (párbeszédablakban vagy jelentés formájában).

Az adatbázis készítése során fontos, hogy az az alkalmazással együtt, és ne külön szerverként működjön. Ilyen célt szolgálnak az ún. in-memory adatbázisok (pl. SQLite, HSQLDB).

A modult Java nyelven kell elkészíteni és illeszkednie kell a 2D rajzolóprogram eddigi moduljaihoz.

Szakirodalom: magyar és angol nyelvű

Előismeretek, feltételek: Nincs

Engedélyezte: A téma kiírását engedélyezem: Nyúl László, tanszékvezető, 2014. május 8.

Tartalmi összefoglaló

- ***A téma megnevezése:***
  Adatbázis-kezelő modul fejlesztése 2D műszaki rajzolóprogramhoz.
- ***A megadott feladat megfogalmazása:*** Egy 2D CAD programhoz illeszkedő olyan modul elkészítése, ami egy memóriában futó adatbázist kezel. Az adatbázisban a rajzolóprogramban elkészített, műszaki rajz rajzelemeihez lehet kiegészítő információkat rendelni. A modul feladata a felhasználónak biztosítani olyan felületet, ahol ezeket a plusz információkat kezelheti, hozzáadhatja a rajzelemekhez.
- ***A megoldási mód:*** A rendszer megtervezése az SSADM rendszertervezési módszertan segítségével, majd az adatbázis megtervezése. Az adatbázis tervezésénél a fő szempont, mik és hogyan legyenek strukturálva a rajzelemekhez hozzáadott információk. Az adatbázis létrehozása „in-memory" adatbázisokat támogató adatbázis-kezelő rendszer segítségével. A követelmény katalógus előírásait megvalósító, és az alapprogramhoz illeszkedő modul implementálása és hozzáadása az alapprogramhoz.
- ***Alkalmazott eszközök, módszerek:*** HSQLDB (Hiper Structured Query Language DataBase) adatbázis-kezelő rendszer annak a 2.3.2 verzióját, Dia nevű 0.97.2 verzójú, freeware Strukturált diagramok szerkesztésére szolgáló program, Eclipse (Luna service release) 4.4.1 verziójú integrált fejlesztőkörnyezet.
- ***Elért eredmények:*** Típusok és jellemzőket lehet létrehozni, a jellemzőket típusokhoz lehet rendelni. A modulban rajzelemeket lehet kijelölni, akár egyesével akár többet egyszerre. A kijelölt rajzelemekhez lehet típust hozzárendelni, ezáltal megkapják a típus jellemzőit. A rajzelemek jellemzőihez konkrét értékeket lehet hozzárendelni. A rajzelemek, típusok, és ezek jellemzőik egy memóriában futó adatbázisban tárolódnak. Alkatrész katalógust lehet készíteni, megjeleníteni a képernyőn, azt HTML fájlba tetszőleges helyre kimenteni.
- ***Kulcsszavak:*** típus, jellemző, típus jellemzői, rajzelem jellemzői, „in memory" adatbázis, HSQLDB, MVC, modell, nézet, vezérlő, SSADM, „drag and drop" kijelölés, adatfolyam diagram, egyed-esemény mátrix, egyed-kapcsolat diagram, alkatrész katalógus, HTML, DAO

Tartalomjegyzék

1. BEVEZETÉS

A hagyományos mérnöki tervezőmunka emberi erőforrásigénye a feladatok komplexitásával egyre nőtt. Az ipari termelés fejlődésével felgyorsult az építőelemként használható produktumuk szabványosítása, megjelentek a katalógusok. Az olcsó és nagy teljesítményű digitális számítógépek előtt, a csak az egyes részfeladatok megoldására használható szoftvereket felváltottak a rohamtempóban fejlődő integrált tervező eszközök, a CAD rendszerek. Kezdetben egyszerű rajzeszközként használható alkalmazások gyors ütemben fejlődtek. A műszaki rajzolók a szoftveres fejlődés hatására átalakult, majd a technikusok feladatait is részben a szoftverek, részben a tervezőmérnökök vették át. A katalógusokból kezdetben független gyártói elem-adatbázisok, majd a tervezőszoftverekbe importálható alkatrész-adatbázisok lettek. A CAD szoftverek használatával a tervezési folyamat,másolatok készítése leegyszerűsödött, az elkészített tervek minősége nőtt, a szabványkövetés ellenőrzése egyszerűvé vált, a prototípusgyártás felgyorsult.

A XXI. századi mérnöki tervezőmunka nélkülözhetetlen kellékei lettek a különféle CAD alkalmazások. Ezen szoftverek szimulációs feladatokat is ellátó komplex rendszerekké fejlődtek és a tervezési folyamat teljes spektrumát lefedik mára. A tervezési folyamat kreatív része a már meglévő - előre definiált - elemek mellett, a tervező által készített egyedi elemek felhasználásával történő szoftver-támogatott építkezés. Az építőelemek, hogy használhatóvá váljanak, szükséges, hogy információt hordozzanak magukról. Az elemekhez rendelt különféle adatok, a tervezési folyamat különböző fázisaiban válhatnak fontossá. Például egy integrált áramköri chip lábkiosztása és fizikai kiterjedése nélkül a huzalozás tervezése nem lehetséges, de az alkatrészlistán történő felsoroláshoz, elég csak a típusszáma. Az aktív vagy passzív hűtés tervezéséhez más - csak arra az integrált áramkörre - jellemző tulajdonságokat is figyelembe kell venni. A különféle szimulációs és dokumentum-generáló folyamatok a hozzáadott adatok nélkül nem lehetségesek. Ezeket az adatokat tudni kell kezelni.

Az adatok kezelésének az egyik elterjedt módja a relációs adatbázis-kezelő rendszerek használata. Egy ilyen rendszer az adatokat és az adatok közötti kapcsolatokat tárolja. Az adatok létrehozhatóak, írhatóak, olvashatóak, módosíthatóak és törölhetőek. A CAD szoftverek jellemzően nagyszámú hozzáadott adat kezelését követelik meg. Az ilyenformán felmerülő igényt a relációs adatbázis-kezelő rendszerek segítségével is ki lehet elégíteni.

Szakdolgozatomban egy már meglévő néhány alapfunkciót megvalósító 2D műszaki rajzolóprogramhoz készítek egy - a hozzáadott adatok tárolására használható - adatbázis-

modult. A modul integrálódik az alkalmazásba; a használható funkciók a rajzolóprogramon belül vállnak elérhetővé. Az alapprogramra épülő több más modul is elkészült szakdolgozatok keretében. A már elkészült modulok a következők:

- Magyar Zsuzsanna Gizella - 2D műszaki rajzolóprogram rajzoló és szerkesztő moduljának fejlesztése (2013)
- Németh Júlia Boróka – 2D műszaki rajzolóprogram transzformációs valamint attribútum- és rétegkezelő moduljainak fejlesztése (2014)
- Tomkovics Balázs – 2D műszaki rajzolóprogram blokk készítő és kezelő moduljának fejlesztése (2014)
- Szőcs Réka – Méretező modul fejlesztése 2D műszaki rajzolóprogramhoz (2013)

2. A TERVEZÉS

Ebben a fejezetben a rendszer és az adatbázis tervezése kerül tárgyalásra. A tervezés során az SSADM(Structured Systems Analysis and Design Method)[1] rendszertervezési módszertant követtem.

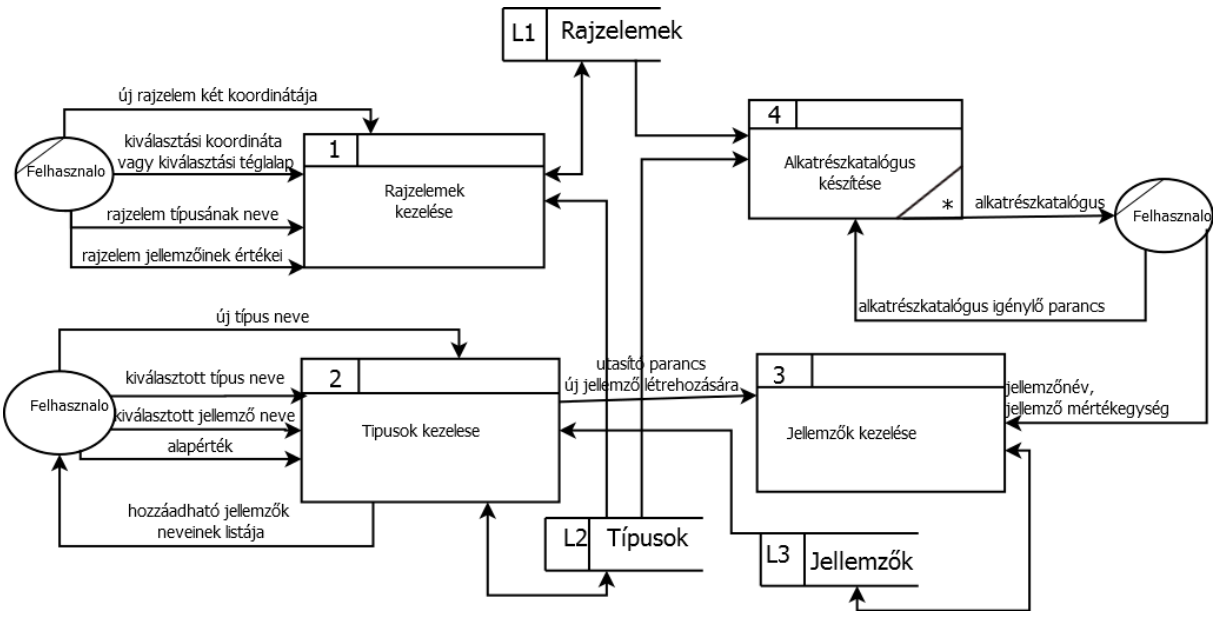
2.1. Követelmény katalógus

Funkcionális követelmények: Illeszkedés az elkészült eddigi modulokhoz. Rajzelemek kiválasztása a rajzlapon. Típusok létrehozása, jellemzők hozzáadása típushoz akár alapértékekkel. Jellemzők létrehozása. Rajzelemekhez típus hozzárendelés. Rajzelemekhez a típushoz megfelelő jellemzőkhöz konkrét érték meghatározása. Katalógus készítése a rajzlapon lévő rajzelemekből.

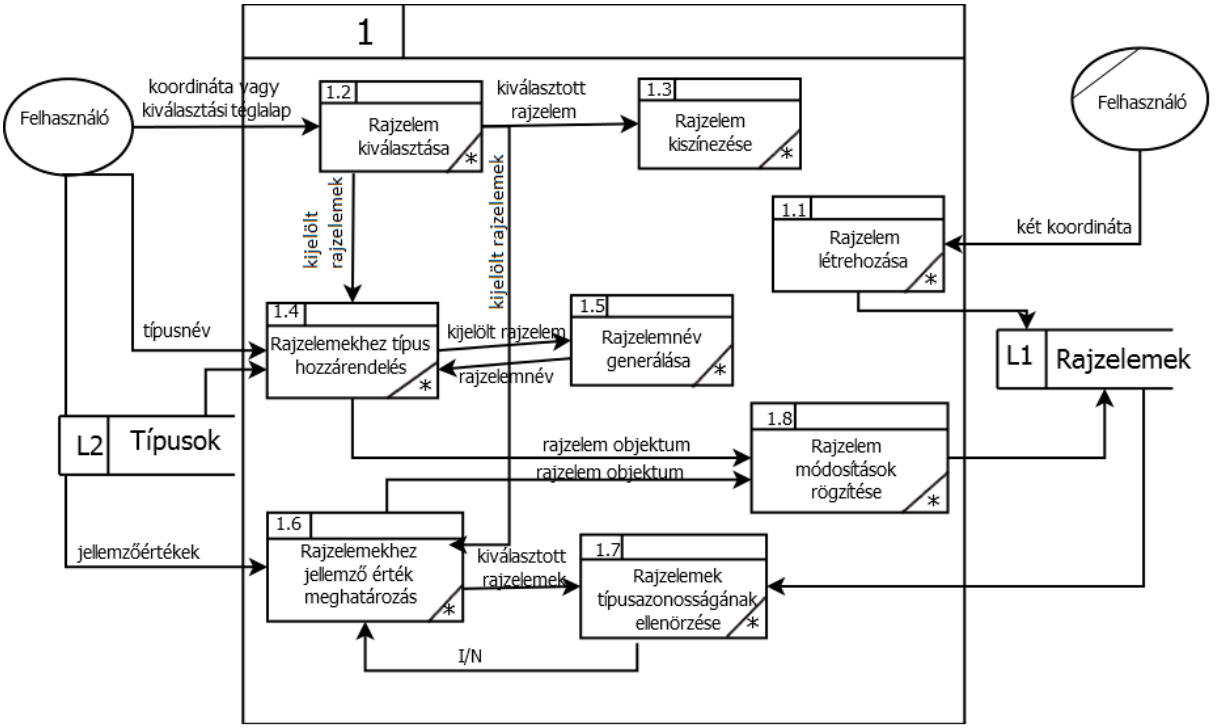
Nem funkcionális követelmények: Kis méretű, RAM-ban tárolható adatbázis, ebből következik adatbázissal való kommunikáció gyorsan történjen.

2.2.A tervezett rendszer adatfolyam diagramjai

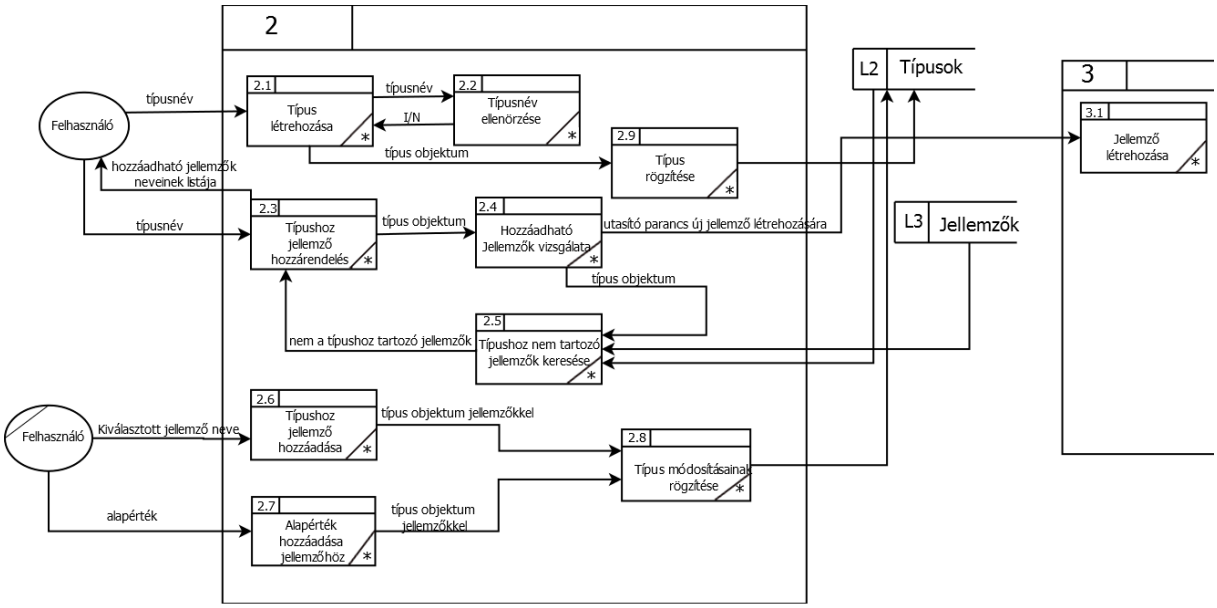
Logkai adatfolyam diagramok



2.1 ábra A tervezett rendszer 1. szintű logikai adatfolyam diagramja

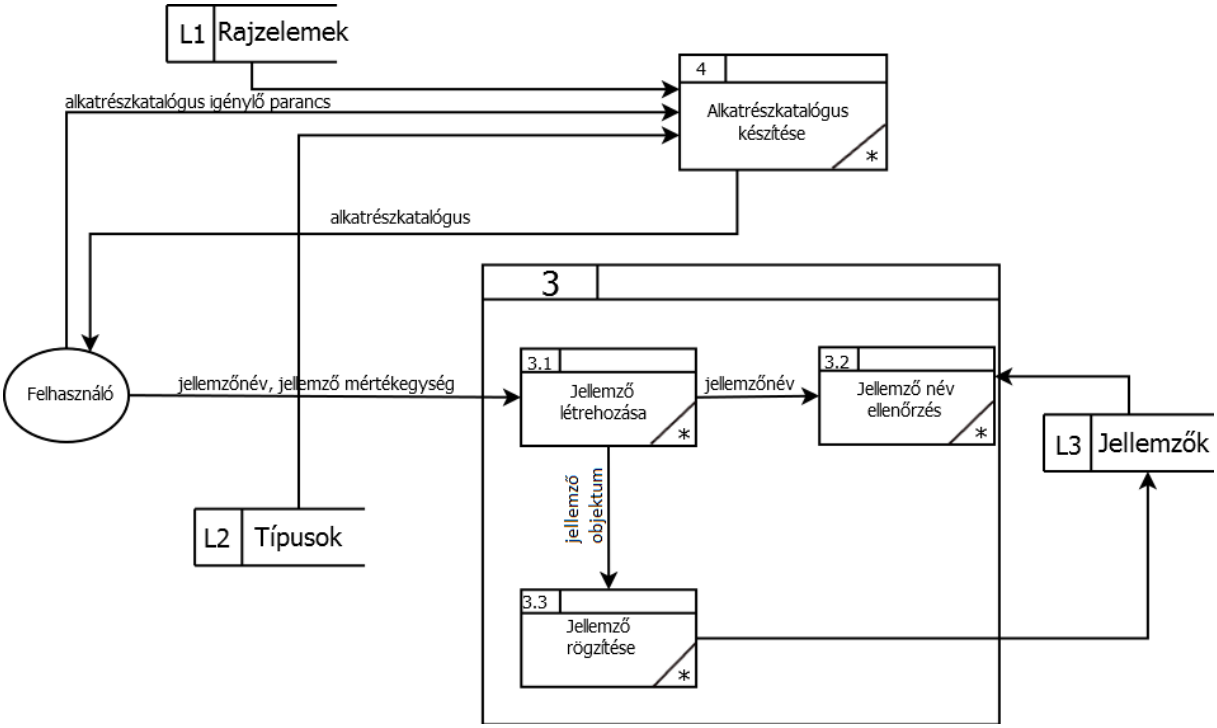


2.2 ábra az 1. folyamat és részfolyamatai



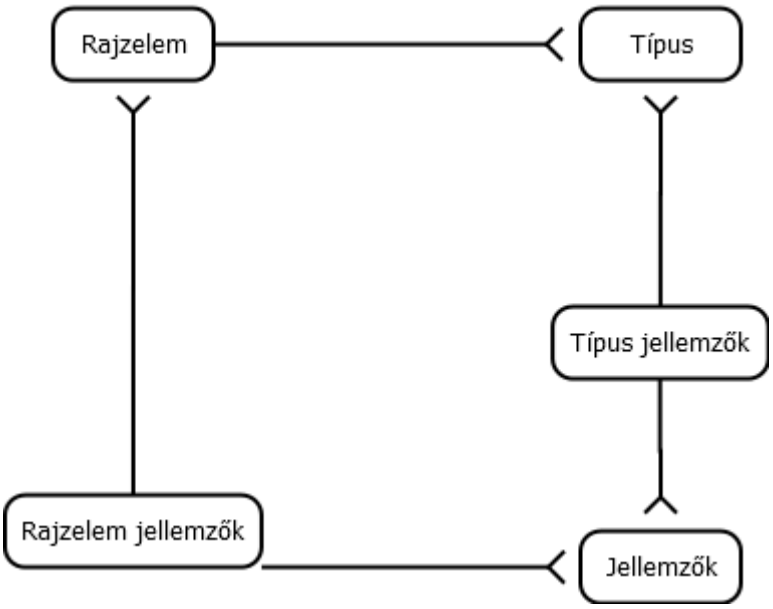
2.3 ábra 2. folyamat: a típusok kezelése.

A tervezett rendszer fizikai adatfolyam diagramjai a melléklet *A függelékében* található.



2.4 ábra 3.folyamat: Jellemzők és Alkatrész katalógus készítése.

2.3. Az Egyedmodell diagram



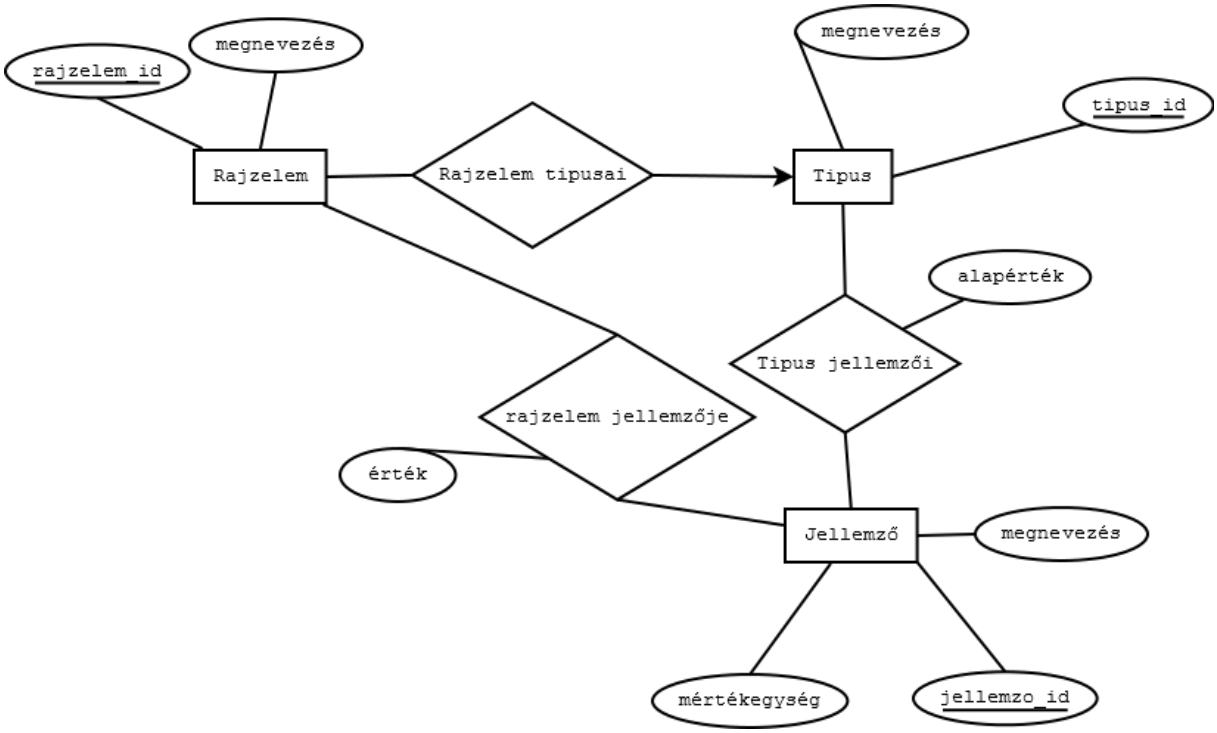
2.5 ábra Egyedmodell diagramja

2.4. Egyed-esemény mátrix

	Konkrét érték felvétele	Mértékegység meghatározása	Típus jellemzőinek meghatározása	Rajzelem elkészítése	Alapértékek felvétele	Speciális típus definiálása	Érvényes érték elbírálása
Rajzelem				**L M**			
Típus			**O**	**O**	**O**	**L O**	
Jellemzők		**O M**					**M T**
Típus jellemzők	**O L M**		**L M**		**L M**	**L O**	**M T**
Rajzelem jellemzők	**O L M**			**L M**			**M T**

2.5 Az adatbázis tervezése

Az adatbázis tervezésénél legfontosabb szempont volt, hogy a rajzlapra felrajzolt rajzelemekhez típusokat rendelhessünk és az adott típus már meghatározza a milyen jellemzőket kell a rajzelemhez rendelni. Természetesen a valóságban bizonyos típusok bizonyos jellemzőik nem változnak, ezért a típus adott jellemzőjéhez alapértékeket lehet hozzárendelni. De ezektől természetesen adott rajzelem esetén el lehet térni. A fenti megfontolások alapján született a következő egyed-kapcsolat diagram:



2.6 ábra Egyed-kapcsolat diagram

Az egyed-kapcsolat diagram leképezése relációsémákká a következőképpen nézz ki:

Egyedek sémái:

RAJZELEM**(**rajzelem_id, megnevezes**);**

TIPUS**(**tipus_id, megnevezes**);**

JELLEMZŐ**(**jellemzo_id, megnevezes, mértékegység**);**

Kapcsolatok sémái:

A tipus jellemzői N:M típusú kapcsolathoz felveszem a :

TIPUS_JELLEMZOI(tipus_id, jellemzo_id, alapérték) sémát.

A rajzelem jellemzője N:M típusú kapcsolathoz felveszem a:

RAJZELEM_JELLEMZŐI (jellemzo_id, rajzelem_id, érték) sémát.

Ezekután a relációsémák:

RAJZELEM(rajzelem_id, *tipus_id* rajzelem_megnevezes,);

TIPUS**(**tipus_id, megnevezes**);**

JELLEMZŐ**(**jellemzo_id, megnevezes, mértékegység**);**

TIPUS_JELLEMZŐI(tipus_id, jellemzo_id, alapérték) ;

RAJZELEM_JELLEMZŐI (jellemzo_id, rajzelem_id, érték);

A relációsémák normalizálása a következő:

1NF :

Minden attribútum atomi, ezért a séma 1NF-ben van.

2NF:

A TÍPUS, RAJZELEM, JELLEMZŐ sémák elsődleges kulcsa egy attribútumból áll és mivel ettől minden, másodlagos attribútum függ, 2NF-ben van.

A TIPUS_JELLEMZŐI, RAJZELEM_JELLEMZŐI sémákban minden másodlagos attribútum az elsődleges kulcs egészétől függ.

3NF:

A TIPUS**,** TIPUS_JELLEMZŐI**,** RAJZELEM_JELLEMZŐI sémák csak egy másodlagos attribútumból állnak ezért 3NF-ben van**. A** RAJZELEM sémában *tipus_id* és a *rajzelem_megnevezés* között semmilyen függőség nincs. A JELLEMZŐ sémában a másodlagos attribútumok közvetlenül függenek az elsődleges kulcsoktól, sem tranzitív, sem részleges függés nincs, csak a teljes elődleges kulcstól. A RAJZELEM és a JELLEMZŐ séma is 3NF-ben van.

Az egyed-kapcsolat diagram leképezése és a normalizálás után így néz ki a relációs adatbázis harmadik normálformában:

RAJZELEM(rajzelem_id, *tipus_id*, rajzelem_megnevezes);

TIPUS**(**tipus_id, megnevezes**);**

JELLEMZŐ**(**jellemzo_id, megnevezes, mértékegység**);**

TIPUS_JELLEMZŐI(tipus_id, jellemzo_id, alapérték) ;

RAJZELEM_JELLEMZŐI **(**jellemzo_id, rajzelem_id, érték);

A normalizált relációsémák alapján a végleges, használt adatbázis specifikáció így nézz ki:

Tipus		
tipus_id	Numeric(8)	A típus azonosítója.
megnevezés	Varchar(40)	A típus elnevezése

TipusJellemzoi		
tipus_id	Numeric(8)	Az adott típus azonosítója.
jellemzo_id	Numeric(8)	A típushoz tartozó jellemző azonosítója.
alapertek	Varchar(40)	A típushoz tartozó jellemző alapértéke.

Jellemzo		
jellemzo_id	Numeric(8)	A jellemző nyílvántartási

		azonosítója.
megnevezes	Varchar(40)	A jellemző elnevezése.
mertekegyseg	Varchar(20)	A jellemzőhöz tartozó mértékegység(ek).

Rajzelem		
rajzelem_id	Numeric(8)	A rajzelem azonosítója.
tipus_id	Numeric(8)	A rajzelemhez tartozó típus azonosítója.
rajzelem_megnevezes	Varchar(40)	A rajzelem megnevezése. Autómatikusan generálódik

RajzelemJellemzoi		
jellemzo_id	Numeric(8)	A rajzelemhez tartozó adott jellemző azonosítója.
rajzelem_id	Numeric(8)	Az adott rajzelem azonosítója.
ertek	Varchar(40)	Adott rajzelem, adott jellmzőjének az értéke.

3. AZ IMPLEMENTÁCIÓ

Ebben a fejezetben a követelmény katalógusban leírt alkalmazás megvalósításáról lesz szó. Azaz milyen a kód felépítése, milyen osztályok tartoznak a modulhoz, azok hogy kapcsolódnak az alapprogramhoz, az alapprogram miben módosult és a funkciók megvalósítása mi módon történt. Az alapprogram az MVC (Model–View–Controller)[2], magyarul modell-nézet-vezérlő szerkezeti mintát követi. Az MVC a programkódot három funkcionális szegmensre osztja. A View avagy a nézet a felhasználói felületet definiálja és a feladata szerint csak felhasználói interfészt nyújt a program használatához. A nézet nem ismeri az alkalmazás logikáját, nem tartozik a hatáskörébe. A Model osztályok tartalmazzák az adatokat pl. adatbázisból kinyert adatok stb. A Controller avagy a vezérlő, összeköti a modellt a nézettel, feldolgozza a felhasználói interakciót, ami hatására a modell megváltozhat. Ennek alapján frissíti a nézetet. Az MVC modell nagy előnye az alkalmazás fejlesztése közben, hogy a funkciók jól szét vannak választva, pl. azok akik a grafikus elemek elkészítéséért felelősek nem kell azzal részletesen törődniük, hogy milyen mögöttes logikával működik a modell. Másik előny az újrafelhasználhatóság, a modell független a nézettől ezért többféle nézettel fel lehet használni, esetleg más projektek mentén. Az alapprogram, ahogyan ez a modul is a három MVC szegmensnek megfelelően, három különböző csomagba („package”) különíti el a forrásfájlokat. Ezek a csomagok: model, view, control.

3.1. Felhasznált szoftvereszközök

Mivel az alapprogram java nyelven íródott ezért az Eclipse a programot az Eclipse (Luna service release) 4.4.1 verziójú integrált fejlesztőkörnyezetben készítettem el. A futtatásokhoz 8-as verziójú (71. frissítés) Java-t használtam. Az adatbázist a HSQLDB (Hiper Structured Query Language DataBase) [3] adatbázis-kezelő rendszerben hoztam létre. A választásom azért eset éppen rá, mert támogatja az „in-memory” azaz a memóriában futó adatbázis-kezelést és van JDBC meghajtója is. Mivel az adatbázis kicsi és a teljesítmény szempontjából fontos a lemezre írás elkerülése. A HSQLDB-nek a 2.3.2 verzióját használtam.

3.2. A model osztályai, Bean osztályok

Ebben a részben elsősorban a model csomag osztályairól lesz szó.

3.2.1. Type

Ez az osztály írja le a típust. A felhasználó által létrehozva, vagy a adatbázisból kinyerve a típusokat a program Type osztályú objektumaival reprezentálja és ezt módosítja mielőtt az adatbázisba vissza, vagy a képernyőre kerülne a módosítás. A módosításokat egy típuson a Type függvényeivel történik. A `getID()` hívásával nyerhetjük ki a típus adatbázisban eltárolt azonosítóját. A `setID()` segítségével beállíthatjuk a típus azonosítóját. A típus elnevezését a `getname()`-el kapjuk vissza karakterláncként. A `setname()` pedig a típus nevének megadására szolgál. Az adott típus jellemzőinek halmazát a `getParameters()` kérhetjük le. A `setParameters()` metódushívással pedig lecseréli a típus jellemzőinek halmazát a paraméterben átadott halmazra. Jellemzőket egyenként is hozzáadhatunk a típus jellemzőinek a halmazához az `addParameter()`-el. A `deleteParameter()`-el pedig törölhetjük a típus jellemzőinek halmazából az adott paramétert.

3.2.2. Drawing Element

Az alapprogramban is meglévő osztály, ebben a modulban pár új metódussal bővült. Alapfeladata szerint minden rajzolható elemi grafikai primitív (vonal kör stb.) belőle származik. Az adott rajzelem adatbázisban tárolt azonosítóját a rajzelemnek a `getID()`adja vissza, ha van neki. Ennek az azonosítónak a megváltoztatására, beállítására a `setID()` szolgál. A rajzelemhez a típust beállítani a `setType()` hívásával tudjuk elvégezni. A rajzelem típusát a `getType()` hívásával kapjuk vissza. A `setname()` hívásával beállíthatjuk a rajzelem nevét, míg a `getname()`-el visszakaphatjuk azt. A program jelenleg generált rajzelem elnevezéseket használ, ami a rajzelem azonosítója és a típusa nevének kombinációjából adódik össze. A `setParameters()`-el felcserélhetjük a rajzelem jellemzőinek a halmazát, egy új halmazra. A `getParameters()` hívásával pedig visszakaphatjuk ugyanazt a halmazt. Itt a rajzelem jellemzőin az adatbázisban tárolt, rajzelem típusának megfelelő jellemzőt kell érteni, és nem a rajzelem küllemét leíró jellemzők. Az `addParameter()` segítségével, a jellemzőket egyenként is hozzáadhatjuk a rajzelemhez. Egy megadott jellemzőt pedig a `deleteParameter()` segítségével távolítható el a halmazból.

3.2.3. Parameter

Ez az osztály írja le a jellemzőt. A rajzelem és a típus jellemzőit is. Mind a két esetben ugyanúgy lehet használni hisz az adatbázisfüggvények , úgyis csak a számukra fontos attribútumokat használják fel. A jellemzőhöz a `setID()` függvénnyel tudunk beállítani az azonosítót.

Ugyanezt az azonosítót `getID()` segítségével lehet megszerezni. A `getname()` visszaadja az adott jellemző megnevezését. A `setname()` metódushívással beállíthatjuk azt. Az adott jellemző mértékegységét a `setmeasure()` metódus hívásával módosíthatjuk. A `getmeasure()`-el pedig visszakaphatjuk azt. A `setdefaultValue()` a jellemző alapértékét állíthatjuk be. A jellemző alapértéke a típustól függ, tehát egy jellemzőnek több különböző alapértékkel is rendelkezhet. Amikor a típushoz jellemzőt rendelünk akkor van értelme beállítani az alapértéket. A `getdefaultValue()` kaphatjuk vissza a beállított alapértéket.

3.2.4. SelectingRectangle

A SelectingRectangle egy speciális rajzelem (DrawingElement) ami a többszörös kiválasztásnál kirajzolandó téglalapot valósítja meg, ami a kiválasztási területet jelzi. A téglalapot a két átellenes pontjával lehet definiálni. Az egyik sarokpont x koordinátáját a `setX1()` hívásával állíthatjuk be. Az y koordinátáját `setY1()` segítségével. Visszakapni az x koordinátáját a `getX1()` az y-t a `getY1()`- el lehet. A másik sarokpont x,y koordinátáját ennek megfelelően a `setX2()`, `setY2`() segítségével lehet változtatni, és a `getX2()`, `getY2()` metódushívásokkal kinyerni. Megadásukra van egy másik lehetőség. A `setStartpoint()`-al beállíthatjuk a kiválasztó téglalap bal felső sarokpontját, a `getStartpoint`()-al pedig kinyerhetjük azt. A jobb alsó sarokponton pedig a `getEndpoint()`, `setEndpoint()` segítségével dolgozhatunk. Lehetőség van a téglalap kiterjedését a hosszával és szélességével módosítani is. A kijelölő téglalap kirajzolására csak a többszörös kijelölés esetén van szükség. Amint a kijelölésnek vége a téglalapot el kell tüntetni. Erre szolgál a `setToDefault()` metódus, ami visszaállítja a kijelölő téglalapot a kiinduló helyzetébe, ezzel a rajzlapon láthatatlanná válik.

3.2.5. LineSelector

Az osztály arra szolgál megoldási módokat, kiválasztáskor a vonal (Line) rajzelemre kattintva, ha egyszerre több vonalat érint a kiválasztás, akkor mely vonal kerüljön bele a kijelölt rajzelemek közé. Két féle kiválasztási módszer van megvalósítva. Az egyik a ilyen a SELECT_THE_NEAREST azonosítójú, a másik pedig a SELECT_THE_NEWEST. Amikor a vonal kiválasztása megtörténik és több lehetséges vonal rajzelem is szóba jöhet, akkor a `selectLine()` metódus kerül meghívásra ami a beállított kiválasztási módnak megfelelően hívja meg a megfelelő eljárást. A SELECT_THE_NEAREST kiválasztási mód esetén a `selectNearest()` kerül a meghívásra, ami a vonal rajzelemek közül a kiválasztás koordinátához legközelebb eső kerül kijelölésre. Amennyiben SELECT_THE_NEWEST mód van

beállítva akkor a rajzelemek közül az utolsóként megrajzolt kerül kiválasztásra. A kiválasztási módot a `setMode()` metódussal lehet változtatni.

3.2.6. Line

Az alapprogramban is meglévő osztály, a rajzlapra rajzolható egyenes vonalszakaszt valósítja meg. A DrawingElement-től öröklött hozzáadott metódusokon felül egy új metódussal bővült. Az `isOnLine()` függvény a paraméterben megadott koordinátára megmondja, hogy az az adott vonalon van-e vagy sem.

3.3. Adatbázis kapcsolat osztályai

Ebben a fejezetben az adatbázis létrehozásáról, és az adatbázist kezelő osztályokról lesz szó. Továbbá arról, hogy azok hogyan illeszkednek az implementáció többi részéhez. Az adatbázis kapcsolat a program elindulásakor inicializálódik, a DataBaseController osztály segítségével. A Dao osztály createDataBase() függvényével hozzuk létre az adatbázist a memóriában.

3.3.1. DatabaseController osztály

A DataBaseController osztály grafikus felület vezérlői és Dao között helyezkedik el. A vezérlők adatigényléseit és adatrögzítéseit továbbítja a Dao-hoz. A program inicializálásakor hozza létre a Dao (Data Access Object) osztályt.

3.3.2. Dao osztály

Ez az osztály tartalmazza, szolgáltatja azokat a metódusokat, eljárásokat amik közvetlenül írnak az adatbázisba vagy adatokat olvasnak ki belőle. Ezeket a metódusokat a DataBaseController osztály használja hívja az adott kérésnek megfelelően. Az osztály a model csomagban található. Az adatbázis kezelésére a HSQLDB JDBC meghajtó által biztosított függvényeit használja. A program inicializálásakor a DataBaseController osztály meghívja a Dao `createDataBase()` függvényét ami létrehoz egy memóriában futó „in memory” adatbázist. Továbbá definiálja és létrehozza az adatbázis tábláit, szekvenciáit. Csak ezután válik a modulbéli adatbázis használhatóvá. Egy adott azonosítóval rendelkező típus lekérését az a adatbázisból a `getType()` metódus végzi el. A `getTypes()` pedig az összes típust adja vissza ami az adatbázisban fellelhető. Az `AddType()` hívásával egy paraméterben átadott típusnak megfelelően egy új típus rekord kerül be az adatbázisba. A `DeleteType()` segítségével pedig törölni egy adott típust az adatbázisból. A `getTypeParameters()` visszaadja az adatbázisba írt adott típus jellemzőit. A `getNotTypeParameters()` pedig az

összes olyan jellemzőt adja vissza ami nem a típushoz tartozik. Egy típushoz egy adott jellemzőt a `AddTypeParameter()` lehet hozzáadni és a `DeleteTypeParameter()` segítségével lehet törölni. Új jellemzőrekordot az adatbázisba a `AddParameter()` metódus használatával lehet fölvinni. Adott jellemzőt törölni pedig a `DeleteParameter()` használatával történik. Az adatbázisban fellelhető összes jellemző halmazát pedig a `getParameters()` adja vissza. Új rajzelemet hozzáadni az adatbázishoz az `AddDrawingElement()` metódus hívásával lehet. Továbbá a rajzelem felvételekor a rajzelem típusa is beállítódik. Adott rajzelem eltávolítására pedig a `DeleteDrawingElement()` metódus szolgál. Adott rajzelemhez egy jellemző hozzáadását az adatbázisban az `AddDrawingElementParameter()` metódus végzi el. Jellemzők egy adott halmazát is hozzá lehet adni a rajzelemhez ha meghívjuk a `setDrawingElementParameters()` függvényt. A rajzelem egy adott jellemzőjét eltávolítani pedig a `DeleteDrawingElementParameter()` segítségével lehet. A rajzelemhez tartozó jellemzők halmazát a `getDrawingElementParameters()` függvény adja vissza. Az adatbázisban az összes fellelhető rajzelemet pedig a `getDrawingElements()` segítségével lehet lekérni.

3.3.3. Structures osztály

Statikus karakterlánc(String) konstansokat tartalmaz. A fizikai adatbázis tábláinak mezőneveit tartalmazza. Elsősorban DAO osztály használja SQL lekérdezéseknél, a mezőnevekre hivatkozik a Structures karakterláncaival.

3.4. Funkciók megvalósítása

Ebben az alfejezetben a funkciók szerinti lebontásban fogom tárgyalni azt, hogy mely funkciók mely osztályok, metódusok vesznek rész és mik a szerepük és milyen módszereket használtam az adott funkció betöltésére. A funkciókon a követelmény katalógusban leírt funkciók szerint kerül tárgyalásra.

3.4.1. Rajzlap működési mód váltása

A rajzlap, amire rajzolunk háromféle módban (állapotban) működhet ennek alapján dől el milyen események történhetnek a rajzlapon. Ezek a következők: MODE_DEFAULT, MODE_DRAW, MODE_SELECT. Az első az alapvető mód, iniciálisan a program indulásakor állítódik be. Ekkor se rajzolni se kiválasztani nem lehet. A második a rajzolás és a harmadik a rajzelemek kiválasztását reprezentálja. Egyszerre csak egy mód lehet érvényben, a felhasználó módosítani a főablakon, a baloldalon lévő eszközpanel ikonjaival tudja (4.5 ábra). A

funkciót megvalósító osztály a CanvasModeController osztály. A control csomagban található.

A CanvasModeController elkapja az eszközpanel ikonjaira érkező egérkattintás eseményeket és ez alapján állítja be, mely osztály jogosult a rajzlapon történő események kezelésére. A felhasználó rákattint valamelyik ikonra, akkor a vezérlő `mouseReleased()` metódusa kerül meghívásra, ekkor a attól függően melyik ikon lett lenyomva meghívja a `setSelectMode()` vagy a `setDrawMode()` metódust. A `setSelectMode()` esetén a DrawingController `disableController()` metódusával a rajzolási vezérlőt kapcsolja ki, és a SelectController `enableController()` metódus hívásával engedélyt kap a rajzlapon történő egéreseményekfigyelésére. Majd az aktuális mód átállítódik MODE_SELECT-re. A `setDrawMode()` esetén ennek fordítja történik és a jelenlegi mód átállítódik MODE_DRAW-ra. A `setDefaultMode()`-al összes vezérlőt kikapcsolhatjuk a rajzlap eseményeinek figyelésére és a jelenlegi mód MODE_DEFAULT-ra módosul.

3.4.2.Rajzelem kiválasztása rajzlapon

Ahhoz, hogy a rajzelemekhez hozzá lehessen rendelni típusokat, és ennek megfelelően jellemzőértékeket lehessen beállítani, szükséges feltétel az, hogy valamilyen módon a rajzlapon lévő rajzelemeket ki lehessen választani. Kétféle kiválasztás van: egy egérkattintással egy elemre rákattintva, vagy a bal egér lenyomva tartásával, és mozgatásával egyszerre több elemet választhatunk ki, de csak ha a kijelölő téglalapon teljesen belül vannak (drag and drop módszer). A funkció csak akkor lép működésbe, ha a rajzlap működési módja az előző fejezetben említett MODE_SELECT –re van állítva, és ha a vezérlő aktiválva van. A funkciót megvalósító vezérlő osztály a SelectController osztály. A kiválasztás kétféle állapotban állhat az iniciális STATE_DEAFULT, ekkor nincs semmi kijelölve vagy STATE_SELECTED állapotban, ekkor van kijelölt rajzelem a rajzlapon.

Amikor a felhasználó kiválasztja a kiválasztási funkció ikonját, akkor a CanvasModeController meghívja a SelectController `enableController()` metódusát és ekkor a vezérlő megkapja az engedélyt a vászonra érkező egéresemények elkapására. Ekkor a MouseListener osztálya kapja el a rajzlap(vászon) egér eseményeit és a vezérlő a STATE_DEFAULT állapotba kerül. Amikor a felhasználó kattint egyet a rajzlapon bárhol, akkor meghívódik a MouseListener `mousePressed()` metódusa. Amennyiben a felhasználó a bal egérgombot nyomta le és már volt kiválasztva valami, akkor új kiválasztási procedúra kezdődik és az eddigi kiválasztási halmazt üríteni kell. Ezt a feladatot végzi el a `deselect()`. Mivel ekkor nincsen semmi kijelölve az állapot visszaáll STATE_DEFAULT-

ra. Az `ereaseSelectedObjectColor()` meghívására a kiválasztott rajzelem az eredeti színüket nyerik vissza. Ezután vizsgálni kell, hogy a kattintás nem rajzelemen történt-e? A `receiveCursorPoint()`-al megkapjuk a kattintás koordinátáját, a rajzlap koordináta rendszerében. A Line `isOnLine()` metódusával megtudjuk, hogy az adott vonalat érintette-e a kattintás. Amennyiben igen, akkor az a rajzelem bekerül a kiválasztott rajzelemek közé és az állapot STATE_SELECTED-re változik. Abban az esetben, ha több vonalat is érintett, akkor a LineSelector `selectLine()` metódusa kiválasztja a beállított kiválasztási szabálynak megfelelő vonalat. Ezután a kiválasztási téglalap SelectingRectangle kezdőpontját beállítjuk az előbb megkapott koordinátákra annak `setX1()`, `setY1()` metódusokkal. Amennyiben a felhasználó a „drag and drop" módszerével több elemet akar kijelölni akkor automatikusan meghívódik `mouseDragged()` metódus. Az egér mozgásának megfelelően a tartalmazó téglalap koordinátáit is frissíteni kell. Ezt a `refreshSelectRectangle()` valósítja meg. Mivel a kirajzolni való tartalom folyamatosan változik ezért a DrawingCanvas `repaint()` metódusával ennek megfelelően újra kell rajzolni a rajzlapot. Amikor a felhasználó felengedi a lenyomott egérgombot akkor a `mouseReleased()` automatikusan meghívódik. Amennyiben a felengedett egérgomb a jobb volt akkor egy lenyíló menüt (SelectPopUpMenu) kell előhívni az egérkattintás helyén a `doPop()` hatására. A bal egérgomb felengedése hatására a kijelölt rajzelemek számbavétele történik. Az egyszeri kijelölés esete egyszerűbb, hiszen csak a `setSelectedObjectColor()` metódus hívásával ki kell színezni a kijelölő színnel a rajzelemeket. A többszörös kijelölés esetén számba kell venni, mely rajzelemek kerülnek a kijelölő téglalapon belülre, és hozzá kell adni őket a kijelölt elemekhez. Ha volt a kijelölt területen belül elem, akkor a vezérlő állapotot át kell állítani STATE_SELECTED-re. Majd végül ki kell színezni őket. Ezután a kijelölő téglalapot vissza kell állítani az eredeti állapotába a `setToDefault()` segítségével. Végül a rajzlapot a `repaint()`-el újra kell rajzolni, hogy a kijelölések eredményei láthatóak legyenek.

A funkcióhoz az alapprogrami rajzlapon kívül, egyetlen grafikus elem tartozik hozzá: a SelectPopUpMenu. Ez egy lenyíló menü ami a 4.15 ábrán látható. Ezzel érhetők el azok a funkciók amelyekkel a kijelölt rajzelemek típusát meg lehet adni és a rajzelem jellemzőihez értéket lehet rendelni. A két funkció elérését a ACTION_DEFINE_TYPE_MENU_ITEM, és a ACTION_SET_PARAMETERS_MENU_ITEM azonosítójú akcióval lehet elérni. Amikor van kijelölt rajzelem a rajzlapon és a jobb egérgomb lenyomásakor a SelectPopUpMenu feljön és a felhasználó valamelyik menüelemre kattint meghívódik a SelectPopUpMenu `actionPerformed()` metódusa. Ekkor a „Define Type" menüelemre kattintva a ACTI-

ON_DEFINE_TYPE_MENU_ITEM akció hatására a 3.4.5 fejezetben ismertetésre kerülő SetTypeFrame kerül megjelenítésre annak `createAndShowGui()` metódusának hívásával. Az ACTION_SET_PARAMETERS_MENU_ITEM akció pedig a „Set Parameters” menüelemre kattintva lép életbe ekkor SetTypeParameterFrame által megvalósított párbeszédablak kerül megjelenítésre. Amiről bővebben a 3.4.6 fejezetben lesz szó. Azonban ez a menüelem csak akkor válik elérhetővé, ha a rajzelem kiválasztás során kiválasztás során a `doPop()` előtt meghívódik a kiválasztás vezérlőnek a `typeTest()` metódusa ami azt vizsgálja, hogy a kijelölt rajzelemek mindegyike rendelkezik-e már típussal. Ha igen akkor a SelectPopUpMenu `enableParameters()` hívásával válik ez a menüelem elérhetővé.

3.4.3. Típusok létrehozása, törlése. Jellemzők hozzáadása típushoz és azok törlése

Ahhoz, hogy a rajzon lévő rajzelemekhez típust rendeljünk hozzá, elsőként definiálni kell a kívánt típust. Azaz meg kell adni a típus nevét és meg kell adni azt, milyen jellemzők írják le jól az adott típust. Csak ezután lehet, érdemes, a rajzelemekhez típust rendelni. Természetesen fontos, ha van egy kész, használt típusunk akkor azt lehessen módosítani vagy esetleg törölni. Ezekhez a funkciókat látja el a TypeController vezérlő osztály és biztosít hozzá grafikus párbeszédablakot a TypeFrame osztály. A TypeController vezérlő osztály a TypeFrame párbeszédablak ki-beviteli interakcióit kezeli, ami a felhasználó és a program között zajlik. A felhasználói interakció közben a következő események, akciók következhetnek be: ACTION_ADD_TYPE, ACTION_DELETE_TYPE, ACTION_SHOW_PARAMETERS, ACTION_ADD_PARAMETER, ACTION_DELETE_PARAMETER. Ezek az események: a típus hozzáadás, törlés, típus jellemzőinek mutatása, típushoz jellemző hozzáadása, típushoz tartozó paraméter(ek) törlése.

A vezérlő inicializálásakor a TypeFrame `addTypeFrameActionListener()` metódusa hívódik meg, ami hozzáadja a vezérlőt a párbeszédablak eseményeinek a figyelésére. Amikor a felhasználó a „Units” menüből kiválasztja a „Define Types” menüpontot akkor a meghívódik a TypeFrame `createAndShowGui()` metódusa és a párbeszédablak megjelenítődik . A párbeszédablakban adott akció bekövetkezésekor (pl.: megfelelő gomb lenyomásakor) automatikusan meghívódik a vezérlő `actionPerformed()` metódusa. Az akciókat, azok fent említett azonosítója alapján kerül megkülönböztetésre és ez alapján hívódik mega megfelelő eljárás.

Amikor a felhasználó az „Add a Type” nyomógombra kattint akkor az ACTION_ADD_TYPE akció lép életbe és az `actionPerformed()` metódus ennek megfelelően a

vezérlő `addType()` metódusát hívja meg, ami a típus létrehozás interakcióját kezeli. Először a típus nevét kell megadni, tehát a vezérlő utasítja a párbeszédablakot a `showInputMassage()` metódusának hívásával, egy szöveges bemeneti mezővel rendelkező dialógusablak megjelenítésére majd a begépelt típus nevét a `getInputString()` metódushívással kapja meg. Ezután a vezérlő a `isTypeAlreadyExist()` függvényét hívja, ami visszaadja, hogy a megadott típusnév használatban van-e? Érvényes típusnév esetén, meghívja a DAO osztály `addType()` metódusát így azt adatbázisba felkerül egy új rekord ami az új típust reprezentálja. Végül a TypeFrame `addType()` metódus hívásának köszönhetően hozzáadódik az új típus, a grafikus felület listájához. A „Delete a Type" gomb lenyomásakor azaz ACTION_DELETE_TYPE akció hatására a vezérlőben a `deleteType()` metódus hívódik meg, ami a típus törlés interakcióját kezeli. Elsőként lekéri a párbeszédablaktól, a listában kiválasztott típust a `getSelectedType()` metódus hívásával. Mivel a kiválasztott rajzelemet a grafikus felületről csak a típus nevét adja vissza ezért a vezérlő a `lookupTypeByName()` metódussal nyeri ki a konkrét Type objektumot az adatbázisból. Ezután a `isTypeinUse()` metódussal a vezérlő megvizsgálja, hogy van-e a típussal rendelkező rajzelem. Amennyiben nincs, akkor kijelölt típus törölhető. A vezérlő törli a típust az adatbázisból a Dao objektum megfelelő függvénye segítségével és utasítja a párbeszédablakot `removeType()` hívásával, hogy törölje ki a típus nevét a listából.

A típus paramétereinek kiíratásához a felhasználó a „Show Parameters" gombra kattint akkor a ACTION_SHOW_PARAMETERS azonosítójú akció hatására a `showParameters()` vezérlő metódus hívódik. Amennyiben van értelmes kijelölt típus akkor a `showPreferences()` metódusban az adatbázisból lekérjük a típus paramétereit a Dao `getTypeParameters()` metódusának hívásával. Ezután a kinyert jellemzők objektumaira meghívjuk a `getDataLine()` függvényt ami az adott jellemző tulajdonságait adja vissza táblázatba beírható formátumba. A vezérlő a jellemzőket a grafikus felületre való kiírására a TypeFrame `addParamter()` függvényhívásával valósítja meg. Az „add Parameter" gomb lenyomására az ACTION_ADD_PARAMETER akció hatására a vezérlő meghívja az `addParameter()` esemény kezelő metódust. Ha van párbeszédablakban kiválasztott típus (amit a korábban említett a `getSelectedType()` metódussal lehet kinyerni a grafikus felülettől) akkor lekérjük az adatbázisból azokat a jellemzőket, amik nem a kiválasztott típushoz tartoznak a Dao `getNotTypeParameters()` metódusának segítségével. Ezután a vezérlő utasítja a TypeFrame-t egy szelekciós dialógusablak (4.11 ábrán látható) kirajzolására, ahol a felhasználó kiválaszthatja a jellemzőt amit hozzá akar rendelni a típushoz. Amennyiben nincs

hozzáadható jellemző, akkor vezérlő megjeleníti a jellemző létrehozására alkalmas párbeszédablakot (ParameterFrame). A ParameterFrame-ről a következő fejezetben lesz szó. Abban az esetben, ha van hozzáadható jellemző, akkor a típus nevének megadásához hasonló módon bekérjük az adott jellemzőhöz az alapértéket, majd hozzárendeljük a Parameter `setdefaultValue()` metódus hívásával. Ekkor hozzáadjuk a típus objektumhoz az adott jellemzőt a Type `addParameter()` metódusával .Végül a jellemző bekerül az adatbázisba a típus jellemzői közé. Amennyiben már vannak a kiválasztott típussal megegyező típusú rajzelemek a rajzlapon, akkor a vezérlő `updateDrawingElements()` metódushívással a rajzelemekhez is hozzáadja a típushoz hozzáadott jellemzőt. Az utolsó akció amit a vezérlőnek kezelni kell az ACTION_DELETE_PARAMETER akció ami a „Delete Parameter” gomb lenyomására válik relevánssá. Ezt a feladatot a `deletePreferences()` metódus végzi el. Először lekérjük a párbeszédablaktól a felhasználó által, a táblázatban kijelölt sorokat a TypeFrame `getTableLines()` metódusával. Mivel a jellemző(ke)t szövegesen kapjuk vissza ezért a paraméterek objektumait a `lookupParameterbyName()` metódussal nyerjük ki, ami visszaadja a névnek megfelelő jellemző objektumot. Ezután töröljük az adott típus objektum jellemzőjét, végül az adatbázisból is töröljük. A jellemző hozzáadáshoz megfelelően `updateDrawingElements()` itt is szerepet játszik, csak ezúttal a típus kitörlött jellemzőit törli a rajzelemek jellemezői közül.

A grafikus felület a TypeFrame osztály biztosítsa a funkcióhoz. A TypeFrame inicializálásakor az `initGUI`() a párbeszédablak komponenseit, a `setupUpperPanel()`, `setupLowerPanel()`, pedig a párbeszédablak felső és alsó paneljeit gyártja le. A TypeFrame osztály két másik osztályt használ(kompozíció) névileg: TypeListPanel és a TablePanel osztályt. A vezérlő csak a TypeFrame osztályt utasítja, a párbeszédablak megváltoztatására, de az utasítások alapján változik meg a TypeListPanel és TablePanel tartalma. A TypeListPanel egy olyan JPanel komponens, ami grafikus felületet biztosít a típusok listázásához, kiválasztásához, hogy azokat módosítani lehessen. A 3.4.5 fejezetben újra felhasználásra kerül, amikor rajzelemekhez rendelünk típusokat. Az TypeListPanel `addNullType()` metódusával egy „(null)” elemet adunk a listához ami később arra használunk, hogy a rajzelem típusát beállíthassuk null-ra. A `clear()` törli a lista teljes tartalmát, kivéve a „(null)” elemet. A `getSelectedType()` visszaadja a lista kiválasztott elemét, a `remove()` törli a lista egy adott elemét. Az `add()`-el lehet a listához elemet hozzáadni. A `select()` a megadott indexnek megfelelően állítja be a kiválasztott elemet a listában. A `mouseReleased()` pedig frissíti a kiválasztott elemet, ha egy elem fölött történt az egérgomb felengedése. A

TablePanel egy olyan JPanel, ami felhasználói felületet biztosít a típus jellemzőinek kiíratásához, kiválasztásához, törléséhez táblázatos formában. Az `addRow()`-al lehet táblázathoz egy sort hozzáadni. A `setData()` metódus egyszerre több sor hozzáadását biztosítsa. A `cleartable()`-el lehet az összes sort törölni a táblázatból. A `getSelectedLines()` pedig visszaadja a kijelölt sorokat a táblázatból.

3.4.4. Jellemzők létrehozása

A típusok definiálásakor meg kell adni annak jellemzőit, amely jellemzőket majd a rajzelemek is rendelkezni fognak, ha beállítjuk azok típusát. A jellemzők létrehozásánál meg kell adni a jellemzők neveit és megadhatjuk hozzá annak mértékegységét. A jellemző létrehozásának a vezérléséért a ParameterController vezérlő osztály a felelős és a hozzá tartozó grafikus felületet a ParameterFrame írja le. A ParameterFrame kinézete a 4.9 ábrán látható.

A vezérlő inicializálásakor a ParameterFrame `addParameterActionListener`() metódusának hívásával beállítja a vezérlőt a ParameterFrame gombeseményeinek figyelésére. Ekkor készül el a párbeszédablak felülete `setupUpperAndLowerJPanel()` metódushívással. Amikor a típusokat definiáljuk és nincsen a típushoz hozzáadható paraméter, vagy amikor a „Units” menüből kiválasztjuk a „Define Parameter” menüelemet, akkor megnyílik egy párbeszédablak a jellemzők létrehozására. Ezt a ParameterFrame `createAndShowGui()` metódusának hívásával lehet elérni. Majd a ParameterController vezérlő osztályra hárul a felelősség, hogy figyelje a grafikus felület eseményeit. Ezek az események az ACTION_ADD, és az ACTION_CANCEL, amik az „Add” és a „Cancel ” gombok lenyomását kísérik. Az események bekövetkezésekor a vezérlő `actionPerformed()` metódusa kerül hívásra, ami gomblenyomásnak megfelelően meghívja az `action_add()` vagy az `action_cancel()` metódusokat. Az utóbbi eltünteti a párbeszédablakot, de a szövegmezőkbe írt szöveg nem veszik el. Az `action_add()` esetében, a ParameterFrame `getParameterName()`, és a `getParameterMeasure()` függvényei segítségével, lekéri a szövegmezőkbe írt paraméternevet, mértékegységet. Ezután a vezérlő a `isParameterAlreadyExist()` függvény hívásával megvizsgálja, hogy adott nevű jellemző létezik-e már. Amennyiben még nem létezik ilyen elnevezésű jellemző akkor meghívódik a Dao osztály megfelelő adatbázis manipulációs függvénye és az új jellemző bekerül az adatbázisba. Végül a ParameterFrame `clearText()` metódusával kitörlődik a szövegmezők tartalma. Az ACTION_ADD esemény végeztével a párbeszédablak nyitva marad az újabb események bekövetkezéséig.

3.4.5 Rajzelemekhez típus hozzárendelés

Amikor a rajzlapon vannak kijelölt rajzelemek akkor lehetőség van nekik típust definiálni, korábban a 3.4.2 fejezetben említett SelectPopUpMenu lenyíló menüvel érhető el ez a funkció. Ekkor a funkciónak megfelelő menüpontot kiválasztva egy új JFrame, a SetTypeFrame válik láthatóvá és annak eseményeinek vezérlését a SetTypeFrameController osztály a felelős.

A SetTypeFrame aktiválódásakor (amikor fókuszt kap vagy megnyitáskor) a SetTypeFrameController értesítést kap, azaz meghívódik a `windowActivated()` metódus meghívja a `updateView()` függvényt ami frissíti a típus kiválasztó listát. Erre azért van szükség, mert ha időközben pár típus törlődik vagy új típus kerül hozzáadásra, akkor ne maradjon olyan típus kiválasztva, ami már nem is létezik. A párbeszédablaktól gomblenyomással kétféle akciót kaphat el a vezérlő: ACTION_OK, és az AKTION_CANCEL akció. Ezeket az akciókat az `actionPerformed()` metódus kapja el. Az ACTION_CANCEL akció esetén meghívódik az `action_cancel()` metódus és bezáródik(dispose) a SetTypeFrame ablak, bármiféle módosítás nélkül a rajzelemen. Az ACTION_OK akció bekövetkezése esetén a vezérlő lekéri a kiválasztott rajzelemeket és leellenőrzi, van-e egyáltalán rajzelem kijelölve a SelectController `getInputs()` metódus hívásával. Ezután meghívja a SetTypeFrame-nek `getSelectedType()` metódusát, ami visszaadja a listában kiválasztott típust, ha van kiválasztva. Mivel a listában, a típusok nevei vannak tárolva szövegesen ezért a vezérlő `lookupTypeByName()` függvénye felelős azért, hogy visszaadja a típusnévnek megfelelő típus objektumot, amit az adatbázisból keres ki. A rajzelemek az adatbázisba csak akkor íródnak be ha már van hozzájuk típus rendelve ezt a vezérlő `addNewElement()` függvényhívással éri el. Amennyiben a rajzelemnek már volt típusa akkor új típus hozzárendelés esetén a régi típus jellemzői megszűnnek. Ezt a feladatot a vezérlő `clearDrawingElement()` metódusa végzi el. Mielőtt a rajzelem adatbázisba írása megtörténik, előtte a vezérlő új nevet generál neki, a rajzelem azonosítóját és a típusának nevét felhasználva.

A grafikus felület (SetTypeFrame osztály) inicializáláskor az alsó és felső JPanelt a `setupUpperAndLowerPanel()` függvényhívással gyártja le. Továbbá ugyanazt a TypeListPanel-t használja a típusválasztó komponensként, mint amelyikről a 3.4.2. fejezetben már szó volt. Az `addSetTypeFrameActionListener()` függvényhívással lehet figyelőket hozzáadni a gomblenyomást kiváltó eseményekre. Amikor aktiválódik, az előzőekhez hasonlóan `createAndShowGui()` metódust biztosítsa a vezérlőnek a megjelenítéshez. Aktivizálódáskor a `clearlist()` metódussal tudja törölni a listát a vezérlő. És az `addType()`

segítségével lehet újabb típusokat hozzáadni a listához. Ezzel a két metódussal manipulálja a lista tartalmát a vezérlő.

3.4.6 Rajzelemekhez jellemzőértékek hozzárendelése

Azok a rajzelemek amiknek már meg van határozva a típusuk, akkor a típusnak megfelelő jellemzőkhöz lehet konkrét értéket hozzáadni. A rajzelemekhez típus hozzárendeléshez hasonlóan, ez a funkció is a 3.4.2. fejezetben már tárgyalt SelectPopUpMenu lenyíló menüvel érhető el, de csak akkor ha az összes kijelölt rajzelemhez van típus hozzárendelve. Az előző fejezethez hasonlóan új ablak (JFrame) jelenik meg a SetTypeParameterFrame és ennek eseményeit, felhasználói interakciójának kezeléséért a SetTypeParameterController vezérlő osztály a felelős.

A SetTypeParameterController vezérlő osztály figyeli a SetTypeParameterFrame által definiált párbeszédablak aktivizálódását. Előző fejezethez hasonlóan, itt is meghívódik a `windowActivated()` metódus, ezúttal a SetTypeParameterController vezérlőé. Ezután `contentTest()` metódus hívásával azt vizsgáljuk, hogy az előző kiválasztáshoz képest változott-e a kiválasztott rajzelemek és azok típusa. Ez azért fontos, hogy elkerüljük a párbeszédablak fölösleges újrarajzolását. Majd meg kell vizsgálni azt, hogy a kiválasztott rajzelemek ugyanolyan típusúak-e, ezt a `typeTest()` függvény végzi el. Csak azonos típusú rajzelemekhez lehet paraméterekhez értéket adni (még akkor is ha a különböző típusok amúgy ugyanazokkal a paraméterekkel rendelkeznek), hogy elkerüljük olyan jellemzőértéket hozzáadását egy rajzelemnél, ami típusából fakadóan nem is rendelkezhetne. Amennyiben a rajzelemek típusai megegyeznek akkor az `updateView()` függvény a típus paramétereinek megfelelően beállítja a SetTypeParameterFrame grafikus felület kinézetét. Gomblenyomásra itt is kétféle akció lehetséges: ACTION_OK, és az AKTION_CANCEL akció. Ezen eseményeket az `actionPerformed()` metódus kezeli és ez alapján hívja meg az `action_ok()`, `action_cancel()` metódusokat. Az ACTION_CANCEL akció esetén ugyanaz történik mint a 3.4.5 fejezetben, azaz bezáródik (dispose) a funkcióhoz tartozó grafikus felület. A rajzelem pedig az eredeti állapotában marad. Az ACTION_OK akció esetén a helyzet bonyolultabb. A rajzelem paraméterei HashMap-ban, a grafikus felület pedig a szövegmezőket azonban ArrayList-ben tárolódnak. Mivel a HashMap-ban az elemek kivételében sorrendiség változhat ezért a paraméter nevének alapján kell kikeresni melyik szövegmező mely paraméterhez tartozik. Ezt a feladatot látja el a `searchValue()`, ami visszaadja a paraméter nevének alapján, a paraméterhez tartozó értéket a megfelelő szövegmezőből. Ezután megtörténik a módosítások feltöltése az adatbázisba és a grafikus felület bezáródik(dispose).

A SetTypeParameterContent osztály írja az adott rajzelem kiválasztást. Más szóval, mely rajzelemek kerültek kiválasztásra, azok milyen típusúak, és az adott típushoz milyen jellemzők tartoznak. A párbeszédablak aktiválódásakor az aktuális SetTypeParameterContent frissül, a fentebb említett `contentTest()` metódus hívásakor. Amennyiben az adott rajzelem kiválasztás nem változott, azaz a SetTypeParameterContent `equals()` igazzal tér vissza, tehát a kijelölt rajzelemeknél nem történt változás és nem kell az ablakot újrarajzolni. Így a párbeszédablakban a szövegmezőbe írt tartalom nem veszik el. Abban az esetben ha a kiválasztott rajzelemek halmaza bővült, vagy a típusa megváltozott, vagy a típus paramétereinek halmaza megváltozott az ablakot újra kell rajzolni és emiatt a szövegmező tartalma elveszik.

A grafikus felület (SetTypeParameterFrame osztály) inicializáláskor a `initComponents()` függvényhívással gyártja le a komponenseit. A felületnek saját speciális elrendezés kezelője van amit a SetTypeParameterLayout osztály valósít meg. Működése szinte hasonló a beépített FlowLayout-éhoz, csak egy kis módosításon esett át, hogy az elemek vertikálisan adódjanak hozzá a párbeszédablakhoz, ne horizontálisan. Az `addSetTypeParameterFrameActionListener` függvényhívással lehet eseményfigyelőt hozzáadni a gomblenyomás eseményeire. A `createAndShowGui()` megjeleníti a felhasználói felületet. Az `addParnameLabel()`, `addParValueTextField()` metódusokkal tud a vezérlő a paraméterszámnak megfelelő komponenseket adni, a felület aktivizálódásakor (a vezérlő `updateView()` függvénye használja őket). A `removeAllField()` függvény törli a fő panel tartalmát és a komponenseket tartalmazó kollekciókat. A vezérlő hívja a felület aktiváláskor , abban az esetben, ha pl. kiválasztott rajzelemek vagy azok típusa vagy azok jellemzői változnak. Ezután újra hozzá lehet adni a párbeszédablak komponenseit, az adott kiválasztott rajzelem típusának megfelelően.

3.4.7 Alkatrész katalógus készítése

Amennyiben ha már vannak típusaink, akkor alkatrész katalógus készíthető belőlük. Az alkatrész katalógus az úgy nézz ki hogy, végigmegyünk a típusokon és típusokként, kiíratjuk táblázat(ok) formájában, amelynek soraiban a típushoz tartozó rajzelemeket tüntetjük fel, oszlopaiban pedig az egyes rajzelemhez tartozó jellemzők konkrét értékeit. Az alkatrész katalógust HTML fájlba elmentjük és azt meg is jelenítsük a felhasználói felületen. A funkciót megvalósító vezérlő osztály a CatalogController, és az ehhez tartozó grafikus felület pedig a CatalogFrame osztály biztosítsa.

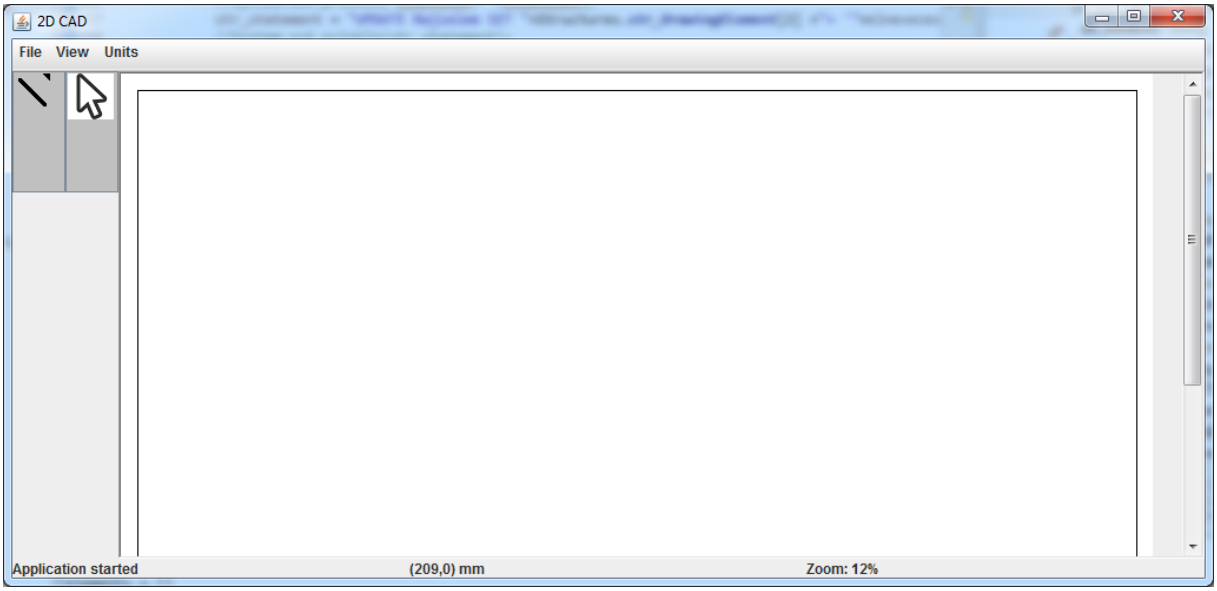
Amikor a felhasználó kiválasztja a „Create Catalog” menüelemet a „Units” menüből, akkor a `createAndShowGui()` metódussal láthatóvá válik a párbeszédablak. A

CatalogController figyeli a CatalogFrame ablak aktivizálódását. Amikor megtörténik, akkor meghívja a `saveToFile()` metódust ami meg próbálja a lokális könyvárba elmenteni a katalógust „output.html” néven. Majd meghívódik a `saveStyle()` metódus ami elmenti a weboldalhoz tartozó stílusfájlt „output.css” néven, ugyanoda. Ezután meghívja a CatalogFrame `setEditorPane()` metódusát ami betölti az előbb elmentett „output.html” dokumentumot a párbeszédablak közepére. Amikor a felhasználó az „export” gombra kattint akkor az ACTION_EXPORT azonosítójú akció hatására a vezérlő `actionPerformed()` metódusa veszi át a vezérlést, ami megnyit egy dialógusablakot az alkatrész katalógus elmentésére.

4. AZ EREDMÉNYEK BEMUTATÁSA

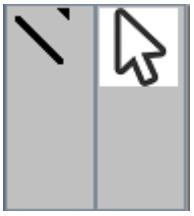
4.1. Felhasználói útmutató

A program indítása után a 4.1 ábrán látható kezdőképernyő fogad bennünket. A kezdőképernyő tetején egy menüsor található, az alkalmazás ablakának a jobb oldalán egy eszközsáv foglal helyet. Az ablak alján egy információs sáv található.

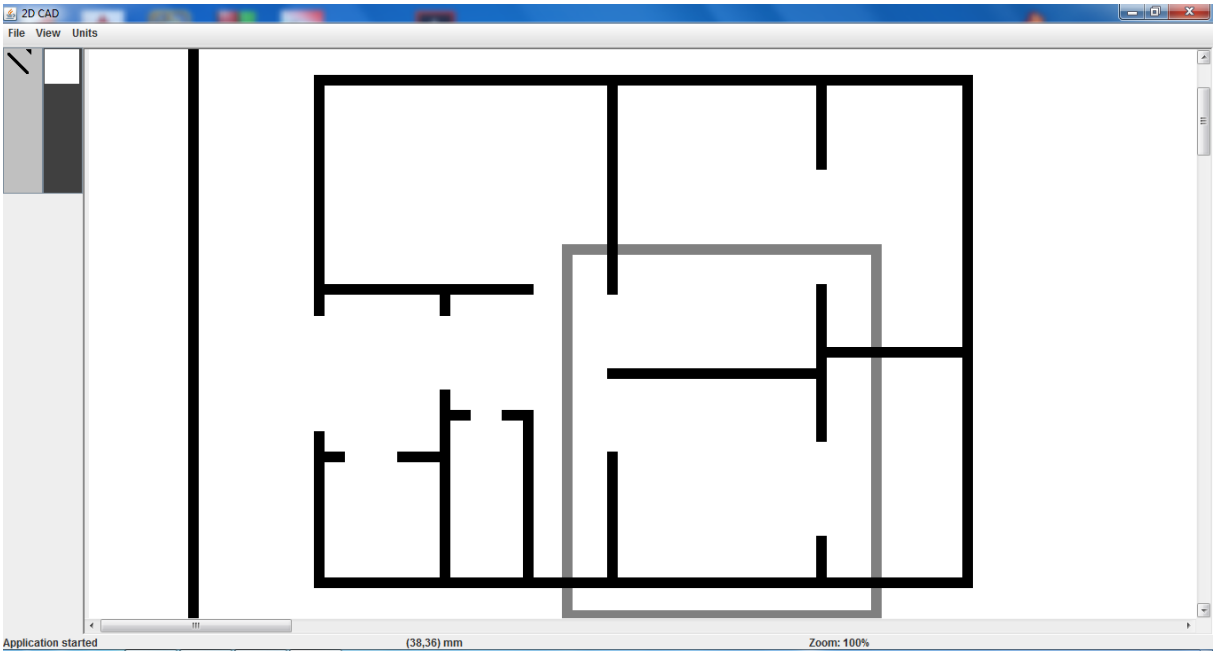


4.1 ábra. Kezdőképernyő

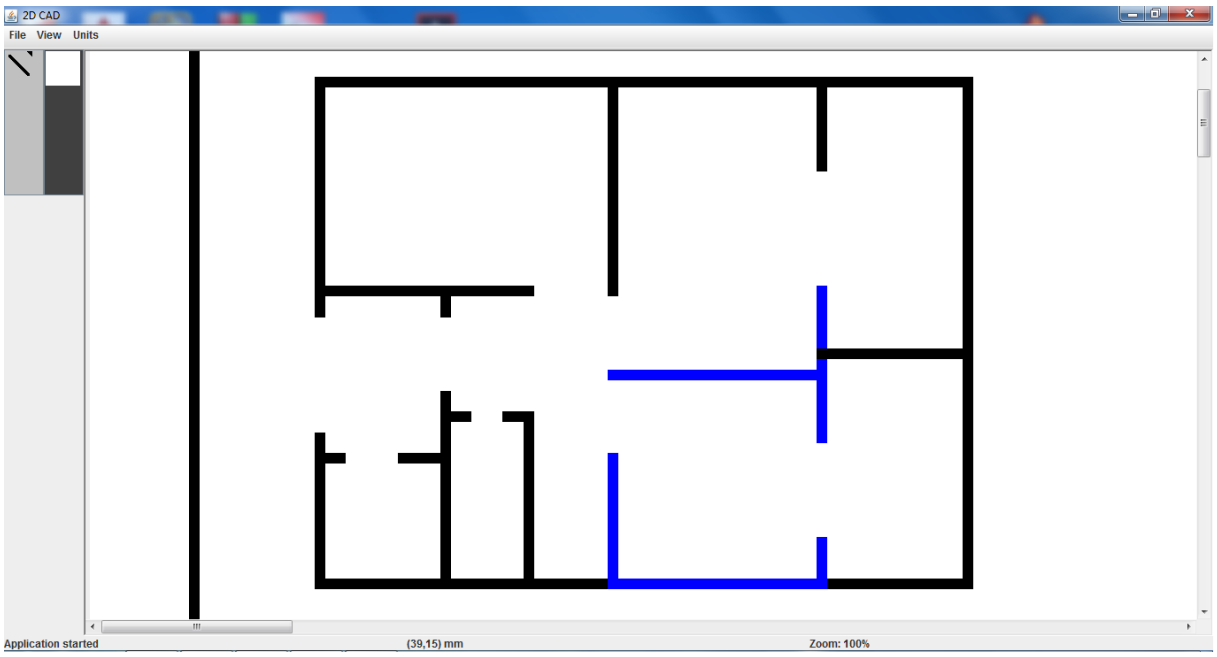
Az eszközsávunkon (4.2 ábra) található egy rajzolási és egy kijelölési funkciót biztosító nyomógomb. Kijelölés üzemmódban az egérmutató segítségével egy elemre kattintva az elem kijelöltté válik. Több elem együttes kijelöléséhez egy kijelölő téglalapot rajzolhatunk az egér mozgatásával a bal egérgombot nyomva tartva(“drag and drop” módszerrel), mely a teljesen(végpontjaival) bennfoglalt elemeket kijelöltté válik az egérgomb elengedésével. Kijelölés után a kijelölt elemek kijelölő színnel lesznek megrajzolva. Többszörös kijelölés a 4.3 és a 4.4 ábrán van szemléltetve



4.2 ábra. Bal oldalon a rajzoló üzemmód, jobb oldalon a kijelölő üzemmód.

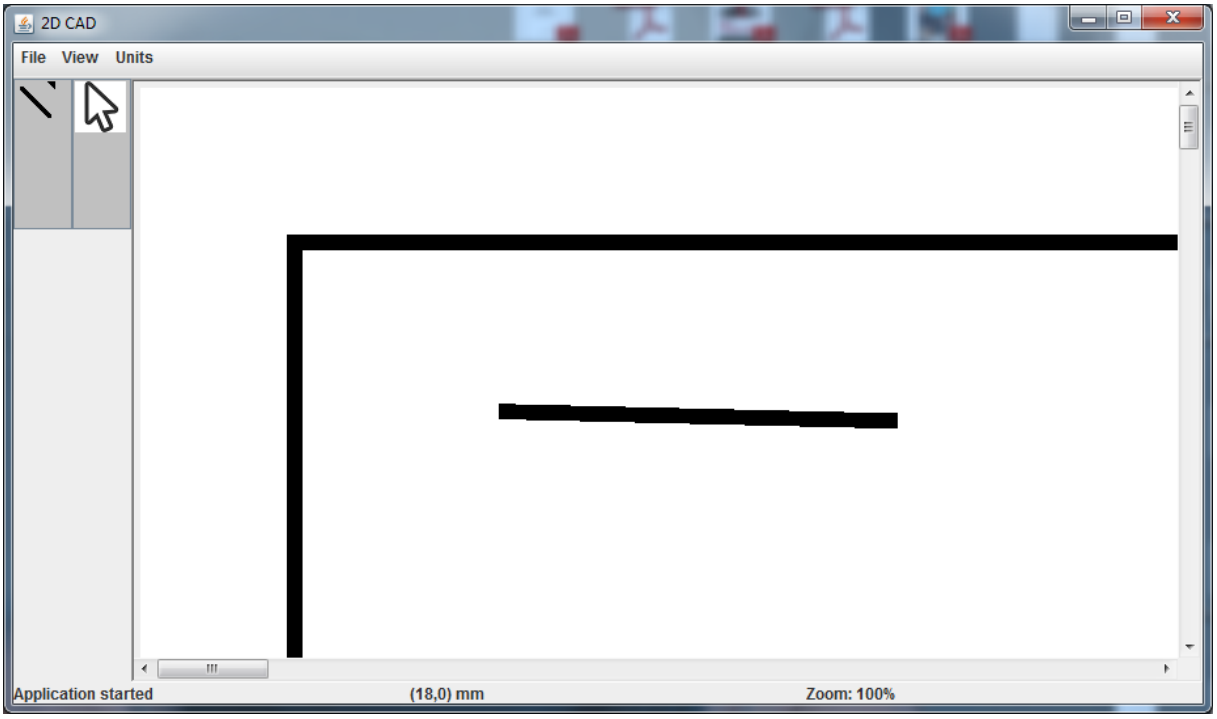


4.3 ábra. Kijelölés befoglaló négyzettel



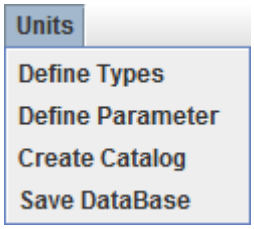
4.4 ábra. Kijelölt elemek kék színnel kiemelve

Rajzolás üzemmódban a rajzlapunk két tetszőleges pontjára kattintva egy egyenest rajzolhatunk. Azaz minden páratlan kattintással a rajzlapon, egy új egyenes szakasz kezdőpontját határozzuk meg,míg a párossal annak végpontját.



4.5 ábra. Rajzolt egyenes

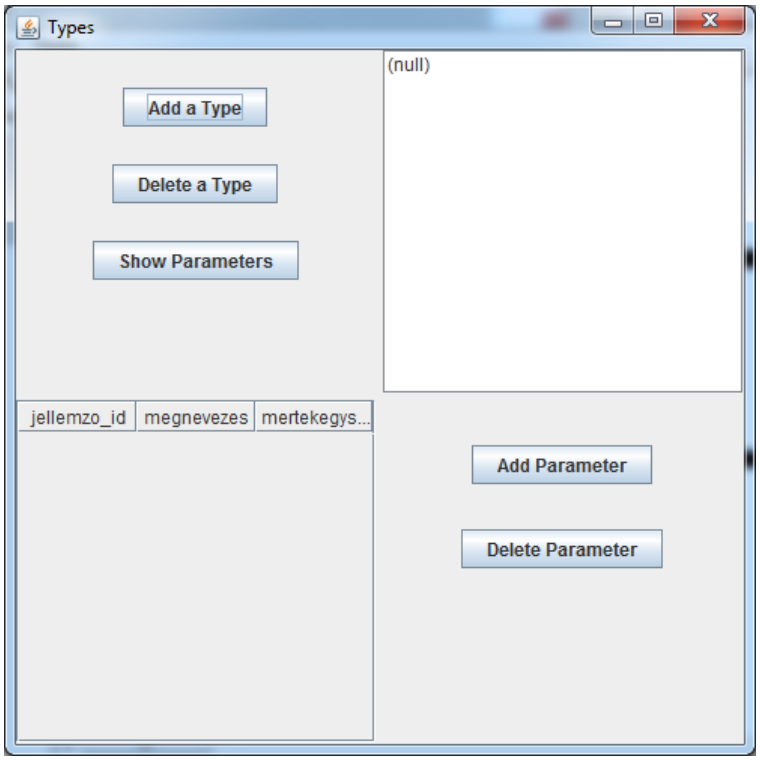
A menüsor elemei közül a „Units” menüt (4.6 ábra) lenyitva több funkciót érhetünk el. Definiálhatunk típusokat „Define Types” menüre kattintva vagy paramétereket hozhatunk létre „Define Parameter” menüpontban. Létrehozhatunk alkatrész katalógust, amit exportálhatunk HTML kiterjesztésű fájlba is a „Create Catalog” menüponttal.



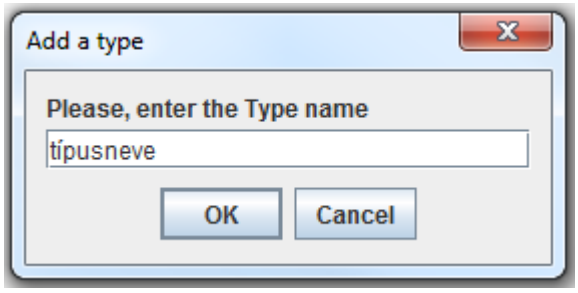
4.6 ábra. Units menü

A típus definiálása menüpontot választva a típus definiáló ablak jelenik meg. Ezt a párbeszédablak látható a 4.7 ábrán. Létrehozhatunk új típust, azonban a típus nevei egyedinek kell lennie, nem lehet típusnéveggyezés. Lehetőségünk van típus törlésére a „Delete a Type” gombra kattintva. Csak akkor törölhetjük az adott típust, ha nincs a olyan rajzelem ami a törölni kívánt típussal rendelkezik. Amennyiben csak a típus jellemzőit módosítjuk akkor már meglévő rajzelemek is módosulnak. Az „Add Parameter” gombra kattintva adhatunk hozzá jellemzőket a kiválasztott típushoz. Ilyenkor a program, csak a nem típushoz tartozó jellemzőket kínálja fel hozzáadásra. Ha nincs ilyen jellemző, akkor automatikusan a 4.9 ábrán látható paraméterkészítő ablak jelenik meg. A típusjellemzők meghatározásánál, a program föl-

ajánlja lehetőséget az adott típus jellemzőjéhez alapértéket beállítani. Amennyiben erre nincs szükségünk elég üresen hagyni a 4.11 ábrán látható szövegdobozt.

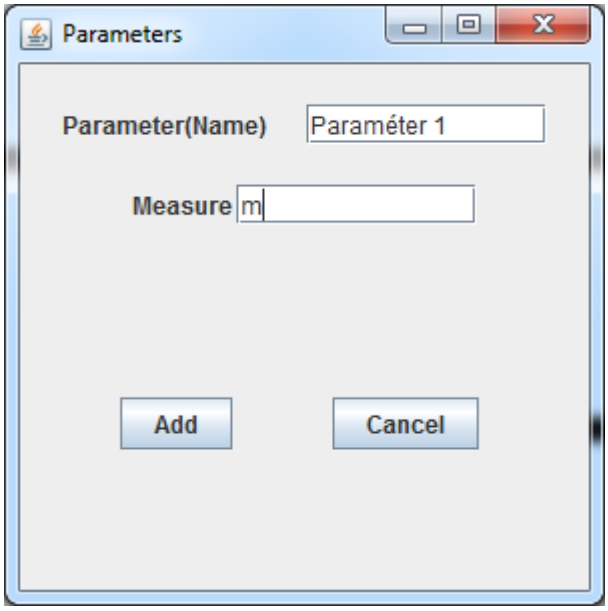


4.7 ábra. Típusok képernyő

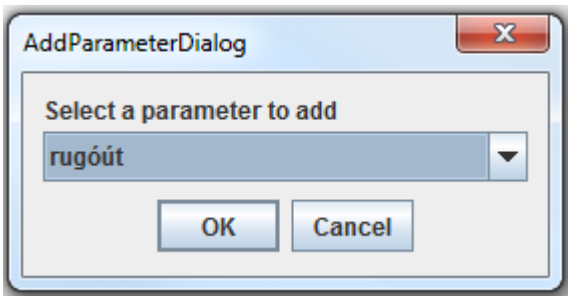


4.8 ábra. Típus nevének megadása

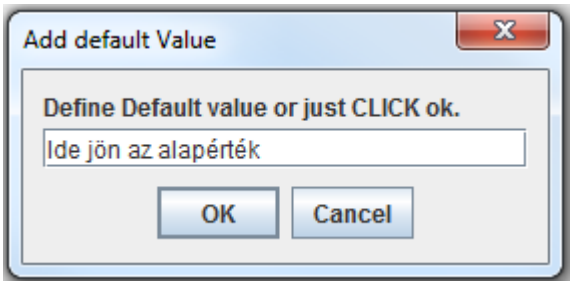
A „Define Parameter” menüpontot kiválasztva vagy típusok jellemzők hozzáadásánál a 4.9 ábrán látható képernyőre juthatunk. Ahol az „add” gomb megnyomásával, új jellemzőket adhatunk hozzá, a már meglévőkhöz. A jellemzők neve és mértékegysége állítható be. A jellemző nevének is egyedinek kell lenni.



4.9 ábra. Paraméter definiálása

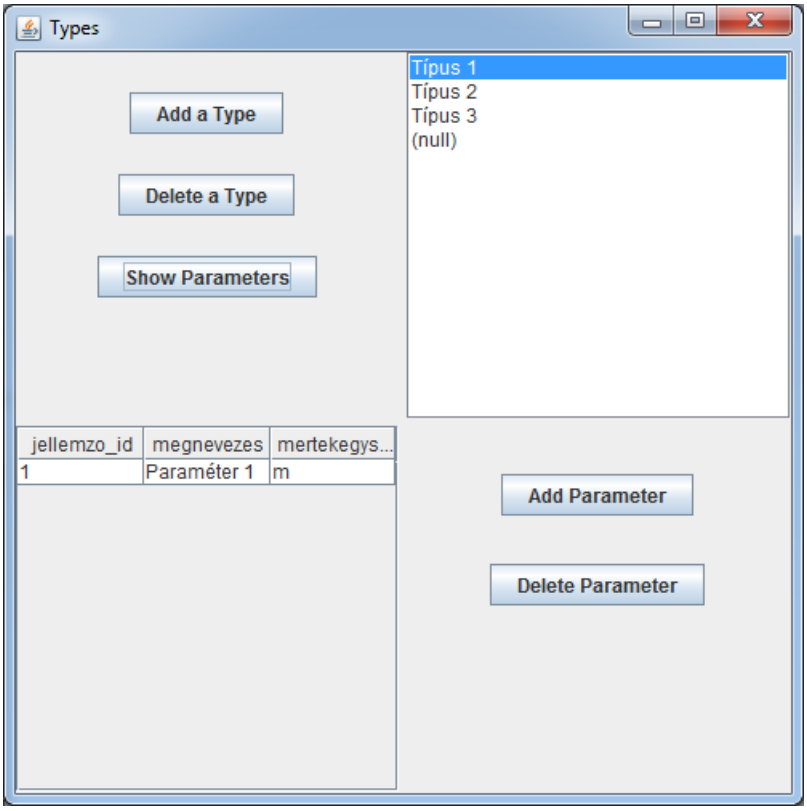


4.10 ábra. Paraméter hozzárendelése típushoz



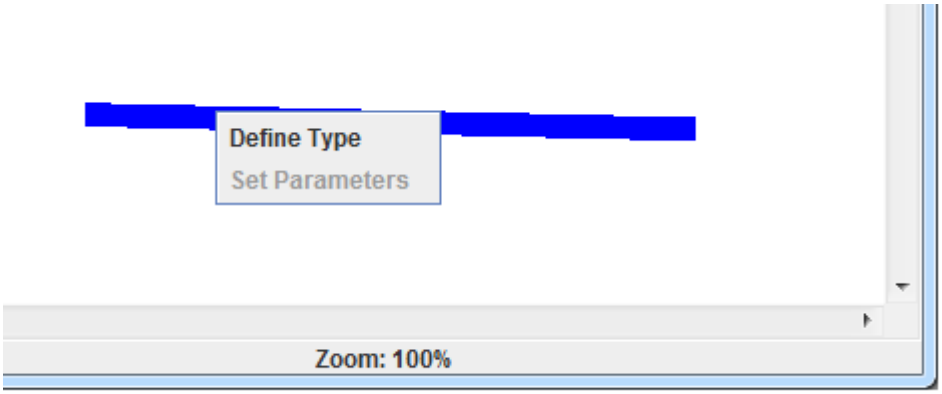
4.11 ábra. Paraméter alapértelmezett értékének beállítása

A típusok képernyőn jellemzőt adhatunk a típusainkhoz, a kívánt jellemzőt egy dialógusablakban (4.10 ábra) egy legördülő listáról választhatjuk ki. A típusok ablakunkban az adott típushoz tartozó jellemzők is listázásra kerülnek táblázat formájában. Amit a "Show Parameters" gombra kattintva jeleníthetünk meg. A típus jellemzőinek hozzáadásakor,törlésekor a táblázat frissítődik. A típus jellemző(i)nek törlése a "Delete Parameter" gombbal lehetséges. Egyszerre több jellemző is törölhető.

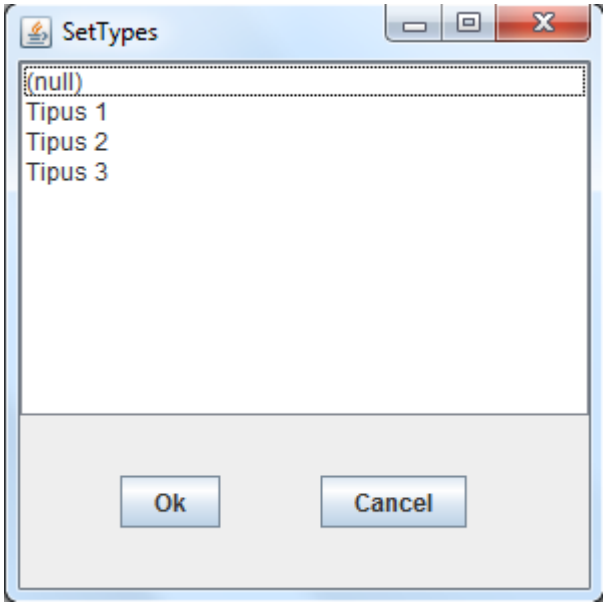


4.12 ábra. Paraméter hozzáadva típushoz

Rajzlapunkon ha van rajzelem kiválasztva, majd jobb egérgomb lenyomásával egy helyi lenyíló menüt hozhatunk elő (4.13 ábra). Ezen menü segítségével a rajzelem típusát tudjuk beállítani, a már definiált típusok közül egyet kiválasztva. Vagy lenullázhatjuk a „(null)” típust választva. A 4.14 ábrán látható a rajzelem típusbeállító párbeszédablaka. Az „ok” gombra beállítja a rajzelemek típusát a kijelölt típusra. A „cancel” gomb esetén csak becsukja a párbeszédablakot, a rajzelemek típusa ekkor nem változik.

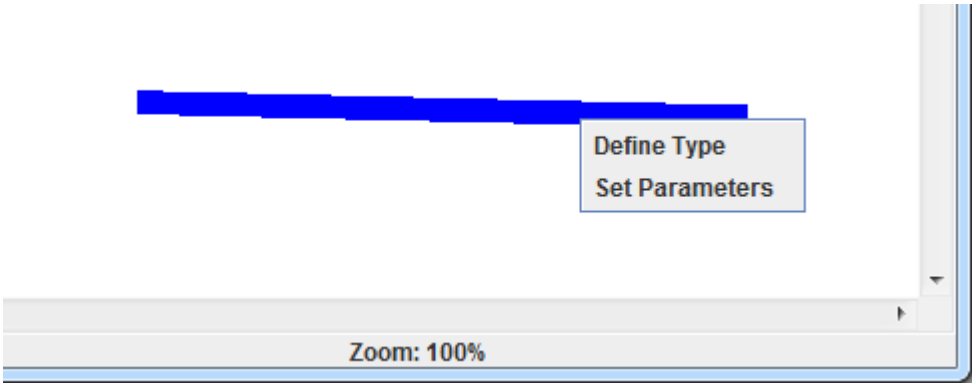


4.13 ábra. Rajzelem típusának megadása

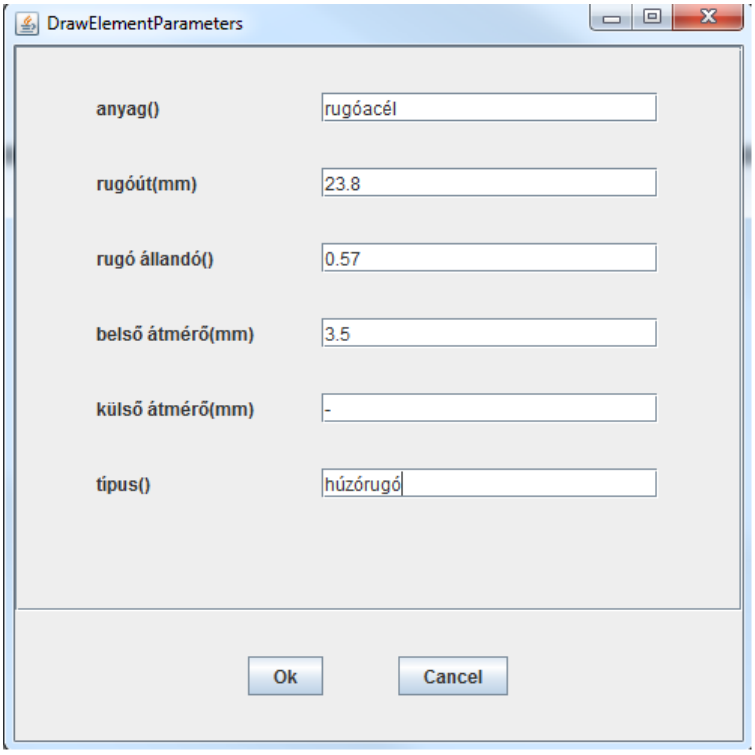


4.14 ábra. Típus kiválasztása

A típusinformációval ellátott rajzelemek jellemzőit be tudjuk állítani, az előbb bemutatott lenyíló menüben, a “Set Parameters” menüelemre kattintva (4.15 ábra). Ekkor a rajzelem jellemző értékbeállító párbeszédablaka kerül megjelenítésre ami a 4.16 ábrán látható. Egyszerre több rajzelem jellemzőit is beállíthatjuk , de csak akkor ha típusuk megegyezik. A rajzelem azon jellemzői amelyek alapértékkel rendelkeznek azok automatikusan beíródnak, de természetesen adott rajzelemek eltérhetnek ettől. A jellemzőnév mellet zárójelben a jellemző mértékegysége van.

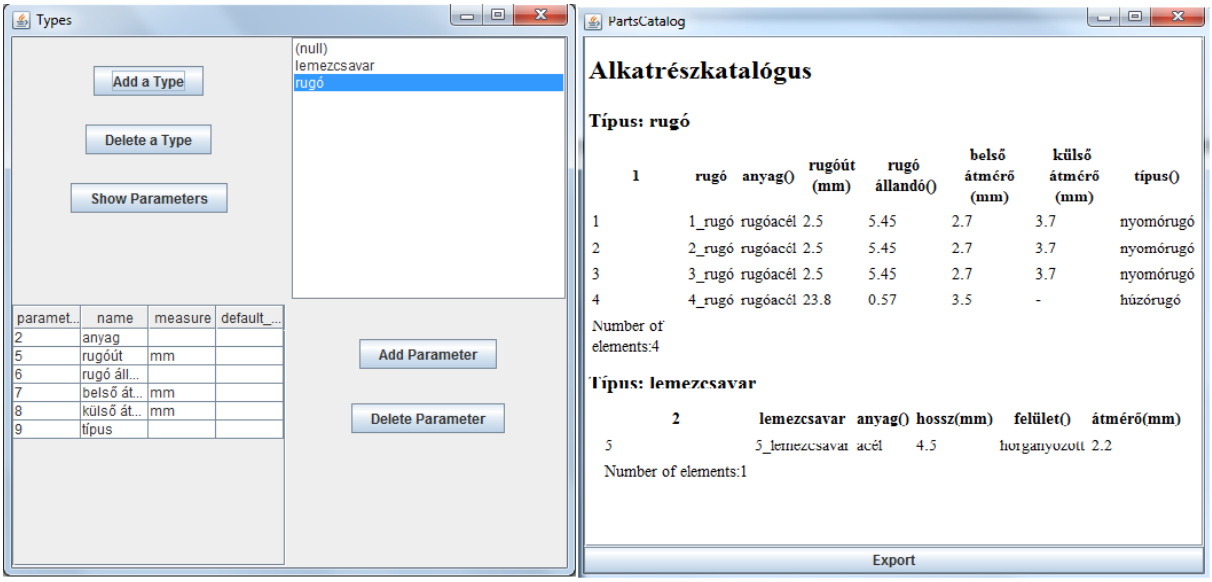


4.15 ábra. Rajzelem paramétereinek beállítása



4.16 ábra. Jellemző érték beállítása

Az Units menü “Create Catalog” menüpontja alatt tudjuk megtekinteni az alkatrész katalógusunk. Itt lehetőségünk van a katalógusunk exportálni HTML fájlba. A HTML fájl megnyitása esetén a böngészőben a karakterkódolást célszerű Unicode-ra állítani. Az alkatrész katalógus típusonként külön táblázatba írja ki az adott típusú rajzelemeket. A táblázat fejlécében az első oszlop a típus adatbázis béli azonosítója. A második oszlop a típus nevét tartalmazza. A többi oszlop a típus jellemzőit, zárójelben azok mértékegységével. A táblázat soraiban, az első oszlopban a rajzelem adatbázis béli azonosítója, másodikban a rajzelem neve. A többi oszlopban az adott rajzelem adott jellemzőkhöz tartozó konkrét értékeit tartalmazza. A 4.17 ábra jobb oldalán egy alkatrész katalógus látható két definiált típussal, és öt típussal és jellemzőkkel megadott rajzelem található. A bal oldalon a típusdefiníciós ablaka két típusról. Ugyanehhez az alkatrész katalógus HTML dokumentumáról a melléklet *B függelékében* található képernyőkét.



4.17 ábra. A bal oldalon a Típusdefiníciós ablak: lemezcsavar és a rugó típussal. Alul bal alsó sarokban a rugó jellemzőivel. Jobb oldalon az Alkatrész katalógus látható, négy rugó és egy csavar rajzelemmel.

5. Összefoglalás

A XXI. századra mérnöki tervezőmunka a CAD szoftverek elterjedésével teljesen átalakult. Míg a hagyományos papír alapú tervezőmunka esetén, a befektetett erőforrás és az idő a feladat komplexitásával hatványozottan nőtt. Addig a CAD szoftverek fejlődésével egyre több tervezői részfeladatban vette át, segítette a tervezőmérnökök munkáját. Az építőelemek definiálásában és hatékony újrafelhasználhatóságában, szimulációs és dokumentum-generáló, szabványkövető eljárásoknak köszönhetően a tervezőmunka felgyorsult. Emellett az ipari termelés, digitális technika fejlődése növekvő számú hozzáadott információ kezelését tette szükségessé. Megjelent az igény grafikus és nem grafikus adatokat egyszerre, és integráltan kezelő információs adatbázis rendszerek iránt.

Szakdolgozatomban, az volt a feladatom, hogy egy már meglévő és több modulból álló 2D műszaki rajzoló programhoz egy ahhoz illeszkedő új modult, egy adatbázis-kezelő modult készítsek, abból a célból, hogy rajzlapon lévő rajzelemekhez kiegészítő információt lehessen csatolni. Ezeket az információkat a programmal együtt memóriában futó, adatbázisban tárolni. A modult leíró rendszert az SSADM rendszertervezési módszertan segítségével terveztem meg. Az adatbázis tervezésénél azt vettem figyelembe, milyen módon kell a rajzelemekhez csatolt információt strukturálni. Ez alapján készült el az adatbázis relációsémája.

Az implementációnál az alapprogramhoz hűen az MVC szerkezeti mintát követtem. az alapprogramban, vonal rajzelem rajzoló funkció működött. Mielőtt a rajzelemekhez hozzá lehetne kiegészítő információkat rendelni, valamilyen módon meg kell határozni rajzelemek azon halmazát amelyeken ezt el akarjuk végezni. Ezért szükséges feltétel ehhez, hogy a rajzlapon lévő rajzelemeket valahogy ki lehessen jelölni. Kétféle kijelölést valósítottam meg, egy egyszeres kijelölést, és egy „drag and drop” módszerével meghatározott, téglalapnyi terület alapú többszörös kijelölést. Így egyenként és többszörösen lehet a rajzelemekhez hozzárendelni a kiegészítő információkat azaz meghatározni a rajzelemek típusát és megadni a jellemzőkhöz a konkrét rajzelemre vonatkozó jellemzőértékeket. Ahhoz, hogy a rajzelemekhez típust lehessen rendelni, előtte azokat a típusokat definiálni kell. A típusok definiálásakor meg kell adni a nevét a típusnak, és azt milyen jellemzők írják le azt a típust jól. Mivel a valódi életben bizonyos típusok, bizonyos jellemzői mindig vagy sokszor ugyanazt az értéket veszik fel, ezért lehetőség van rá, hogy ezen jellemzőkhöz alapértéket határozzunk meg. Végül, amikor egy műszaki rajz készen van, akkor lehetőség van annak rajzelemeinek (alkatrészeinek) a valamilyen rendszer szerinti listázásához. Ezért azok a rajzelemek és a kiegészítő információkból alkatrész katalógus készíthető.

Irodalomjegyzék

1. Bana István: ***Az SSADM rendszerszervezési módszertan.*** LSI Oktatóközpont. 1995
2. Eric Freeman - Elisabeth Robson: ***Head First Design Patterns***. O'Reilly Media. 2004
3. HSQLDB felhasználói útmutató. Megtalálható: http://hsqldb.org/doc/2.0/guide/index.html

Nyilatkozat

Alulírott Godó Viktor programtervező informatikus BSc szakos hallgató, kijelentem, hogy a dolgozatomat a Szegedi Tudományegyetem, Informatikai Tanszékcsoport Képfeldolgozás és Számítógépes Grafika Tanszékén készítettem, programtervező informatikus BSc diploma megszerzése érdekében.

Kijelentem, hogy a dolgozatot más szakon korábban nem védtem meg, saját munkám eredménye, és csak a hivatkozott forrásokat (szakirodalom, eszközök, stb.) használtam fel.

Tudomásul veszem, hogy szakdolgozatomat / diplomamunkámat a Szegedi Tudományegyetem Informatikai Tanszékcsoport könyvtárában, a helyben olvasható könyvek között helyezik el.

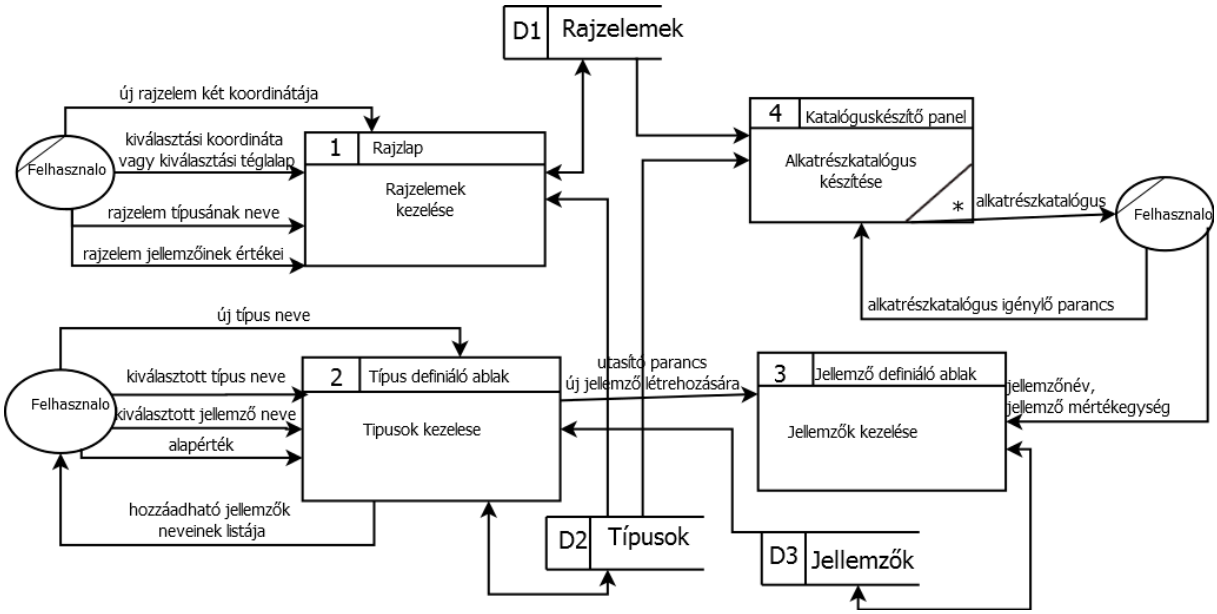
Dátum 2016. május 13. Aláírás

Köszönetnyilvánítás

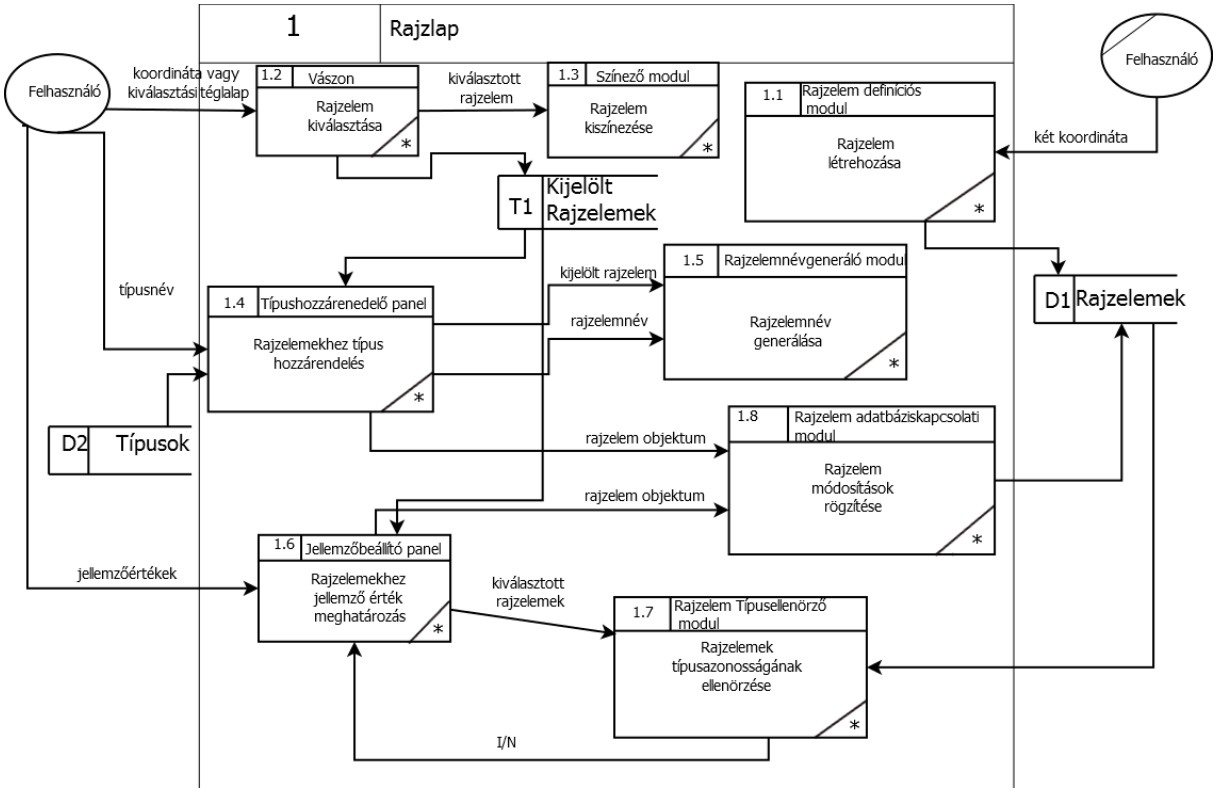
Szeretném megköszönni mindenkinek aki részt vett, vagy hozzásegített a dolgozat létrejöve-telében. Köszönöm a családnak, rokonoknak akik megteremtették a körülményeket, a nyu-godt légkört a szakdolgozat befejezéséhez. Külön köszönet Németh Gábornak, a konzulen-semnek, aki tanácsaival és segítőkészségével, rengeteget hozzátett a dolgozat megszületésé-hez. Köszönöm.

Mellékletek

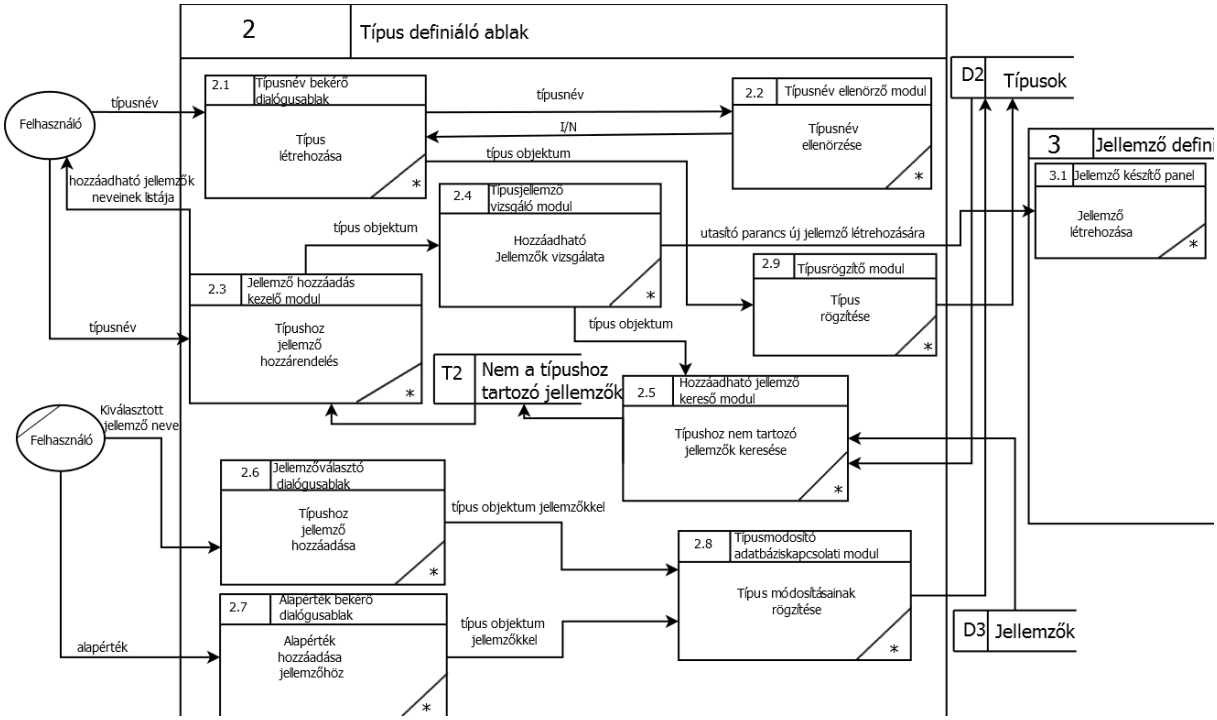
A melléklet: A tervezett rendszer fizikai adatfolyam diagramjai.



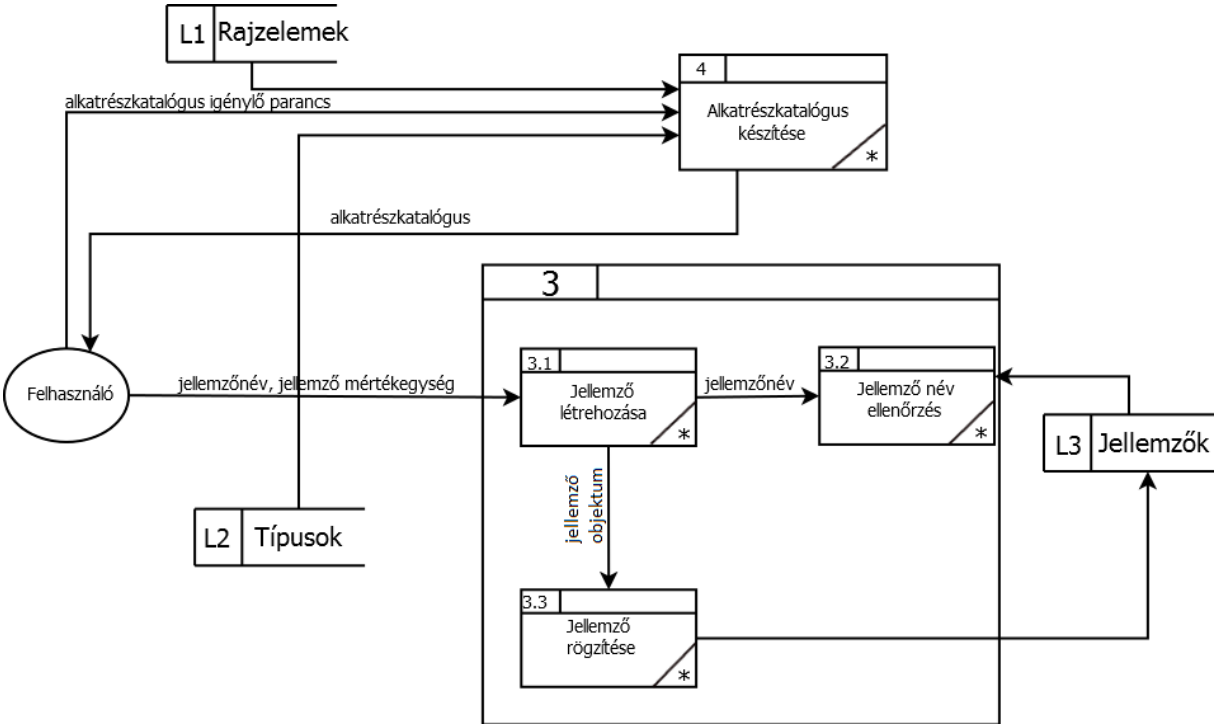
A.1 ábra: Tervezett rendszer fizikai adatfolyam diagram 1.szint.



A.2 ábra Tervezett rendszer fizikai adatfolyam diagram 2.szint 1.folyamat



A.3 ábra Tervezett rendszer fizikai adatfolyam diagram 2.szint 2.folyamat



A.4 ábra Tervezett rendszer fizikai adatfolyam diagram 2.szint 3-4.folyamat

B melléklet: Képernyőképek

Alkatrészkatalógus

Típus: rugó

1	rugó	anyag()	rugóút(mm)	rugó állandó()	belső átmérő(mm)	külső átmérő(mm)	típus()
1	1_rugó	rugóacél	2.5	5.45	2.7	3.7	nyomórugó
2	2_rugó	rugóacél	2.5	5.45	2.7	3.7	nyomórugó
3	3_rugó	rugóacél	2.5	5.45	2.7	3.7	nyomórugó
4	4_rugó	rugóacél	23.8	0.57	3.5	-	húzórugó
Number of elements:4							

Típus: lemezcsavar

2	lemezcsavar	anyag()	hossz(mm)	felület()	átmérő(mm)
5	5_lemezcsavar	acél	4.5	horganyozott	2.2
Number of elements:1					

B.1 ábra Képernyőkép az alkatrész katalógus HTML oldaláról, teljes képernyős módban.