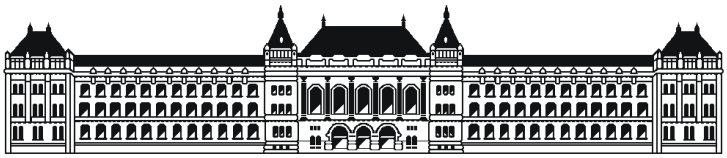


Budapesti Műszaki és Gazdaságtudományi Egyetem
Méréstechnika és Információs Rendszerek Tanszék

Informatikai modellépítés interaktív támogatása

TDK dolgozat

Készítette
Tasi Katalin

Konzulens
Dr. Pataricza András
egyetemi tanár

2011.10.28.

Tartalomjegyzék

Kivonat

Bármely informatikai rendszer kidolgozásának első lépcsője az alkalmazást befoglaló környezet megismerése és a felhasználói követelmények felmérése. Tipikusan mind a környezetről, mind pedig a követelményekről szóló információ csak a végfelhasználónál létezik, gyakran egyáltalán nem dokumentált formában. Ezen rejtett informális tudás kinyerése és egzakt megfogalmazása egyike a legnehezebb feladatoknak. A dolgozat ezért olyan módszereket keres, amelyek könnyen érthetőek azért, hogy a rendszertervező interaktívan feltárhassa az esetenként nem informatikus végfelhasználó tudását, ugyanakkor lehetőséget teremt arra, hogy a modellvezérelt tervezés által megkívánt szabatosságú, induló modell jöhessen létre.

A dolgozat a tárgyterületi szakértő végfelhasználó és a rendszertervező közötti kommunikációs szakadék áthidalására keres megoldást.

A tárgyterületi szakértő rendszertervezési szaktudás hiányában nem tud pontos, szabatos specifikációt adni a rendszertervező számára. A rendszertervező pedig tárgyterületi szaktudás hiányában nem tudja szakmailag megállapítani a rendszer helyes működésének követelményeit.

A dolgozat bemutatja, hogy a tárgyterületi szakértő tudásából milyen technológiák és eszközök használatával lehet szabatos, a rendszertervező és a tárgyterületi szakértő által egyaránt ellenőrizhető formális specifikációt kinyerni.

A tudás kinyeréshez leggyakrabban használt vizuális tudásbázis építő eszköz a gondolattérkép (mind map) és a fogalomtérkép (concept map). A grafikusan felépített tudásbázis esetén fontos szempont a formális környezetbe történő átvezetés lehetősége, ezáltal egy formalizált, tudásbázis jön létre, amelyen ellenőrizhető többek között az ellentmondásmentesség, vagy a teljesség is. A dolgozatban a formális környezetet az ontológia biztosítja.

A feladat megvalósításának fő alkotóeleme az ontológia és a szemantikus tudásbázis közötti kapcsolat meghatározása, vagyis annak a vizsgálata, hogy a grafikusan felépített tudásbázis jelentése garantáltan megegyezik-e a grafikus tudásbázisból generált ontológiában szereplő tudásbázis jelentésével.

A TDK dolgozat az informális és formális technológiák segítségével felépített tudásbázisok egyezőségét vizsgálja, valamint bemutatja a szemantikus tudásbázis és az ontológia közötti leképezés lehetőségét, és belátja a leképezés helyességét, vagyis azt, hogy az informálisan felírt tartalom jelentése azonos a formálisan felírt tartalommal.

Abstract

The first step of the workflow of any IT system design is the elaboration of the specification of its embedding environment and user requirements. Typically, only the end user is completely aware of all the information related to the environment and requirements. Frequently this fundamental information exists only in the form of an undocumented tacit knowledge. Accordingly, one of the most difficult specification tasks is the typically iterative collection, ordering and formalization of this informal knowledge.

The aim of this work is a novel method which facilitates interactive exploration of this tacit knowledge of the end user by the specification architect in an easy to understand form for validation by the end user. Simultaneously the methodology has to provide an opportunity to derive a precise initial model of the designated IT system.

The main objective of the work is a solution bridging the communication gap between specification architects and end users playing typically the role of domain experts.

The core of the problem is that end users possessing a high level of domain expertise cannot formulate a precise specification of the designated application without the help of the specification architect familiar with model formalization. Vice versa, specification architects cannot determine a set of faithful system requirements without domain expertise.

The work summarizes the candidate technologies and tools for exploration, precise ordering and formalization of the tacit knowledge of domain experts, which can be formally verified by both domain experts and specification architects.

The most commonly used visual tools for knowledge acquisition and initial ordering are mind map and concept map. These semi-formal graphical knowledge representation methods facilitate on the one hand domain expert-specification architect communication; on the other hand they form a basis for knowledge formalization. These steps have to result in a formal knowledge base where for instance completeness and consistency can be formally verified. In this thesis ontology ensure the formal environment.

The core of the problem is a semantics preserving transition from the semi-formal representation by mind or concept maps to the formalized one typically stored in form of an ontology.

The report analyses the equivalence of semi-formal and formal knowledge base representations, and demonstrates a semantics preserving mapping between them.

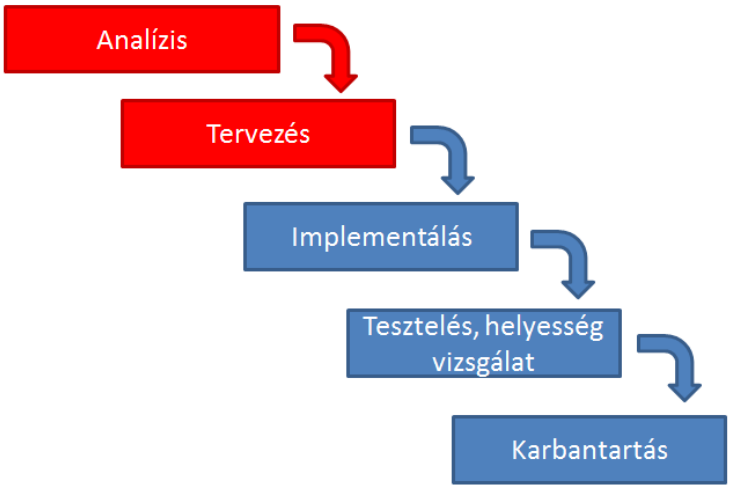
1. fejezet

Bevezető

1.1. Célkitűzés

Bármely informatikai rendszer kidolgozásának első lépcsője az alkalmazást befoglaló *környezet megismerése* és a *felhasználói követelmények felmérése*. Tipikusan mind a környezetről, mind pedig a követelményekről szóló információ csak a végfelhasználónál létezik, gyakran egyáltalán nem dokumentált formában. Ezen rejtett (tacit) informális tudás kinyerése és egzakt megfogalmazása egyike a legnehezebb feladatoknak.

A dolgozat ezért olyan módszereket keres, amelyek *könnyen érthetőek* azért, hogy a rendszertervező *interaktívan* feltárhassa az esetenként nem informatikus végfelhasználó tudását, ugyanakkor lehetőséget teremt arra, hogy a modellvezérelt tervezés által megkívánt szabatosságú, induló modell jöhessen létre.



1.1. ábra. *Vízesés modell.*

Egy rendszer tervezésének kezdeti fázisa (1.1. ábra pirossal jelölt lépései) a megrendelő és rendszertervező kommunikációján alapszik, hiszen a tervezési folyamat elején alakul ki a megvalósítandó rendszer követelményeit összefoglaló specifikáció. A tervezés során kulcsfontosságú a *hatékony kommunikáció* a megrendelő és a rendszertervező között, mert sikeres kommunikáció nélkül a rendszertervező képtelen a felhasználói igényeknek megfelelő

specifikáció és megvalósítás kialakítására.

A megrendelő és a rendszertervező között a hatékony kommunikáció általában nehezen valósul meg, mert a megrendelő, mint tárgyterületi szakértő nem rendelkezik rendszertervezői szaktudással, a rendszertervező pedig a tárgyterület szaktudásával nem rendelkezik, ezáltal hatékony, a kölcsönös megértést segítő eszköz segítsége nélkül nem képesek a megfelelő együttműködésre.

A dolgozat célja egy megoldást adni a *kommunikációs szakadék* áthidalására, így a tervezési folyamat hatékonnyá tételére. További cél a megoldás helyességének vizsgálata, azaz, annak garantálása, hogy a két fél által *informálisan felírt tartalom* jelentése azonos a rendszertervező által felírt *formális tartalommal*.

A TDK dolgozat a BSc szakdolgozat [31] elkészítése során szerzett tapasztalatokra és korai eredményekre építkezik. A TDK dolgozat elsősorban a szakdolgozatban csak ötletszerűen bevezetett formalizálási lehetőségeket fejti ki részletesen, valamint matematikai ellenőrzések segítségével alátámasztja a formalizálás helyességét. A TDK dolgozat érthetőségének érdekében néhol változatlan formában átvettem rövidebb szakaszokat, mondatokat a szakdolgozatból. A hosszabb átvett részeket, amelyek elsősorban a téma bevezetéséhez kapcsolódnak irodalmi hivatkozásként jelöltem.

1.2. Specifikációtervezés folyamata

A feladat az alkalmazásfejlesztési folyamat azon lépése, amely a *követelmények megfogalmazását*, a *környezet specifikálását*, és a *tervezés kezdeti lépéseit* foglalja magába (1.1. ábra). A specifikálásnál és a tervezésnél kiemelten szükséges a hatékony együttműködés a megrendelő és a rendszertervező között.

A specifikációtervezés előfeltétele, a tacit felhasználói tudás összegyűjtése.

- A megrendelő tudása informális, alkalmazásspecifikus, összességében és nyelvezetében létezik, ami a rendszertervező szempontjából strukturálatlan, és informatikai nyelvezetre fordítandó.
- A rendszertervező tudása formális, modellvezérelt alkalmazásokat fejleszt, képes komplex feladatok megoldására, de el kell sajátítania az alkalmazási terület alapvető ismeretanyagát és nyelvezetét.

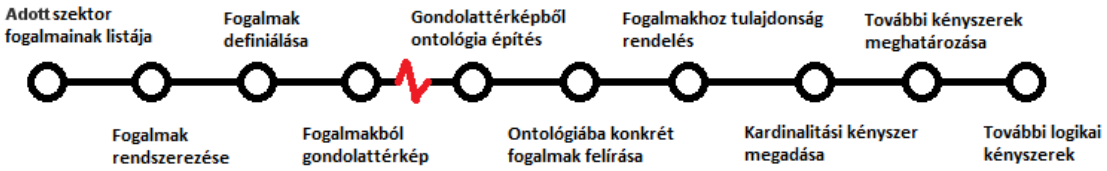
A rendszertervező számára szükséges a strukturált, szabatos tudásbázis felépítéséhez a felhasználó tudásának formalizálása. Ennek a tudásfeltárási folyamatnak azonban vannak alkalmazási területtől független általános vonásai is. A rendszertervező ezért rendelkezhet egy *általános sablonnal*, amely ezeket az általános elemeket helyezi egy sémába. A tacit tudás területei szinte minden alkalmazásnál az *archiváláshoz*, *naplózáshoz*, vagy a *biztonsági feladatokhoz* kapcsolódik.

Egy ilyen, a követelmények általános sémáját összefoglaló sablon segítségével a rendszertervező képes a felhasználót *szisztematikusan* kérdezni annak érdekében, hogy a feladatspecifikus sajátosságok kiderüljenek. Ilyen értelemben a sablon egy metamodell, és az alkalmazásspecifikus elemek annak egy példányát jelentik.

A használt sablon segítségével feltett kérdések alapján a rendszertervező képes megfogalmazni a kezdeti specifikációt, amit formális környezetbe átültetve kap egy képet a rendszerről. A formalizálás szükséges az összegyűjtött tudásbázis tartalmának vizsgálatához. A vizsgálatok segítségével ellenőrizhető a *konzisztencia*, ami biztosítja az esetleg több forrásból származó feltárt tudás *ellentmondás-mentességét*, valamint *teljességét*.

A formalizálás fontos a későbbi feldolgozhatóság szempontjából is, hisz a formális tudásbázis lehetőséget nyújt a legkorszerűbb fejlesztési paradigma egy kiinduló modelljének megalkotására is. A formalizálás során nélkülözhetetlen a *precíz*, *matematikailag helyes* gondolatmenet, hiszen a formalizálás során elkövetett hibák hatása a teljes további folyamaton végigvonul.

A tudásformalizálásnak kialakult gyakorlata van a tisztán formális modellező eszközök használata területén. Például egy ontológia felépítés tipikus folyamata az 1.2. ábrán [10] látható.



1.2. ábra. *Formalizálási folyamat.*

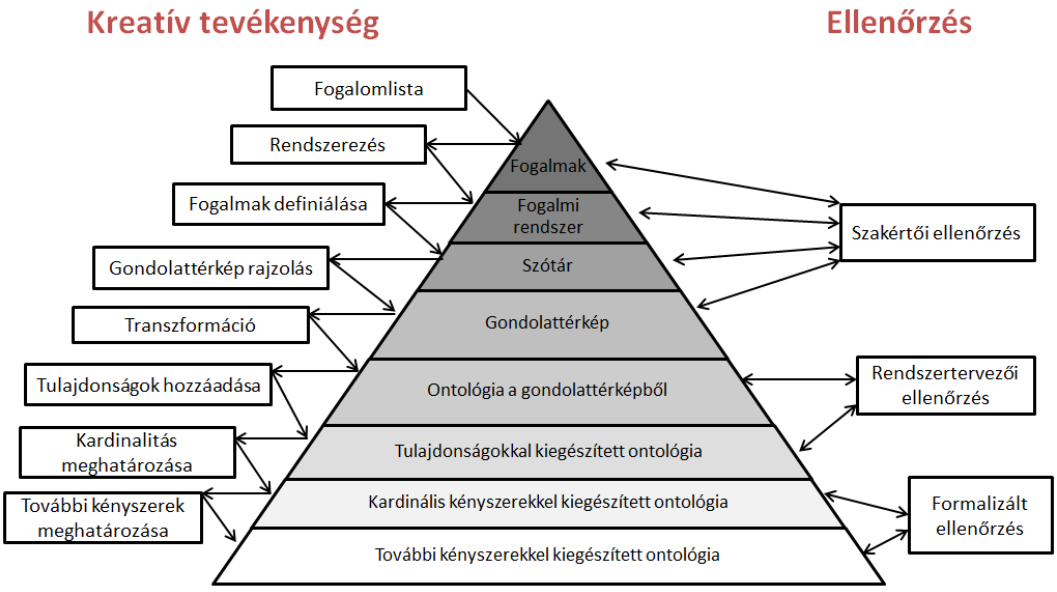
Ez az ábra azonban olyan vegytiszta környezetet tételez fel, amelyben a végfelhasználó csak *passzív információforrás* szerepét játssza, és maga még a kialakuló modellről visszacsatolást sem kap általában (természetesen annak kivételével, ha maga is képes ontológiát olvasni).

Dolgozatomban egy olyan *heterogén formalizálási és megjelenítési modell* kialakítását tűztem célul, amely kiemelt hangsúlyt helyez a felhasználóval való interakcióra, és ezért a kezdeti fázisban számít a szakterületi szakértők ellenőrző tevékenységére is. Ennek megfelelően a dolgozat a folyamat elején olyan eszközöket javasol, amelyek könnyen érthetőek, és a formalizált ábrázolást fokozatosan, de folyamatosan erősítik (1.3. ábra).

Az 1.3. ábrán [10] látható az a folyamat, amely végigvezeti, hogy hogyan lesz a fogalmak összegyűjtött, de rendszerezetlen csoportjából modellvezérelt tervezéshez szükséges formális tudásbázis. Az ábra bal oldalán a *kreatív tevékenységek* láthatóak, azaz a tudásbázis építésének lépései. A háromszögben a kreatív tevékenység elvégzése után kapott *eredményt* foglalja össze. A jobb oldalon a kapott *eredmény ellenőrzésének módját* tünteti fel. Az ábrán megjelenő színek az összegyűjtött tudás egyértelműségét ábrázolják: a fogalmak kezdeti összeírásánál a pontos jelentésük még homályos, míg a folyamat végén már kitisztult, egyértelmű tudásbázis áll rendelkezésre.

- Szakterületi fogalommodell kialakítása

  1. Az első lépés tárgyterülethez kapcsolódó *fogalmak összegyűjtése*. Ezekből a fogalmakból egy fogalomlista készül. A rendszerezetlen fogalomlista csak a fogalom



1.3. ábra. *Tervezési folyamat.*

megnevezéseket tartalmazza, nem határozza meg sem a fogalom jelentését, sem a fogalmak jelentés szerinti rendezését.

2. Az összegyűjtött *fogalmak rendszerezése* egy fogalmi rendszert eredményez. A tárgyterület kontextusában elhelyezett fogalmak alkotják a rendszerezett fogalomlistát, tehát a fogalmi rendszerben megtalálható, hogy az egyes fogalmak mely részterülethez kapcsolódnak.
3. A rendszerezett *fogalmak definiálásával* egy szótár készül, amely tartalmazza a fogalmak jelentését. A szótár annyival több a fogalmi rendszernél, hogy minden egyes fogalomhoz tartalmazza a fogalom definícióját.
4. Az elkészült *szótárból gondolattérkép* rajzolható, amely alkalmas a fogalmak közötti relációk grafikus ábrázolására.

- Formális modell kialakítása

5. A grafikus, informális tudásbázis egy *transzformáció* segítségével formális, hierarchiával rendelkező tudásbázissá alakítható. A formális tudásbázist a mi esetünkben az ontológia jelenti. Az elkészült formális tudásbázis helyességének ellenőrzése a rendszertervező feladata.
6. Az ontológiában szereplő fogalmak *tulajdonságokkal kiegészíthetők*. A struktúra helyességének ellenőrzése a rendszertervező feladata.
7. A tulajdonságokkal kiegészített ontológiához *kardinalitási kényszerek* adhatóak. Az így kiegészített ontológia helyességének ellenőrzése már automatizáltan ellenőrizhető.
8. Végül az ontológiához *további*, például adattípusokat, kizárásokat érintő *kényszerek* adhatóak, ezáltal egy teljes, formális tudásbázist kapunk a folyamat vé-

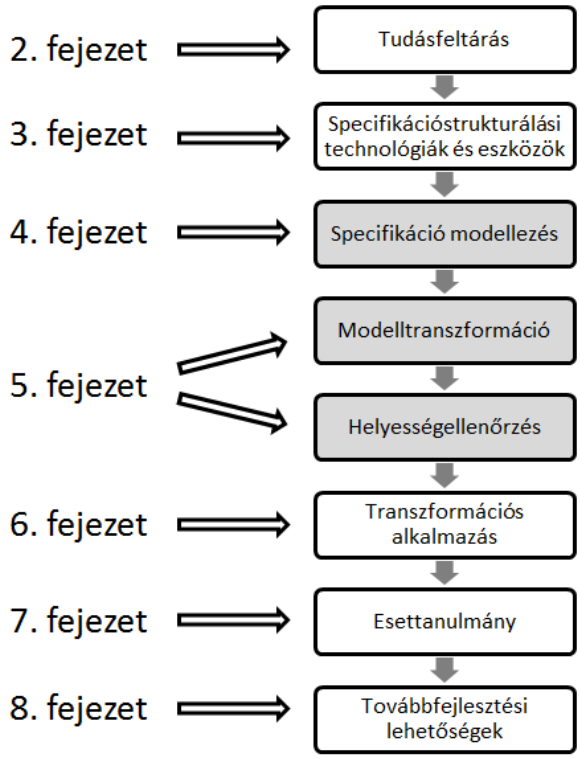
gén. A kapott tudásbázis helyességének, teljességének, és ellentmondás-mentességének ellenőrzése automatizáltan elvégezhető.

A gondolattérképig bezárólag az egyes lépések eredményének helyességének ellenőrzése a szakértő feladata. Miután a gondolattérkép képes a korábbi lépések eredményeinek önálló megjelenítésére is, ez történhet csak ez utóbbi modell vizsgálatával. Az ellenőrzés célja minden esetben annak vizsgálata, hogy a célterületi tudás teljesen és helyesen jelenik-e meg. Miután mind a modellben mind pedig az ellenőrzésben a szakterületi tudás dominál, ezért az *érthetőség*, *átláthatóság* kiemelt jelentőségű, hiszen a vizsgálatot a célterületi szakértő fogja végezni.

A folyamat végén szereplő formális tudásbázis alkalmas modellvezérelt architektúrák (MDA) tervezéséhez, hiszen egy strukturált felépítésű, szabatos tudásbázist biztosít, amely tartalmazza a *követelményeket*, a *specifikációs környezetet*, és a tudásbázisban szereplő *fogalmak közti kényszereket*.

Ezt az átmenetet támogatja például az OMG az ODM (Ontology Definition Metamodel) és a tervezésre szolgáló UML (Unified Modeling Language) harmonizált szabvány kidolgozásával.

1.3. A TDK dolgozat felépítése



1.4. ábra. *A dolgozat felépítése.*

A TDK dolgozat felépítését az 1.4. ábra szemlélteti. Az ábrán szürkével kitöltött téglalapok jelzik azokat a részeket a dolgozatban, amelyek kidolgozása jelentősen bővült a

szakdolgozat elkészítése óta.

A dolgozat második fejezete röviden ismerteti a szakdolgozat [31] elkészítése során megismert információgyűjtési módszereket, részletezi a TDK dolgozat megértéséhez szükséges és fontos részeket. Ez a szakasz elsősorban a tervezési folyamat (1.3. ábra) első lépéseivel foglalkozik, egészen a szótár elkészítéséig.

A harmadik fejezet tartalmazza az információk rendszerezésére alkalmas eszközök bemutatását. E fejezet a specifikációstrukturáláshoz szükséges technológiákat és eszközöket ismerteti.

A negyedik fejezet a specifikáció modellezésének lehetőségeit mutatja be.

Az ötödik fejezet az informális tudás formalizálási folyamatát, a kiválasztott alkalmazás metamodelljét, és a metamodell segítségével a formalizáláshoz szükséges szókészlet megfeleltetést mutatja be.

Az hatodik fejezet a dolgozatban bemutatott formalizálási folyamatnak egy implementálását mutatja be.

Az hetedik fejezetben található esettanulmány a rendszer működését mutatja be. Az esettanulmány a BSc szakdolgozatban ösztönösen elkészített esettanulmány alapvető átdolgozása, kiegészítve a TDK dolgozat keretein belül bemutatásra kerülő formalizálási lehetőségekkel. A fejezet elemzi a kézzel elkészített megvalósítás, és az automatizált formalizálás közötti különbséget.

A dolgozat végén, a nyolcadik fejezetben a továbbfejlesztési lehetőségek, valamint a gyakorlati felhasználások szerepelnek.

2. fejezet

Tudásfeltárás

2.1. Specifikációgyűjtési módszerek

A tervezendő rendszer felhasználói specifikációjának kialakítása során kritikus az, hogy az alkalmazási terület fogalomkészletéhez illeszkedjen a specifikáció (azaz feltáruljon, egy később akár metamodell formájában is formalizálható alkalmazásterület specifikus nyelvezet), majd ezzel a fogalomkészlettel épüljön fel a modell.

A szakterület-specifikus fogalomkészlet használatának előnye az, hogy ha a végfelhasználó saját természetes nyelvét használhatja a specifikáció akár interaktív elkészítése során, akkor a specifikáció *pontosabb*, és a szakterületi szakértők által *ellenőrizhető* lesz. Ezért akár interjú, akár egyéb formában fogalmazódik meg a specifikáció, a szabatos első lépés *tömörebb*, *precízebb*, és így várhatóan *kevesebb pontatlanságot* és *ellentmondást* tartalmazó specifikációt eredményez.

Minden, a specifikációt megalapozó tudás feltárása ugyanis kétféle veszélyt hord magában. A pongyolaság hátrányos következménye az, hogy a specifikáció *nem egyértelmű*, illetve *hiányos* lesz. Mind a két esetben, ha a specifikáció hibája nem derül ki az alkalmazás implementációjának megkezdéséig, annak sokszoros többletköltséget, és a tapasztalatok szerint minőségi romlást is jelent korrigálása.

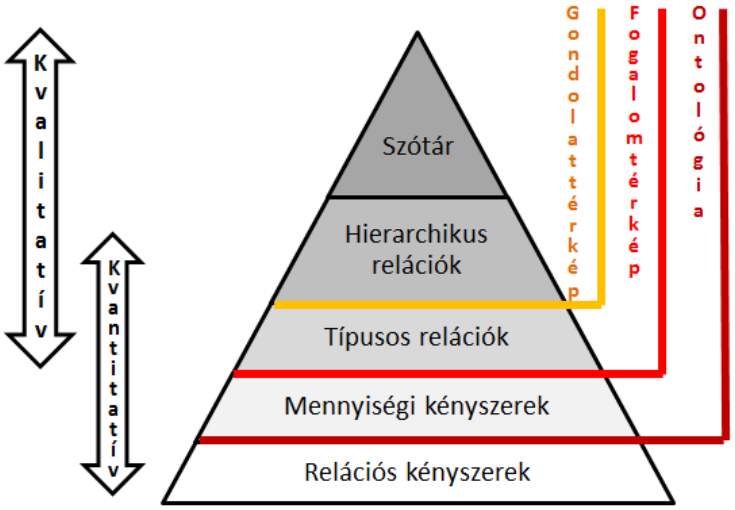
A felhasználó tacit tudásának megszerzését gyakran egy korábban bevezetett alkalmazáson szerzett tapasztalatok segítik. A korábban használt rendszer megismerése jó alapot ad a megtervezendő új rendszer specifikációjához. A korábbi specifikáció és tapasztalatok egybevetésével kiküszöbölhetőek a korábbi rendszerben szereplő hibák, pótolhatóak a hiányosságok.

A megtervezendő rendszert esetleg több, inhomogén háttér kultúrájú felhasználói csoport fogja használni, ezáltal az új rendszerrel kapcsolatban *eltérő igények* merülhetnek fel. Szisztematikusan vezetett interjútechnikák segítségével az egyes felhasználói csoportok nézőpontjai megismerhetőek, ezáltal az új rendszerrel szemben állított követelmények megállapíthatóak [31].

2.2. Tudásfeltárás

A specifikáció érdemi kialakítása a végfelhasználók, illetve a képviseletükben eljáró szakterületi szakértők feladata, ugyanakkor az informatikusnak ezt a tudást fel kell tárnia, és azt formalizálnia.

Az interjúzás segít a felhasználó tudásának megszerzésében, hiszen a feltett kérdésekre adott válaszok megadják a specifikáció összeállításához szükséges információt. Az alkalmazott modellezési nyelven megfogalmazott kitöltendő elemeket egyértelműen előíró sablont, ugyanakkor a végső implementáció általános vonásaihoz szükséges elemeket is összegyűjtve tartalmazhatja (2.2. táblázat).



2.1. ábra. *Tudásfeltárás.*

A 2.1. ábra összefoglalja, hogy a tudásfeltárás során az egyes lépésekkel hogyan bővül a tudásbázis. Az ábra bal oldalán látható, hogy a tudásfeltárás mely típusával meddig bővíthető a tudásbázis, jobb oldalon pedig az látható, hogy a használható technológiák milyen szintig képesek befogadni a tudásbázist. A tudásfeltárási típusok, és a tudásrendszerezési technológiák határai gyakran nem húzhatóak meg élesen, hisz lehet olyan terület, amelyet valamelyik technológia részben képes megvalósítani, ám a teljes lefedéshez már nem elegendő a kifejezőereje.

A tudásfeltárás során rendkívül fontos a tudásfeltárás menetének *szervezett*, *átgondolt*, és *precíz* volta. A tudásfeltárásnak alapvetően két típusa van: *kvalitatív* és *kvantitatív* tudásfeltárás.

Kvalitatív A kvalitatív interjú feladata az interjúalany tudásának összegyűjtése folyamatos szöveg formájában. Kvalitatív interjú használatával kognitívan értelmezhető eredmények születnek. Ezen eredmények feldolgozása nem egyszerű feladat, hisz a strukturálatlan, folyamatos szövegből a lényegi elemek kigyűjtése önmagában sem egyértelmű folyamat. A strukturálatlan információ források strukturálása önmagában is egy nagy tudományos terület [36], de kis és közepes méretű alkalmazásoknál ezt a részt kevéssé szokás algoritmizálni.

Kvantitatív A kvantitatív interjú feladata az interjúalany tudásának összegyűjtése előre elkészített, fix kérdőív segítségével. A fix kérdőíven jellemzően előre meghatározott válaszok szerepelnek, amelyek közül az interjúalany választhat. Az interjú végén akár automatizáltan feldolgozható, számszerű eredmények is születhetnek.

2.3. Tudásfeltárási típusok összehasonlítása

A két tudásfeltárási technika közül az első teljesen *szabad fogalomkészlet* használatát engedi meg, míg a második esetében a megvalósításhoz szükséges információ struktúra megszerzése a fő cél, de ez egyúttal *korlátozza a kérdezés spektrumát* is.

A tudásfeltárási típusok közötti alapvető különbség, hogy kvantitatív tudásfeltáráskor a tudásfeltárást végző rendszertervező az agilis szoftvertechnikákhoz hasonlóan nem csak a végső rendszer specifikációját keresi, hanem azt is, hogy hogyan kell priorizálni a feladatokat.

A tudásfeltárási módszerek jellemzőit a 2.1. táblázat foglalja össze [31].

2.1. táblázat. *Kvantitatív és kvalitatív módszer összehasonlítása.*

	Kvantitatív	**Kvalitatív**
Adat	Mennyiségi	Szöveg, kép, cselekvéssorozat kategóriák, fogalmak stb.
Feldolgozás	Automatizálható	Egyenkénti kézi feldolgozás szükséges az esetek nagy részében
Adatgyűjtés alapja	Tudásfeltárás előtt felállított kérdéssorokra adott válasz kiválasztása megadott lehetőségek közül	A tudásfeltárási előtt létrehozott kérdéssor minta alapján, kötetlen beszélgetés
Szemlélet	Merev, zárt	Rugalmas, nyitott
Tudásfeltárási stratégia	Strukturált / statikus	Strukturálatlan / folyamatos

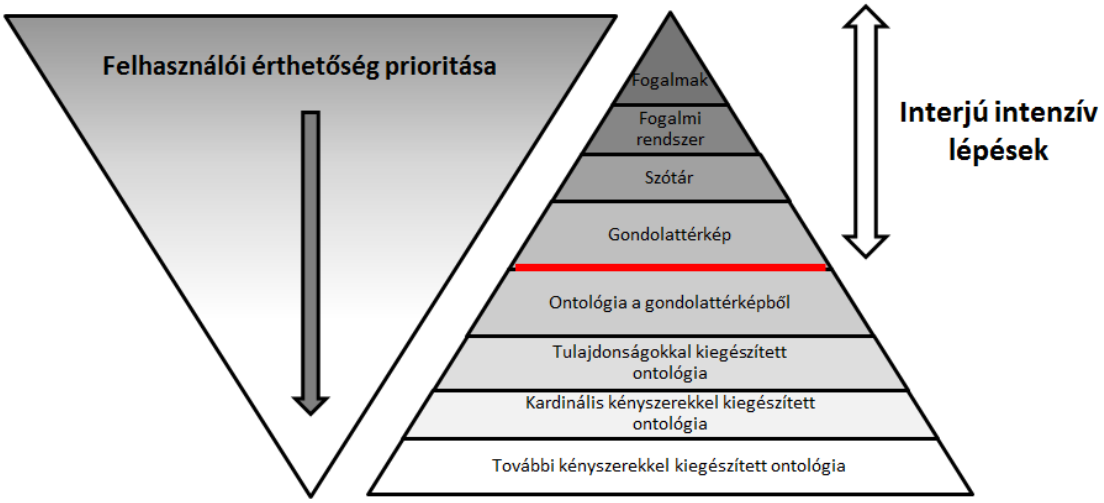
A tudásfeltárás során hasznos, ha rendelkezésre áll a tárgyterület fogalmait összefoglaló kérdés sablon, hiszen a sablon segítségével biztosítható, hogy minden fontos részlet előkerüljön a tudásfeltárás során.

A kérdés sablon kezdeti összeállítása bonyolult feladat, hiszen a szükséges tapasztalat nélkül nehéz egy olyan sablon összeállítása, ami minden lényeges területet lefed. A sablon összeállítása során nagy hangsúlyt kell fektetni a speciális esetek felderítésére, hisz a későbbi implementáció során ezen speciális esetek kezelése kritikus feladat.

A tárgyterületet érintő speciális kérdések megfogalmazásához nélkülözhetetlen a tárgyterület megismerése, és egy tárgyterületi szakértő segítsége.

2.4. Gyakorlati megvalósítás

A 2.2. ábrán a bevezetőben is ismertetett háromszög (1.3. ábra) jelenik meg, amelyen látható, hogy a tudásfeltárás kezdeti lépései (egészen a transzformációig), *interjú intenzív* lépések. Az interjú intenzív lépéseknél van kiemelt fontossága a tárgyterületi szakértővel



2.2. ábra. *Érthetőség.*

folytatott hatékony kommunikációnak, a hatékony interjúzásnak. A folyamat ezen szakaszában az interjúk alkalmával előfordulhatnak visszalépések, tehát az ebben a részben végzett tudásbázis építés még *kétirányúnak* tekinthető. Megfelelő interjúzás esetén az összegyűjtött információhalmazban szereplő hibák, hiányosságok és ellentmondások döntő része már ekkor javításra kerül.

A bal oldali háromszögábra jelöli a *felhasználói érthetőség prioritását*, tehát a folyamat legelején a legfontosabb az, hogy a felhasználó értse a folyamat lépéseit, tehát azt, hogy mely lépésben milyen tudást kell átadnia, valamint, hogy az épülő tudásbázist átlássa, képes legyen javítani. A transzformáció után már kevésbé van jelentősége annak, hogy az ontológia tartalmát, a formális tudásbázist a felhasználó megértse, hisz ennek megértése már rendszertervezői szaktudást igényel.

A TDK feladat szempontjából elsősorban a kvalitatív jellegű tudásfeltárási módszer a célravezető, mert kvalitatív jellegű interjúzás ad lehetőséget arra, hogy a megrendelő a tudásának megfelelően válaszolhasson a feltett kérdésre, hiszen kvantitatív esetben a megrendelőnek az általunk kiadott válaszok közül kellene választania, ami nem vezetne használható eredménye, ezáltal hibás specifikáció készülne el az interjú végén.

Viszont a kvantitatív tudásfeltárási módszer is szükséges a hatékony interjúzáshoz, hiszen a korábbi tapasztalatok alapján megfelelően be lehet határolni a kérdéseket, és a kérdésekre adott válaszokat egyaránt. Ez azt jelenti, hogy a rendszertervező tudása alapján szakmailag behatárolható, hogy mely kérdésterületre milyen válaszok tekinthetők elfogadhatónak, esetleg konkrét válaszlehetőség felkínálására is van mód.

A kvantitatív tudásfeltárás jellemzően *célterület specifikus*, külön kell kezelni az egyes fogalmakhoz tartozó kérdőíveket, tehát a kvantitatív tudásfeltárás főként a fogalmak definiálására, a fogalmakhoz kapcsolódó kényszerek felderítésére alkalmas.

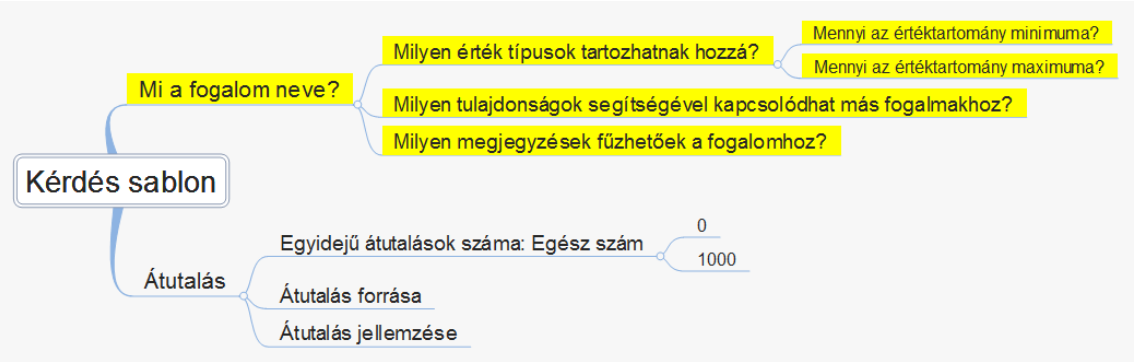
A fogalmakhoz kapcsolódó kérdőívek nem dolgozhatók fel egységesen, hisz a kérdőívek egyediek minden fogalomnál, így ezek ellenőrzése is csak részben automatizálható, viszont metrikák készítésére megfelelő adathalmazt biztosít.

A későbbi feldolgozás során a kvalitatív eredmények ontológiába történő átültetésekor szükséges annak a figyelembevétele, hogy mivel *eltérő kérdéssor* tartozik az egyes fogalmakhoz, ezért azok feldolgozása másképpen történik. A metrikák meghatározásában segíthetnek az ontológiába átültett interjú eredmények, ám nem garantált, hogy a kizárólag kvantitatív módon összegyűjtött információkból megfelelő ontológia készíthető. Fontos megszabni azokat a határokat, amíg az ontológia és a kvantitatív interjúzás együtt használható hatékonyan és a célnak megfelelően.

A teljesség igénye nélkül a 2.2. táblázat egy rövid példán bemutatja, hogy milyen kérdések feltétele szükséges akkor, ha az ontológia adott elemének kitöltése a cél. A táblázat alapján látható, hogy az ontológia teljes elemkészlete [15] lefedhető az interjú alkalmával úgy, ha a megfelelő kérdések előjönnek minden egyes fogalomnál. A teljes kérdés sablon az F.1 függelékben megtalálható.

Elemkészlet	**Kérdés**
Classes	Mik a fogalmak?
Object Properties	Milyen fogalomak közti kapcsolatok vannak?
Annotation Properties	Milyen egyéb típusú megjegyzések fűzhetőek a fogalomhoz?
Individuals	Mik a példányai a fogalomnak?
Datatypes	Mik a tudásbázisban ábrázolandó fogalmak értéktartományai?
Data Ranges	Milyen értékbeli korlátok tartoznak a fogalomhoz?
Data Properties	Milyen tulajdonságai, attribútumai vannak a fogalomnak?

2.2. táblázat. *Kérdés sablon*



2.3. ábra. *Kérdés sablon.*

A 2.3. ábrán látható egy példa a 2.2. táblázban szereplő kérdések használatára. Az ábrán felül látható a kérdés sablon egy részlete, ahol látható, hogy a kérdések általánosan megfogalmazottak, az interjút végző személynek csak egy útmutatást adnak arról, hogy milyen információkra lehet szükség az ontológia felépítéséhez. Az ábrán alul látható egy lehetséges válasz a feltett kérdésekre. A példában megadott válaszok nem teljesek, valamint

az értékhatárokat én adtam meg, a valóságban ettől eltérő számadatok szerepelhetnek.

Gyakorlati szempontból a sikeres interjú alapja az, hogy az interjúzást végző személy kellően felkészüljön abból a témakörből, amelyben az interjút el kell készítenie. Egy példával szemléltetve ez azt jelenti, hogy ha egy bank szeretne új szolgáltatást bevezetni az ügyfelei számára, akkor az interjút végző személynek tisztában kell lennie a banki szektor alapfogalmaival, a banki rendszerek alapműködésével, hiszen ennek hiányában nem tudja sikeresen elkészíteni az interjúhoz tartozó kérdéssor mintát, ezáltal fontos tudnivalók maradhatnak tisztázatlanul.

Az interjút készítő személy a szakterület megismerése után egy *kérdéssor átgondolásával* és *összeállításával* készül az interjúra. Szükséges, hogy a kérdéssor rendszertervezői, és szakterületi szempontból is megfeleljen szakmailag, ezáltal az elkészített specifikáció minden lehetőséget, tervezési szempontból kritikus témakört magába foglaljon.

A tárgyterületi szakértő terjedelmes tacit tudással rendelkezik, amelyek csak a megfelelő kérdéssor összeállítása esetén kerülnek elő. Tacit tudásra jellemző példa a működést szorosan nem befolyásoló, jogi témakört érintő területek.

Az interjú alkalmával nélkülözhetetlen a *precizitás* és a *körültekintés*, hiszen a megrendelő nem a speciális esetekre fekteti a hangsúlyt, hanem az általánosan elvárható működésre, hisz számára az a fontos, ezáltal a *speciális esetek feltérképezése* egy külön feladat. A speciális esetek ismerete, és összegyűjtése a specifikációba nagyon fontos, még akkor is, ha az a speciális eset ritkán fordul elő, hiszen a fejlesztőnek fel kell készíteni a rendszert minden eshetőségre, ezáltal lesz az elkészített rendszer működése helyes, és stabil.

3. fejezet

Specifikációstrukturálás

3.1. A strukturálás célja

A strukturálás célja a felhasználótól az információgyűjtési fázisában összegyűjtött információ *áttekinthetővé*, ezáltal különböző szempontok alapján *ellenőrizhetővé*, majd a fejlesztés további szakaszaiban *felhasználhatóvá tétele*.

- Az *átláthatóság*, *strukturáltság* és a *teljesség* tartalmi ellenőrizhetősége egyaránt javul, ha a rendelkezésre álló információ olyan tudásbázist alkot, amely a szakértő által közvetlenül is olvasható és különösebb erőfeszítés nélkül értelmezhető.
- Az informális kommunikáció egyik leghatékonyabb eszköze a *grafikus ábrázolás*, hiszen egy jó ábra bárki által értelmezhető, és akár szaktudás nélkül is megrajzolható.
- Fontos, hogy ez a tudásbázis megőrizze az interjú során feltárt fogalmakat, és a fogalmak között fennálló teljes kapcsolatrendszert, hiszen ez a fő tárgya az információgyűjtést követő szakértői ellenőrzésnek.

A szakértői ellenőrzést követően az ellenőrzött informális tudásbázist a *specifikáció szabatossága és teljessége* érdekében célszerű formális modellé transzformálni. Ezen a formális modellen matematikai ellenőrzések segítségével kimutathatóak a szakértői ellenőrzés során felderítetlenül maradt ellentmondások és hiányosságok.

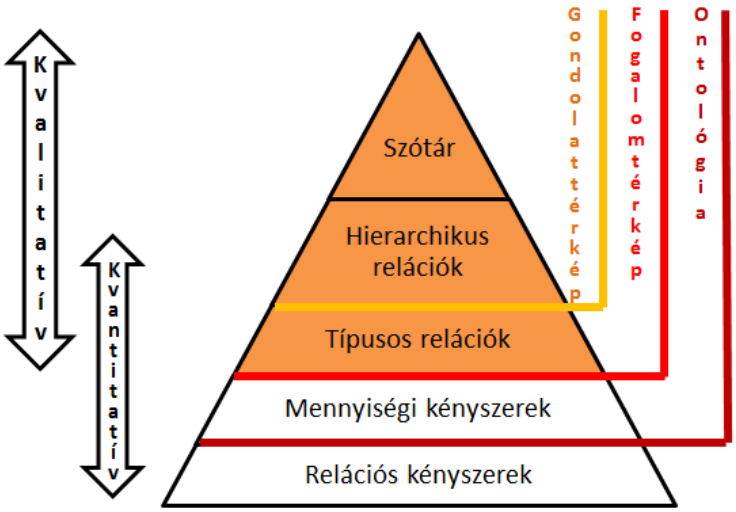
Az érthetőség és formalizáltság első ránézésre ellentmondásban van, hiszen a formalizált matematikai ábrázolás sokak számára nem közérthető. Az induló modellezés során azonban inkább a fogalmak rendszerezése dominál (mi minek a része? mi mihez kapcsolódik?), amelynek matematikai háttere tulajdonképpen a halmazelmélet és relációalgebra, amelyek a köznapi gondolkodásban is jól érzékelhetőek. A kulcskérdés ezután az, hogy a választandó modellezési eszköz mennyiben a formalizáltsághoz, illetve mennyiben a köznapi gondolkodáshoz áll közel. A következőkben ezért néhány modellezési megközelítést vázolok fel. Az ontológia szemantikailag precíz és szabványos, ezért a széleskörű gyakorlati alkalmazást is figyelembe véve ezt tekintem a formalizálás céljának.

Az érthető notációk jellegzetes példáiként az ugyancsak precíz de szűkebb szemantikával rendelkező *fogalomtérképet* (concept map, 3.6 fejezet), illetve a szabadon definiálható értelmezésű *gondolattérképet* (mind map, 3.5 fejezet) ismertetem. Különösen interjúzás közben

fontos az, hogy legalább az alapvető ellenőrzések szinte szempillantásra megtörténhessenek, ezért olyan eszközöket is kidolgoztak, amelyekben az interjúalany figyelmét könnyű egy-egy kérdésre fókuszálni, ennek leghatékonyabb eszköze a dinamikus vizualizáció, amelyet a PersonalBrain segítségével mutatok be (3.9 fejezet).

3.2. Szempontrendszer

Grafikus tudásbázis építésnél olyan technológiára és eszközre van szükség, amely felépítésben hasonló az ontológiához, tehát fogalmakkal és kapcsolatokkal dolgozik.

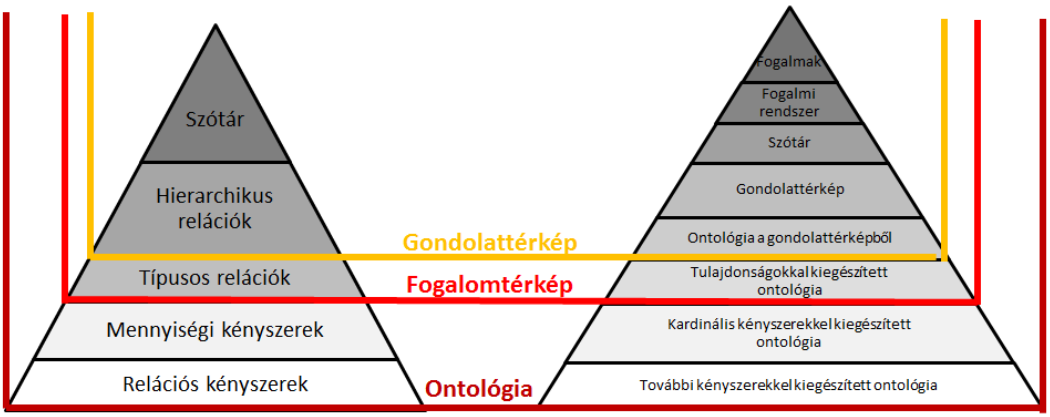


3.1. ábra. *Tudásfeltárás.*

A 3.1. ábrán látható, hogy a bemutatásra kerülő technológiák milyen viszonyban állnak egymással, valamint a tudásfeltárási típusokkal. Az ábrán narancssárgával megjelölt szakaszokat megvalósító technológiák keresésénél a fő szempontok az alábbiak voltak:

- *ontológiához hasonló felépítés*: fontos szempont, hogy fogalmakkal és relációkkal dolgozzon a technológia, hisz ezek az ontológia alapelemei,
- *eszköztámogatás*: fontos, hogy a technológia használatához létezzen grafikus szerkesztő eszköz, hisz a cél a tudásbázis prezentálása a végfelhasználók felé érthető formában,
- *eszközök kezelhetősége*: a technológiát támogató eszköz kezelhetősége fontos szempont, hisz a tudásbázis építésének sebessége kritikus egy interjú során,
- *eszközök áttekinthetősége*: az eszköz által felépített tudásbázis áttekinthetősége szükséges ahhoz, hogy a végfelhasználó képes legyen értelmezni a felépített tudásbázist,
- *eszközök által támogatott kimeneti formátumok*: formalizálás szempontjából fontos, hogy az eszköz használatával felépített tudásbázis elmenthető legyen formális kimenetbe. Formális kimenetnek tekinthető például XML.

A 3.2. ábrán összefoglalva látható, hogy mely technológiák a tudásfeltárás mely folyamatában tudnak segíteni, valamint, hogy ez a formalizálási folyamat mely lépéseit fedi le.



3.2. ábra. *Tudásfeltárási eszközök.*

3.3. Ontológia

3.3.1. Használhatóság

Az ontológia erőssége a *formális tudásbázis kezelés*ében van, hisz az informatikai feldolgozás alapja a formális tudás. A formális tudásbázis tartalmazza az adott tárgyterületnek a *fogalmait*, a *fogalmak közötti relációkat*, valamint olyan *kényszereket* is, amelyek egyaránt vonatkozhatnak fogalmakra és relációkra. Ezen relációk és kényszerek segítségével az ontológia képes *bonyolult és összetett kapcsolatrendszer* kezelésére.

Viszont az ontológiában felépített tudásbázis *nem tekinthető zárt világnak*, hisz nem garantálható, hogy minden fogalmat tartalmaz, ezáltal az sem biztosítható, hogy a tudásbázisban minden kapcsolati szabály megjelenik.

- Az ontológiában a fogalomhoz kapcsolt tulajdonság típusa lehet *mennyiségi jellemző*, amely számszerűsítve határozza meg a más fogalomhoz fűződő viszonyát, például egy fogalom pontosan egy másik megadott típusú fogalommal lehet kapcsolatban.

- A fogalomhoz kapcsolt tulajdonság típusa lehet *tartalmazási tulajdonság*, amely meghatározza, hogy az adott fogalom mely más fogalmakat foglal magában.

- A fogalomhoz kapcsolt tulajdonság típusa lehet *kizárási tulajdonság*, amely meghatározza, hogy az adott fogalom használata mely más fogalmak használatát zárja ki.

Az ontológiában felépített tudásbázisban különböző bizonyító automatákkal ellenőrizhető a tudásbázis ellentmondás-mentessége valamint a teljessége. Némely bizonyító automata képes a tudásbázisban szereplő redundanciákat is felismerni, és az egyszerűsítéseket elvégezni.

3.3.2. Modellezési metanyelv

Mind a kezdeti kommunikáció-orientált, mind pedig a későbbi fázisokban használt modellezési nyelvek közös tulajdonságai, hogy objektumokat, és a közöttük fennálló kapcsolatokat írják le. Közös gyökerük természetesen a halmazelmélet, és ezen belül kiemelten a relációalgebra. Az egyes modell leíró nyelvek ezek után elsődlegesen abban különböznek, hogy egy közös gyökérre milyen *származtatott konstrukciókat* vezetnek be.

A származtatott konstrukciók tekinthetők egyik oldalon kényelmi szolgáltatásnak is azáltal, hogy a tipikusan előforduló modellezési kérdésekhez nyelvi szintű támogatást adnak, ugyanakkor a modell tömörsége az olvashatóságot, így az ellenőrizhetőséget is jól szolgálja. A következő részben ezért elsőként az univerzálisan használható, és a matematikai reláció algebrához legközelebb eső *RDF*-et (Resource Description Framework), majd a fogalmi modellezést közvetlenebbül támogató *OWL* (Web Ontology Language) nyelvet ismertetem.

RDF

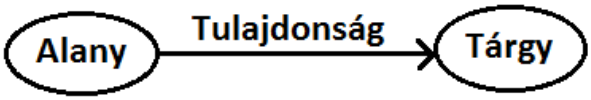
Az RDF a W3C által kiadott szabványcsalád része.

Az RDF az internetes tartalomhoz (kép, dokumentum, stb.) meta-adatok hozzárendelésével szemantikai rendezésre képes keretrendszert jelent. Az internetes erőforráshoz rendelt meta-adat XML (Extensible Markup Language) formátumban kerül tárolása, emiatt formális tudásbázist alkot [22].

Az RDF-ben tárolt adatok egyszerű kijelentő mondatok, amelyek alany - állítmány - tárgy hármasból (triplet-ből) állnak.

Az alany az erőforrás, amit az URI (Uniform Resource Identifier) egyértelműen tud azonosítani. Az alanyhoz kapcsolódik az állítmány, ami gyakran tulajdonság néven szerepel, illetve az alanyhoz kapcsolódik a tárgy is.

Az RDF-ben szereplő adatok grafikus ábrázolásra az RDF gráf nyújt lehetőséget, ahol a gráf csomópontokból és irányított élekből áll. A csomópontok közötti kapcsolatok iránya biztosítja az RDF gráf egyértelműségét. Az RDF gráfban a csomópontok jelentik a kijelentő mondat alanyát és a tárgyát, a gráf élei az állítmányt, más néven tulajdonságot (3.3. ábra).



3.3. ábra. *RDF gráf.*

Az RDF gráf az alany és a tárgy által megjelölt fogalmak viszonyát írja le, ahol a viszony típusa a tulajdonsággal van definiálva. Az RDF gráf jelentése a benne megfogalmazott állítások *ÉS* kapcsolata, ezáltal az *ÉS* kapcsolat által lesz egyértelmű az RDF gráfban ábrázolt tudásbázis.

Az RDF-ben tárolt adatok elmentésére egy XML szintaxist definiált a W3C, aminek segítségével az RDF gráf alany - tulajdonság - tárgy hármasát le lehet generálni az XML információhalmaz, és XML bázis fogalmai alapján. A W3C által megadott szintaxis a `http://www.w3.org/1999/02/22-rdf-syntax-ns#` névtérben található meg, tipikusan az

rdf prefixszel szokás helyettesíteni a névtér URL-jét. Az RDF elemeit a `http://www.w3.org/2000/01/rdf-schema#` névtér tartalmazza, amit *rdfs* prefixszel helyettesítenek [20].

Az rdfs és rdf névtérben szereplő osztályok az alábbiak [31]:

- *rdfs:Resource (Erőforrás)* Minden egyéb osztály az rdfs:Resource osztálynak az alosztálya, minden erőforrás az rdfs:Resource tagja. Minden ami RDF-el leírható, az egyben erőforrást is jelent. Maga az rdfs:Resource az rdfs:Class-nak egy egyede.

- *rdfs:Class (Osztály)* Az rdfs:Class önmagának egy egyede, az RDF osztályok osztályát jelenti. Minden RDF osztály az rdfs:Class-nak egy egyede.

- *rdfs:Literal (Literál)* Azon értékek osztálya, melyekre nem névvel történik a hivatkozás, hanem egy karakterlánccal, vagy számértékkel. A literálok lehetnek típusosak, vagy típus nélküliek, a típusos literálok típussal vannak ellátva, és az rdfs:Datatype osztály egyedei közé tartoznak.

- *rdfs:Datatype (Adattípus)* Az RDF modellben szereplő összes típus az rdfs:Datatype osztálynak az egyede. RDF modellben egész szám, lebegőpontos szám, és dátum adattípusok segítségével vannak ábrázolva. Az adattípussal való ábrázolás egy leképezést jelent, ahol a lexikális tér elemeit és az értéktér elemeit egy transzformáció segítségével megfelelteti egymásnak.

- *rdf:XMLLiteral (XML-literál)* Az rdf:XMLLiteral osztály használatával építhető XML tartalom az RDF modellbe.

- *rdf:Property (tulajdonság)* Az rdf:Property osztály az RDF által használt tulajdonságok osztálya, és az RDF modellben szereplő minden tulajdonság az rdf:Property osztálynak egy egyede. Az rdf:Property osztályban szereplő tulajdonságokkal lehet leírni a viszonyt az RDF modellben megjelenő alany szerepű csomópont, és tárgy szerepű csomópont között.

Az rdfs és rdf névtérben használt tulajdonságok az alábbiak:

- *rdfs:range (értéktartomány)* Az rdfs:range tulajdonság megadja, hogy egy tulajdonság értékei csak egy megadott osztály egyedeiből kerülhetnek ki.

- *rdfs:domain (érvényességi kör)* Az rdfs:domain tulajdonság megadja, hogy egy adott tulajdonság csak azokra az erőforrásokra alkalmazható, amely a megadott osztálynak az egyedei.

- *rdf:type (típus)* Az rdf:type tulajdonság megadja, hogy egy erőforrás mely osztálynak az egyede.

- *rdfs:subClassOf (alosztálya valaminek)* Az rdfs:subClassOf tulajdonság megadja, hogy a megadott osztály minden egyede egy másik osztálynak is az egyede.

- *rdfs:subPropertyOf (altulajdonsága valaminek)* Az rdfs:subPropertyOf meghatározza, hogy minden olyan erőforrás, amely összekapcsolható egy tulajdonság által, az egy másik tulajdonság által is összekapcsolható.

- *rdfs:label (címke)* Az rdfs:label tulajdonság segítségével egy erőforrásnak megadható az ember által is értelmezhető neve.

- *rdfs:comment (kommentár)* Az rdfs:comment tulajdonság használatával az ember számára is értelmezhető leírás fűzhető egy erőforráshoz.

OWL

Az OWL az RDF által megadott szókészletet bővíti ki, ezáltal lehetőséget nyújt az osztályok, tulajdonságok precízebb leírásához, az osztályok közötti viszony pontosabb definiálásához, és kardinalitás megadásához.

Az OWL-nek két verziója van használatban, a korábbi verzió az OWL 1, az újabb verzió pedig az OWL 2. Az OWL 2 szókészletében minden megtalálható, ami az OWL 1-ben is (előfordulnak átnevezések), és ezen túl plusz lehetőségeket biztosít.

Az OWL három alnyelvre osztható, amelyek a szókészletben, azaz kifejezőerőben különböznek egymástól.

- *OWL Lite*: Elsősorban osztályzási hierarchiaépítésre, és egyszerű korlátozásra alkalmas szókészlettel rendelkezik. Csak az OWL 1 használja.

- *OWL DL*: Maximális kifejezőereje van, ezáltal biztosítja, hogy a feltett kérdésre véges időn belül garantáltan választ ad.

- *OWL Full*: Maximális kifejezőereje van. RDF szintaktikai szabadságát biztosítja, cserébe nem garantálható, hogy minden számítás véges időn belül befejeződik.

Az OWL 2 három különböző profilt definiál [17]:

- *OWL 2 EL*: A használata azoknál az alkalmazásoknál hasznos, amelyek olyan ontológián dolgoznak, ami nagyon nagy számmal tartalmaz tulajdonságokat és/vagy osztályokat. A profil az OWL 2 olyan részhalmaza, ami képes polinomiális időben megoldani a bizonyítási problémát, és megtartja a nagyon gazdag kifejezőerejét nagy ontológia esetén is. Ennek a profilnak a rendelkezésére áll egy dedikált következtető algoritmus, amelynek megvalósítása bizonyítottan nagyon jól skálázható.

- *OWL 2 QL*: A használata azoknál az alkalmazásoknál hasznos, amelyek nagyon nagy számú példányt tartalmaznak, és ahol a lekérdezés megválaszolása a legfontosabb feladat. Az OWL 2 QL-ben összefüggő lekérdezések megválaszolása megvalósítható hagyományos adatbázis rendszerekkel. Olyan következtetési technikát használ, amely alkalmas arra, hogy az ontológia méretét is figyelembe véve logaritmikus időben válaszoljon az összetett lekérdezésekre. A kifejezőereje konceptuális modellek, mint például az UML osztálydiagram és a relációs diagram főbb jellemzőit tartalmazza.

- *OWL 2 RL*: A használata azoknál az alkalmazásoknál hasznos, ahol skálázható bizonyításra van szükség a nagy kifejezőerő feláldozása nélkül. Úgy tervezték, hogy kihasználja a nyelv teljes kifejezőerejének hatékonyságát. Az OWL 2 RL következtető rendszer megvalósítása szabály-alapú bizonyító automatákkal történik. Az ontológia konzisztenciájának, az osztálykifejezések kielégíthetőségének, példány ellenőrzésnek és egyszerű lekérdezések megválaszolásának problémái megoldhatóak polinomiális időben.

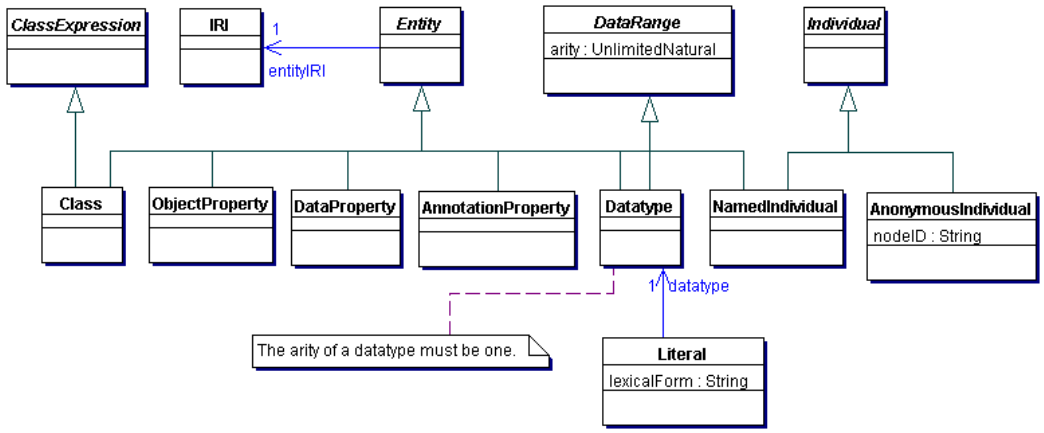
3.4. Ontológiaszerkesztő

Ontológia szerkesztésére több különböző eszköz is rendelkezésre áll, amelyeket Michael Denny egy 2004-ben készült táblázatban részletesen összefoglal [28]. A táblázat elkészítése óta valamelyest változott a rendelkezésre álló eszközök mennyisége, minősége, de táblázat jól szemlélteti, hogy az ontológia használatára és szerkesztésére mekkora igény van.

Az ontológiaszerkesztők közül a legelterjedtebb, ingyenes, java alapú szerkesztő a *Protégé* [19]. Ontológia szerkesztés során célszerű mindig a Protégé legfrissebb verziójának használata, mert az OWL 2-es támogatás régi verziókban nem biztosított. A Protégé kezelése könnyen elsajátítható, egyszerűen létrehozhatóak benne osztályok, valamint az osztályoknak gyerek osztályok.

Azonban hatékonyan használható, ellentmondásmentes, teljes ontológia megírásához szükséges ontológiai előismeret, mert megfelelő előismeret nélkül a kapcsolatok hozzáadásának és jellemzésének formális leírása hibás lesz, ezáltal a tudásbázishoz hozzáadott információk nem felelnek meg a valóságnak, és nem lehet támaszkodni az ontológiában szereplő tudásra.

A TDK dolgozat során használt OWL 2 metamodellje terjedelmes az OWL 2 szókészletének mérete miatt, ezért a 3.4. ábrán [15] az OWL 2 egy egyszerűsített metamodellje látható.



3.4. ábra. *Ontológia.*

Az ontológiaszerkesztő eszközök képességeire a formalizálási folyamat kezdeti fázisában kevésbé van szükség, a kezdeti lépések során a *felhasználói érthetőség* (usability) a legfon-

tosabb, de az ontológiaszerkesztők ezen a téren nem megfelelőek, ezért kezdtem el annak szisztematikus feltérképezésébe, hogy milyen *könnyű fajsúlyú* eszközök léteznek.

3.5. Gondolattérkép

3.5.1. Érthetőség

Tony Buzan a How To Mind Map [26] című könyvében egy grafikus (jellegzetesen tőszavas) vázlatolási módszer kialakítását tűzte célul, amely a vizuális ábrázolás segítségével könnyen érthető módon ábrázolt fogalmakat, és a köztük levő összefüggéseket.

Az egyszerű ötletét a későbbiekben pszichológusok segítségével finomították, elsősorban annak érdekében, hogy az ember vizuális átlátó képességét jól kihasználhatóvá tegyék. Széles közegben használják interjúkészítés során, ahol az interjú készítőjét a jó gondolattérkép megjelenésére és elkészítési sorrendjére vonatkozóan is a legjobb gyakorlatot rögzítő tapasztalatok segítik.

A gondolattérkép készítése *hierarchikus finomításon* alapul; kulcseleme egy, a fogalmak hierarchiáját kifejező *csillag gráf*, melynek csomópontjai a fogalmak. Minden gondolattérkép felrajzolása a központi téma megjelölésével történik, mert a központi téma köré épül a tudásbázis. A központi téma felírását a központi témához kapcsolódó kulcsfontosságú fogalmak feljegyzése követi, ezzel biztosítva, hogy minden fogalom hozzákapcsolódik a központi témához, és ebből továbbépíthető a tudásbázis.

- Az átláthatóságot segítheti (különösen az angolszász országokban), ha a szöveges megfogalmazás mellett a szöveges megfogalmazás rejtve marad, és az egyes fogalmakat *kép szimbolizálja.*
- A természetes színérzék kihasználható arra, hogy a fogalmak közötti különböző relációkat *színkódolással* fogalmazzuk meg/emeljük ki. A vizuális követhetőséget jobban segíti a *görbe vonalak* használata.

A gondolattérképet közérthetősége és átláthatósága megfelelő eszközzé teszi az informális tudás kinyerésekor annak grafikus ábrázolására.

A gondolattérképet gyakran használják az üzleti életben, mert hatékonyan lehet vele csoportosan ötleteket gyűjteni, azaz brainstorming során megfelelő ábrázolásra ad lehetőséget.

Brainstorming során a központi fogalmat feljegyezve a jelenlévők hozzáfűzik a téma kapcsán az ötleteket a megfelelő részegységhez, ezáltal többféle szempont is megjelenik egy gondolattérképen, ami az üzleti terv elkészítésekor létfontosságú. A *brainstorming* során elkészített gondolattérkép egy tudásbázist alkot, a tudásbázis megfelelő formalizálás után informatikai célokra hatékonyan felhasználható.

3.5.2. Kapcsolat az ontológiával

A gondolattérkép és az ontológia egyaránt fogalmak halmazát definiálja. A gondolattérképben megjelenő fogalmak az ontológiában megfeleltethető az osztály típusnak, mert ugyan

azt a szerepet töltik be. A gondolattérkép az ontológiához hasonlóan megadja a fogalmak közötti kapcsolatokat, de a kapcsolatok típusát már nem definiálja nyelvi szinten, azt egyedileg kell kialakítani [31]. A gondolattérkép beépíthető ontológiába, hasonlóan, mint a később ismertetendő kötés.

3.5.3. Hiányosságok

Az ontológia kifejezőereje a fogalmak közötti viszony pontos definiálásában van, hiszen pontosan megadja a kapcsolat szereplőit, és a kapcsolat tulajdonságát. A gondolattérkép eszköztára nem alkalmas a fogalmak közötti viszony definiálására, csak a kapcsolat létezését képes megjeleníteni, alapértelmezésben a csillagstruktúrában szereplő élek a fogalom finomítása (részfogalom relációt) ábrázolják. Az egyéb fogalmak bevezetése, vagy színkódolással történik, vagy a mindmap technikában bevezetett speciális, olyan jelölések segítségével, amelyek kis mértékben oldják a csillag gráf modellezési korlátait. A gondolattérképen szereplő kapcsolatok jellemzésének hiánya nem teszi lehetővé az ontológiában felírt formális tudásbázis ábrázolását gondolattérképen információvesztés nélkül [31].

3.6. Fogalomtérkép

A gondolattérkép erőssége az *egyszerűsége* és *érthetősége*, de számos elemet elsődlegesen a kapcsolatok leírásakor csak az *informális interpretáció* definiál.

A jóval precízebb fogalomtérkép (concept map) a felírt fogalmak közötti kapcsolat egyértelmű definiálását teszi lehetővé, ezáltal képes pontos leírást adni a tárgyterületről. A fogalmak közötti kapcsolat típusok definiálása lehetőséget ad bonyolult kapcsolatrendszer megjelenítésére. A fogalomtérkép megvalósítására az OMG elkészített egy szabványt Topic Maps néven.

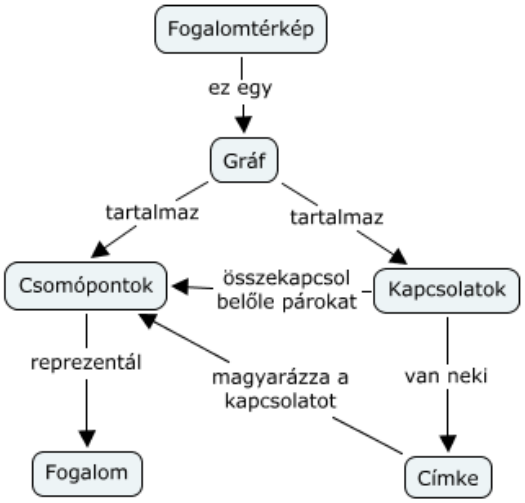
A fogalomtérkép (3.5. ábra) felrajzolása hasonlít a gondolattérkép felrajzolásához, hiszen nem igényel szaktudást, nincsen szigorú szabályokhoz kötve. A rajzolás a fogalmak felírásának, a fogalmak közötti kapcsolatok vonallal való reprezentálásának, majd a kapcsolatot ábrázoló vonalra a kapcsolat típus megjelölésének folyamata.

A fogalmak közti kapcsolat típusának adott szókészletre való korlátozásával formális tudásbázis építhető a fogalomtérképből, ezáltal alkalmassá válik *algoritmikus* feldolgozásra.

3.6.1. Kapcsolat az ontológiával

A fogalomtérkép az ontológiával azonos módon a fogalmak halmazát határozza meg, ezáltal a fogalomtérkép fogalmai leképezhetőek az ontológiában használt osztályokra, ami az ontológia fogalmait ábrázolja. A fogalomtérkép kifejezőereje a fogalmak között fennálló viszony pontos definiálási lehetőségében van, hasonlóan az ontológiához, amiben szintén a fogalmak között fennálló viszony definiálása a kifejezőerő. A kapcsolatok jellemzésének lehetősége által a fogalomtérkép részben alkalmas az ontológia grafikus ábrázolására, valamint a grafikusan ábrázolt fogalomtérkép ontológiába való átültetetésére.

A fogalomtérkép szókészlete és kifejezőereje jóval gyengébb, mint az ontológiáé, ezért



3.5. ábra. *Fogalomtérkép leírása fogalomtérképben.*

az ontológia ábrázolása csak korlátozottan lehetséges. Például egy számossági vagy tartománybeli kényszer definiálása fogalomtérképen nem egyértelmű feladat.

3.7. Korai modellezés eszközei

3.7.1. Hasonlóságok

A gondolattérkép és a fogalomtérkép alkalmas *informális tudásbázis grafikus ábrázolására*, hiszen a tudásbázis felrajzolható fogalmak és kapcsolatok halmazaként. A tudásbázis áttekinthetőségét a grafikus nyelv biztosítja a gondolattérkép és fogalomtérkép esetén.

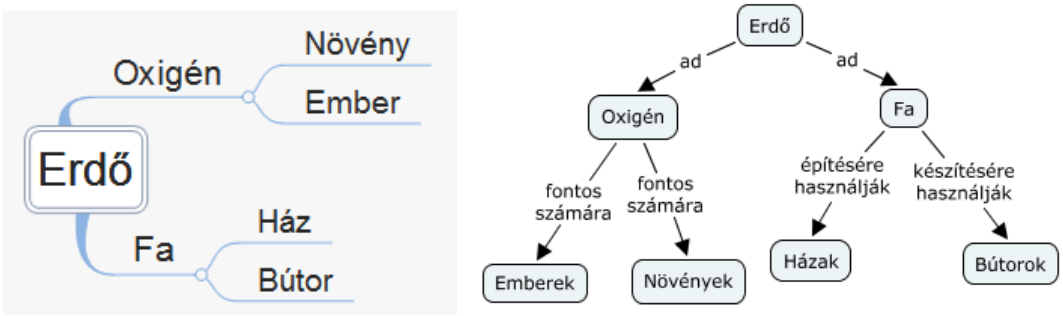
3.7.2. Különbségek

A gondolattérkép erőssége a központi témával kapcsolatos fogalmak strukturális megjelenítése, de annak meghatározására nem alkalmas, hogy jellemezze a kapcsolatot. Ezzel szemben a fogalomtérkép erőssége a fogalmak között fennálló kapcsolatok meghatározásában van, amely közelebb viszi az ontológia felépítéséhez.

A gondolattérkép és fogalomtérkép kifejezőerejében a meghatározó különbség a fogalmak közötti kapcsolat definiálásában van. A 3.6. ábra bal oldalán a gondolattérkép, jobb oldalán a fogalomtérkép által megjelenített, azonos fogalmakból álló tudásbázis látható. A gondolattérképből információ veszik el, mert nem határozható meg például az erdő és az oxigén kapcsolata. A fogalomtérképen az erdő és oxigén kapcsolatáról leolvasható, hogy az erdő oxigént ad, ezáltal megfelel az RDF hármasának felépítésével.

3.8. Megismert eszközök

A gondolattérkép és fogalomtérkép ábrázolására alkalmas grafikus nyelvet használó programok nem mindig választhatók szét egyértelműen, mert az alkalmazások egy része vegyesen



3.6. ábra. *Mind map és concept map.*

használja a két technológiát, emiatt a vizsgált eszközöket két csoportra bontottam: *online* és *offline* eszközök.

3.8.1. Online eszközök

Az online eszközök előnye a felhőben való munkavégzés, ezáltal távolról végezhető vele csoportmunka, mert grafikus megjelenés segítségével biztosítják a rendelkezésre álló jogosultságok alapján a szerveren lévő ábrák olvasását és szerkesztését.

Online eszközök esetén a vizsgálati szempontok az alábbiak:

- *Átláthatóság*: Fontos szempont, hogy az alkalmazás segítségével elkészített ábra mennyire áttekinthető, hiszen olyan alkalmazásra van szükség, amelyben a megrajzolt ábra a tárgyterületi szakértő által könnyen átlátható, ezáltal egyszerűen ellenőrizhető és javítható.

- *Kezelhetőség*: Az alkalmazás akkor használható gyorsan és hatékonyan, ha a kezelése nem bonyolult, hiszen a fogalmak felírásának, és a fogalmak közötti kapcsolatok ábrázolásának tempóját nagyban befolyásolja a munkatempót.

- *Kimenet*: A további munka szempontjából nélkülözhetetlen szempont, hogy az alkalmazás milyen formátumban képes a megrajzolt ábrát elmenteni, hiszen ahhoz, hogy formalizált tudásbázis készüljön belőle, olyan kimenetre van szükség, amely strukturált fájlszerkezetet használ a fogalmak és kapcsolatok tárolására.

Fogalom- és gondolattérkép megrajzolására sok interneten fellelhető online eszköz nyújt megoldás. A kipróbált eszközök közül a *Bubble.us*, *Glinkr* és *WriteMaps* voltak azok alkalmazások, amelyek a szempontok szerint megfelelnek a célnak. Ezek eltérően szép és használható kezelői felülettel rendelkeznek, viszont mindegyik képes XML formátumba elmenteni a benne megrajzolt ábrát.

3.8.2. Offline eszközök

Az offline eszközök feltérképezése azért szükséges, mert ezek az eszközök rendszerint nagyobb funkcionalitással bírnak az online eszközökhöz képest, ezáltal esetleg jobb használ-

hatóságot képesek biztosítani a felhasználó számára.

Offline eszközök esetén a vizsgálati szempontok hasonlóak voltak mint az online eszközöknél, ám offline esetben nagyobb jelentőséget kap a bemeneti formátumok kezelésének módja.

A kipróbált offline eszközök közül a *FreeMind*, *MindMapper* és a *PersonalBrain* voltak azok az alkalmazások, amelyek kiemelkednek valamilyen szempontból. A FreeMind és a MindMapper főként gondolattérkép rajzolására alkalmas, de a PersonalBrain-ben már megjelenik a kapcsolatok tipizálásának lehetősége. Ezek az alkalmazások kényelmesen használható kezelői felületet biztosítanak, és a legfontosabb szempontnak megfelelően képesek formális adatformátumokat kezelni.

3.8.3. Értékelés

Az eszközök és technológiák áttekintése után a 3.1. táblázatban összegyűjtöttem a szempontok alapján a technológiák jellemzőit.

	Ontológia	**Gondolattérkép**	**Fogalomtérkép**
ontológiához hasonló felépítés	-	fogalmak jellemzése hiányzik	fogalmak jellemzését biztosítja
eszköztámogatás	sok elérhető szerkesztőeszköz létezik	sok elérhető szerkesztőeszköz létezik	sok elérhető szerkesztőeszköz létezik
eszközök kezelhetősége	kényelmetlen, szaktudást igényel	bárki képes kezelni	bárki képes kezelni
eszközök áttekinthetősége	bonyolult, többablakos	egyértelmű, grafikus	egyértelmű, grafikus
eszközök által támogatott kimeneti formátumok	ontológia által támogatott formátumok	változó, de az XML jellemző	változó, de az XML jellemző
dinamikus vizualizáció	egyáltalán nem támogatott	nem jellemző	támogatott, létezik implementált megvalósítása

3.1. táblázat. *Szempontok összefoglalása*

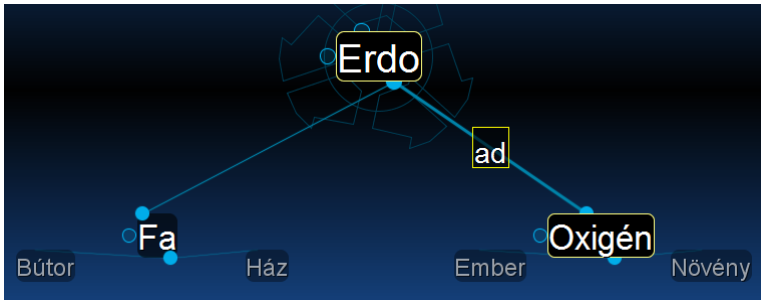
A táblázatból leolvasható, hogy a fogalomtérkép használata a megfelelő irány, hisz ez áll a legközelebb az ontológiához, valamint létezik hozzá dinamikus vizualizációt támogató eszköz.

3.9. Dinamikus vizualizáció

3.9.1. Dinamikus vizualizáció példa

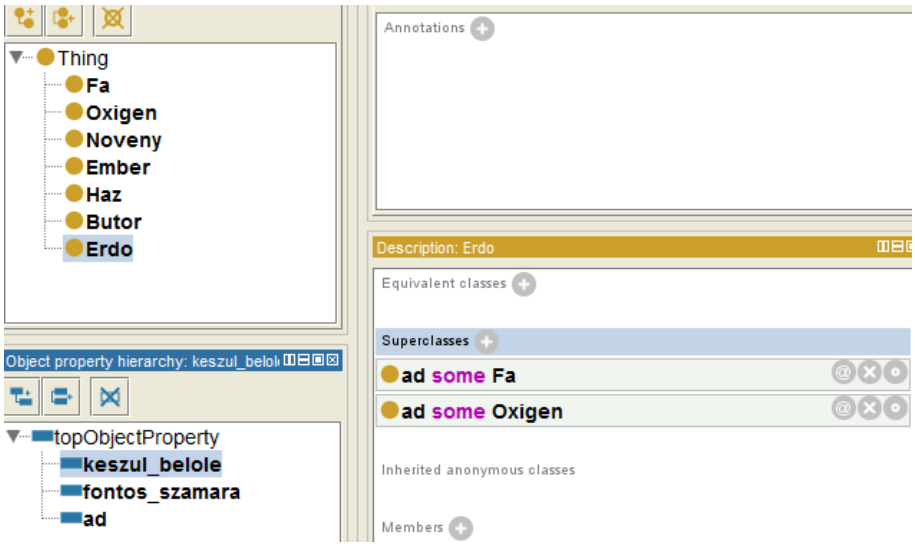
Egy példán végigvezetve a technológiák bemutatása:

A példa informális leírása a 3.7. ábrán látható, amely azt jelenti, hogy az *erdő oxigént* és *fát ad*, a *fából bútor* és *ház készül*, míg az *oxigénre szüksége van a növényeknek* és



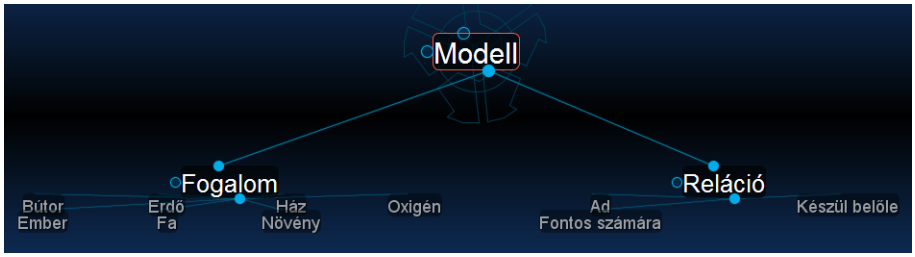
3.7. ábra. *Fogalomtérkép.*

az *embereknek* is. A példa bemutatására használt eszköz sajátossága, hogy a fogalmakat összekötő kapcsolatok típusát csak abban az esetben mutatja, ha az egér a kapcsolatot szimbolizáló vonal fölé megy.



3.8. ábra. *Ontológia.*

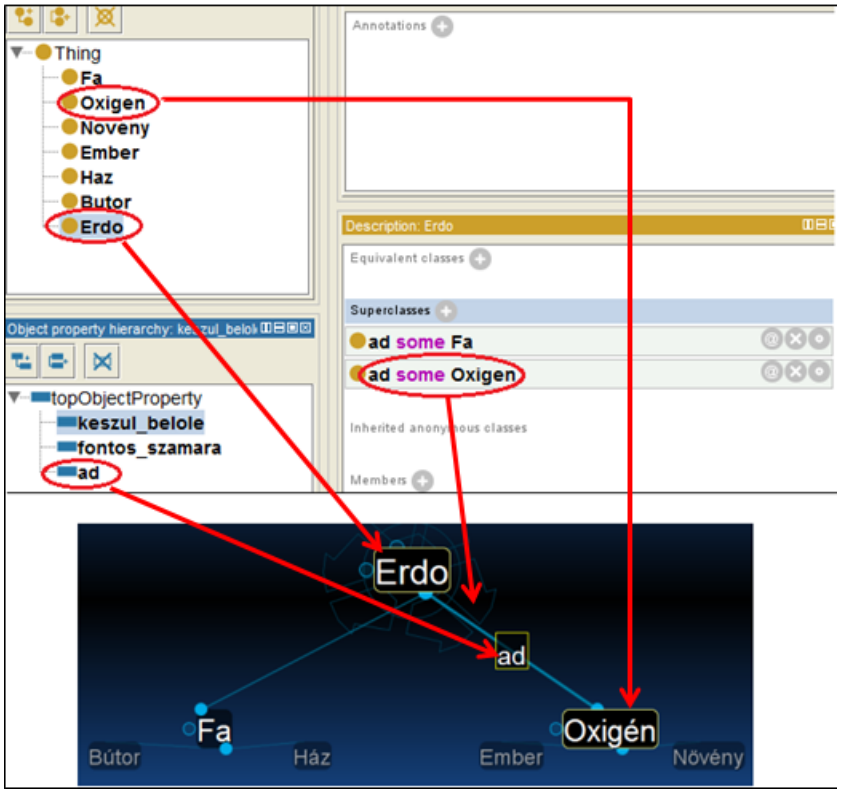
Az informális ábra felírása ontológiába a 3.8. ábrán látható módon tehető meg. Az ábrán az szerepel, hogy a bútor, ember, erdő, fa, ház, növény és az oxigén egyaránt *osztályként* jelennek meg az ontológiában, és ugyanazon a *hierarchia szinten* vannak. Az ábra bal alsó részén látható, hogy a fogalmak között háromféle kapcsolat lehet: *ad*, *fontos számára*, *készül belőle*, míg a jobb alsó ablakban leolvasható, hogy az erdő nem meghatározott számú *oxigént* és *fát* tud *adni*.



3.9. ábra. *A modell elemei.*

A példán jól látható a *Szempontrendszer* fejezet (3.2 fejezet) végén felvetett probléma,

ami az ontológia érthetőségét vizsgálja. A 3.10. képen látható, hogy az ontológiaszerkesztő



3.10. ábra. *Érthetőség.*

többablakos alkalmazás, amely túlzsúfolt módon ábrázolja a tudásbázist, amelyben olyan elemek is szerepelnek, amire a formalizálási fázis elején esetleg nincs szükség. Ugyanakkor ezek a szolgáltatások hasznosak lesznek a formalizálási folyamat későbbi lépéseiben, azonban az ontológia elemeinek egy része jól érzékeltethető egy megfelelően felépített ábrán, ezért a *grafikus megjelenítés* egy *alacsony szintű leírása* az ontológiának.

Az objectProperty-k vizualizációjának megvalósítása egyszerű feladat, nagy gond azonban, hogy hogyha minden elemet és kapcsolatot egyszerre akarunk megjeleníteni, akkor az ábra információval túlzsúfolttá válik, dacára annak, hogy a kognitív pszichológia vizsgálati eredményei szerint az emberi átlátó képesség erősen korlátos, azaz az ember egyszerre 7 dologra tud figyelni [33].

Ezzel a megközelítéssel a fogalomtérkép az információátvitelt korlátozza, viszont az egyes kérdésekre az ember nem egyszerre válaszol, valamint a kérdések feltétele sem egyszerre történik. Ugyanakkor a *dinamikus vizualizáció* segítségével megoldható, hogy egyszerre csak azok a fogalmak, kérdések legyenek láthatóak, amelyek kidolgozása folyamatban van. Ezáltal korlátozott az egyszerre megjelenő információ, de ha az egyes aspektusokhoz tartozó kérdés látható az ábrán, akkor az ábra átrendezhető, így megoldható az is, hogy a megfelelő aspektusra válaszoljon a felhasználó. *Ezt nevezik dinamikus vizualizációnak.*

A 3.10. ábrán látható, hogy az ontológiában felépített tudásbázis ugyan azt jelenti, mint a PersonalBrain-ben felépített tudásbázis, de a PersonalBrain esetén az ábra önmagát magyarázza, ez a dinamikus megjelenítés előnye, tehát nincs az ábrán fölösleges és zavaró

információ.

A példában szereplő, ontológiába átvitt fogalmak szerepük szerint összegyűjtve a 3.9. ábrán láthatóak. A 3.7. és 3.9. ábrákat összevetve leolvasható egy egyszerű megfeleltetési szabály, amely azt jelenti, hogy

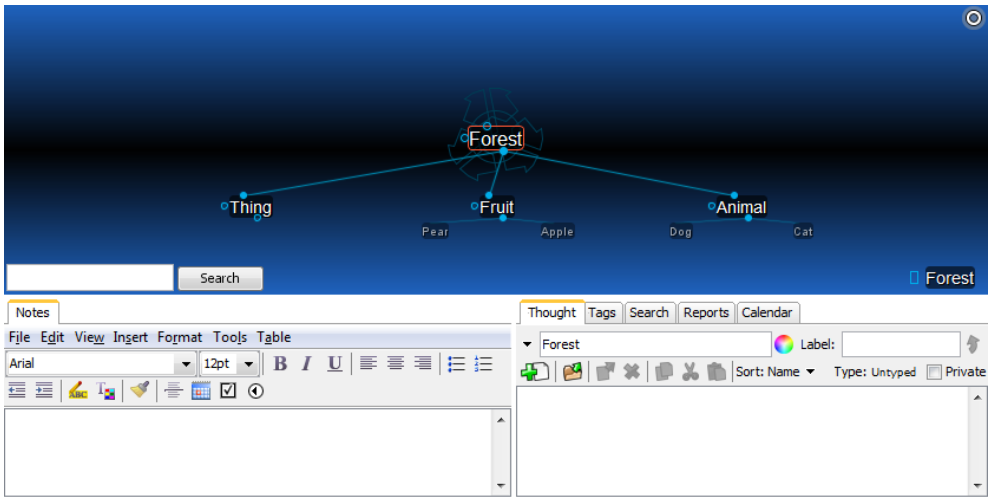
- a fogalomtérképben szereplő *fogalmak osztályként* jelennek meg az ontológiában,
- a fogalomtérképen szereplő *relációk objectProperty*-ként jelennek meg az ontológiában.

A példa felépítéséből egyértelműen látható, hogy a fogalomtérkép felépítése nagyon hasonlít az RDF-ben használt hármasokra (alany-állítmány-tárgy). Az OWL az RDF-ben használt fogalmakat egészíti ki, valamint a kifejezőerejét bővíti, ebből következik, hogy a fogalomtérkép transzformálható OWL-be.

Ellenben már ezen a rövid példán is látható, hogy a fogalomtérkép önmagában nem képes az OWL teljes elemkészletét lefedni, hiszen már ebben a példában is megjelenik a számosság (*ad **some** oxigén*), amit az ontológia felépítése során szükséges megadni, ám a fogalomtérkép nem definiálja pontosan, hogy melyik kapcsolat milyen *számossági kényszert* takar.

3.9.2. Dinamikus vizualizáció megvalósítás

A megismert eszközök közül egyedül a PersonalBrain [18] képes kezelni a dinamikus vizualizációt. A funkciói megismerése alapján alkalmas az ontológiába betöltendő tudásbázis ábrázolására, az ontológiában megírt tudásbázis szemléltetésére, és emellett kellemes, interaktív felhasználói élményt biztosít. Ugyanakkor a PersonalBrain szókészlete kisebb az OWL szókészleténél, ezáltal nem képes egy az egyben megjeleníteni a tudásbázist, tehát *információ veszhet el* az ontológia megjelenítése során.



3.11. ábra. *PersonalBrain-be importált OWL ábra.*

A PersonalBrain eszköztára vegyíti a fogalomtérkép és gondolattérkép készítéséhez szükséges eszközöket, hiszen a gondolattérkép minden funkciója biztosított benne, a létrehozott

fogalmak összekapcsolhatóak, de a fogalomtérkép szigorú szabályát enyhítve, *nem kötelező a* fogalmak között létrejött *kapcsolatot jellemezni*, viszont lehetőség van rá.

A program kezelése könnyű, hiszen egy kattintással egyszerre akár több fogalom és felrajzolható egy már létrehozott fogalomhoz. A felírt fogalmak a képernyőn egy-egy dobozként megjelennek egymás mellé összegyűjtve, ezzel segítve a könnyebb navigálást az ábrán. A felrajzolt fogalmakhoz és kapcsolatokhoz egy kattintással lehet *típust* is rendelni, de ehhez először a típusokat definiálni kell.

A típusok definiálása egy menüpont segítségével történik, ahol a fogalmakhoz és a kapcsolatokhoz külön-külön hozzárendelhetőek típusok. A típusok szövegesen, egy karakterlánccal azonosított és ábrázolt szövegdobozok, amelyek a kapcsolatot ábrázoló vonalon helyezkednek el, ezáltal könnyen olvasható, és értelmezhető az adott kapcsolat jelentése. A típusok létrehozásánál nincs megkötés, bármilyen karakterlánc értékül adható, de *minden kapcsolat és fogalom legfeljebb egy típussal rendelkezhet.*

A PersonalBrain előnyei közé sorolható, hogy nagymértékben testre szabható a kinézete, hiszen többek közt beállítható a háttér színe, a betű árnyékolása, a vonal stílusa, és az animáció sebessége is.

A PersonalBrain-ben elkészített ábra áttekinthetőségén jelentősen javít a PersonalBrain által alkalmazott *csoportosítás*, amely segítségével elrejthetők a számunkra éppen lényegtelen részletek, és kiemelhetőek a számunkra fontos részek. Ez garantálja a dinamikus vizualizáció megvalósítását az alkalmazásban.

A fogalmak csoportosítására ad lehetőséget az is, hogy minden fogalom megjelölhető többféle *címkével*, ezáltal ha egy adott címkével rendelkező fogalmak kigyűjtésére van szükség, akkor az egyszerűen és gyorsan megoldható.

A fogalmak megcímkézésének lehetősége a PersonalBrain egyik leghasznosabb funkciója, hiszen ez is segíti a tervezés folyamatát, növeli az áttekinthetőséget. Címkék használatával egy kattintással összegyűjthetőek az adott címkével rendelkező fogalmak, látható az azonos címkével rendelkező fogalmak közötti kapcsolatrendszer. A *címkék rendszere* a fogalmakhoz hasonlóan *hierarchikusan felépíthető*, egy címkének legfeljebb egy őscímkéje lehet, viszont akárhány alcímkével rendelkezhet. A fogalmak rendszeréhez hasonlóan az alcímkék is rendelkeznek az őscímkék tulajdonságaival, illetve azon felül hozzáadhatóak új tulajdonságok.

A menüben megtalálható prezentációs mód lehetőséged ad az elkészített ábra prezentálására, a visszafogott animációk, és a teljes képernyős megjelenítés hatására *letisztult, átlátható képet* ad az ábra egyes részeiről. Ezen felül lehetőséget ad a teljes ábra megjelenítésére is, de ekkor csak két szint mélységig mutatja meg az elemeket.

3.12. ábra. *PersonalBrain-ben elkészített ábra.*

A rövid ismertető alapján is látható, hogy a PersonalBrain kiemelkedően oldotta meg a

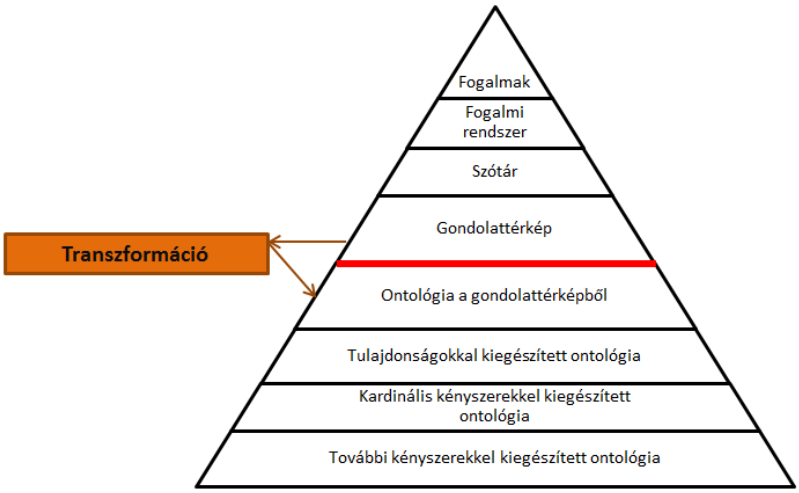
dinamikus ábrázolási, és navigálási lehetőségeket. Az aspektusorientált rendezése nagyban segíti az áttekinthetőséget, a fogalmak egymáshoz való viszonyának ábrázolását.

A PersonalBrain egy hátránya, hogy a kapcsolatok típusait nem jeleníti meg alaphelyzetben, csak akkor, ha a kurzor a linkre mutat (3.12. ábra).

4. fejezet

Specifikáció modellezés

Az előző fejezetben ismertetett áttekintés szerint a kezdeti szakaszban célszerű az *érthetőségre és átláthatóságra koncentrálni*, amelyet legjobban a dinamikus vizualizáció biztosíthat, ezért egyszerű térképtechnikák használatosak, de a formális tudásbázist ontológiában célszerű építeni. Elvben lehetőség van arra is, hogy a tudásbázisa kezdettől fogva ontológiában épüljön, és a térkép annak csak egy vizualizációs rétege legyen, de a jó, és hozzáférhető kereskedelmi eszközök használata érdekében a két fázis elkülönülő technikai eszközkészletet is használhat. Mindkét esetben a kétféle tartalom összekötéséhez *transzformációra* van szükség, vagy *integráltan a megjelenítéskor*, vagy az *egyes fázisok közötti átmenetben.*



4.1. ábra. *Formalizálási folyamat.*

Ezért a formalizálási folyamat következő lépése a 4.1. ábrán megjelenő transzformációs lépés, amely lépés elvégzéséhez szükség van egy megfeleltetési táblázat elkészítése, majd a megfeleltetési táblázat alapján a transzformáció véghezvitelére.

A tudásfeltárás lezártakor szükséges, hogy az informális tudásbázisból, azaz ebben az esetben a PersonalBrain segítségével elkészített tudásbázisból egy formális tudásbázis épüljön. A formális tudásbázis ebben az esetben az ontológiában felírt tudásbázist jelenti. Ennek a megvalósítása két nagy lépésből áll:

- a PersonalBrain fogalmai, és az OWL modellje között egy *megfeleltetési táblázat* elkészítése,
- az elkészített megfeleltetési táblázat *helyességének ellenőrzése* és a helyesség *bizonyítása*.

A megfeleltetés elvégzéséhez szükség van a PersonalBrain és az OWL metamodelljére egyaránt. Az OWL metamodelljét az OMG definiálta az ODM-ben (Ontology Definition Metamodel) [30], amiben az OWL-en kívül további metamodellek is megtalálhatóak, így az ODM nem egy modellt, hanem egy metamodell családot jelent.

A metamodellek különböző információforrások és modelltárakhoz illesztésére az ODM-ben szereplő OWL metamodellt a NeOn [13] összeállította OWL-ben (speciális ontológia-nyelvek kialakításának támogatása), XSD-ben (XML Schema Definition, adatcsere formátumból ontológia séma építés), EMX-ben (Entity Model for XML Schema, későbbi CoDEX, adatbázis illesztés), valamint Ecore-ban (Eclipse Modeling Framework, az MDA tipikus bejárata) is [14], így ezek segítségével egyszerűbben elkészíthető a megfeleltetés a PersonalBrain és az OWL metamodellje között.

Az OMG által specifikált ODM (Ontology Definition Metamodel) öt metamodellt foglal magában:

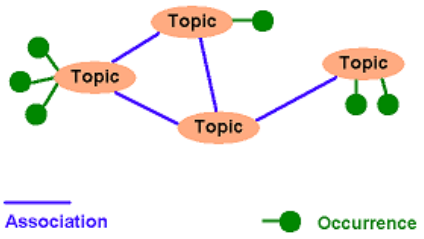
- RDFS
- OWL
- Topic Maps (TM)
- Common Logic (CL)
- Description Logic (DL)

A felsoroltak közül az RDFS-ről, az OWL-ről és a Description Logic-ról a dolgozat korábbi szakaszában volt szó. A szabványban már felbukkannak olyan elemek, amelyek lehetővé teszik a többfázisú modellezést, és fázisok modelljeinek megalapozottan helyes transzformációját. Az ODM ugyanis szabványos kereteket ad a korábbiakban említett fogalomtérkép (concept map) modellezési nyelvére, sőt átveszi az ISO/IEC 24707:2007 szabvány [6] szerinti Common Logic nevű elsőrendű logikát a szemantika szabatos definiálásához.

- A *Topic Maps* megoldást nyújt a nagy, és folyamatosan bővülő információhalmaz reprezentálására, és az információk közötti hatékony keresésre [23]. A Topic Maps az információk ábrázolására *témákat*, *asszociációkat* és *előfordulási számokat* használ (4.2. ábra).
  - A *témák* ábrázolják a fogalmakat, ami akár jelenthet embert, országot, szoftvert vagy eseményt is.
  - Az *asszociációk* ábrázolják a kapcsolatot a témák között.
  - Az *előfordulási szám* reprezentálja, hogy az információ mennyire releváns, hányszor fordul elő a tudásbázisban.

A Topic Maps megjelenésében hasonlít a gondolat-, és fogalomtérképre, azaz grafikusan megjeleníti a fogalmakat és a köztük szereplő relációt, ám a TDK feladatban felvetett probléma megoldásához mégsem használható hatékonyan, mert eddig nem készült hozzá olyan alkalmazás, amely egyszerűen és jól kezelhető, valamint interaktív felhasználói élményt nyújt, azaz hatékony munkára ad lehetőséget.

- Előnye, hogy előre definiált metamodellel rendelkezik, szemben a gondolat-, és fogalomtérképpel.
- Hátránya, hogy nincs nagy választék a grafikus, jól kezelhető és átlátható szerkesztőprogramokból, sőt az XML formátumok szabványosítása sem egyértelmű. A szerkesztőprogramok gyakran csak az egyik típust támogatják.



4.2. ábra. *Topic Maps*

- A *Common Logic* egy keretrendszer az elsőrendű logikát használó nyelvek számára. A keretrendszer célja a szintakszis és az információcsere szemantikájának egységesítése [5].

  A szabványban három logikailag egyenértékű jelölésrendszert definiáltak:

  - Common Logic Interchange Format (CLIF),
  - Conceptual Graph Interchange Format (CGIF),
  - XML-based notation for Common Logic (XCL).

  A Common Logic segítségével megvalósítható a mondatok formalizálása oly módon, hogy a megadott mondat formálisan felírva egyértelmű legyen, és emellett a jelentése megegyezzen az eredeti mondat jelentésével.

  Például a "John is going to Boston by bus." mondat CGIF jelöléssel felírva az alábbi módon néz ki:

```
[Go *x] [Person: John] [City: Boston] [Bus *y]
  (Agnt ?x John) (Dest ?x Boston) (Inst ?x ?y)
```

  Ez ekvivalens az alábbi CLIF jelöléssel:

```
(exists ((x Go) (y Bus))
  (and (Person John) (city Boston)
    (Agnt x John) (Dest x Boston) (Inst x y) ))
```

  A Common Logic Controlled English (CLCE) jelölés rendszert használva, szintén a fentiekkel ekvivalens:

```
The person John goes to the city Boston by a bus.
```

A CLCE jelölésrendszer nem szerepel a szabványban, de egyre több eszköz képest CLCE nyelvet a szabványosított CL jelölésrendszerre lefordítani [34]. A CLCE használata sokban segítheti a CommonLogic használatát, hisz egyszerűen olvasható formában írja le a mondanivalót, viszont fontos a rendkívül precíz fogalmazás használata, mert enélkül a formalizálás során pontatlan CL leírás születhet.

A megfeleltetés elkészítéséhez szükség van a PersonalBrain exportálása során létrehozott BrainXML struktúra metamodelljére, hisz ez tekinthető strukturált tudásbázisnak, amit az ontológiába történő beépítéshez, ezáltal a megfeleltetéshez fel lehet használni. A BrainXML egy olyan XML fájl, ahol a dokumentumban használható elemeket és tulajdonságokat a PersonalBrain által készített DTD (Document Type Definition) [4] határozza meg. A megfeleltetési táblázat elkészítése előtt utánajártam, hogy milyen lehetőségek vannak a megfeleltetés elvégzésére, valamint a megfeleltetés helyességének ellenőrzésére milyen technológiák és eszközök állnak rendelkezésre.

4.1. XSD alapú feldolgozás

A megfeleltetés előkészítésére az egyik lehetőséget az XML feldolgozás jelenti [7], ahol egy XSD séma segítségével az XML feldolgozható és beimportálható az ontológiába. Az XSD-ből OWL-be történő transzformálásban sok kihívás van:

1. anonim típusok transzformálása,
2. egyszerű tartalommal rendelkező komplex típusok konvertálása,
3. OWL property generálás során felmerülő, a beágyazott elemeket és tulajdonság neveket érintő konfliktusok feloldása,
4. OWL class generálás során mikor, és hogyan különböztesse meg az azonos nevű globális elemeket és komplex típusokat,
5. enumeráció generálása,
6. helyettesítési csoportok kezelése az XSD és az XML szinteken egyaránt,
7. XSD típus felülírásának kezelése xsi:type-al az XML-ben.

Tipikus eszközök

A felmerülő kihívásokra egy megoldást ad a Rhizomik *ReDeFer* project-je [21], amelyben egy XSL (EXtensible Stylesheet Language) transzformáció segítségével [24] az XSD-ből OWL készíthető. A Rhizomik által megvalósított transzformációs megoldás Roberto García "A Semantic Web Approach to Digital Rights Management" című PhD. dolgozata [29] alapján készült, amelyben megtalálható a transzformáció menete, valamint megfeleltetési táblázat is. Előnye, hogy az XSL fájl ingyenesen elérhető, ezáltal testre szabható.

Egy másik megoldást a TopBraid Composer használata biztosít. A TopBraid egy fizetős, teljes értékű, Eclipse alapú ontológiaszerkesztő alkalmazás. Képes OWL-t XSD-be elmenteni, valamint XSD-ből OWL-be importálni. A TopBraid használatának hátránya, hogy nem publikálta az XSD-OWL transzformáció menetét, így nem ismert a megfeleltetési táblázata sem.

Alkalmazhatóság

A BrainXML modell OWL-be történő transzformálásának egy lehetséges folyamata az alábbi:

1. Az Allora készített egy webes konvertáló felületet [1], aminek a segítségével a PersonalBrain által publikált DTD fájl átkonvertálható XSD-be.
2. A transzformáció után kapott XSD fájl a TopBraid Composer-be beimportálható, amit a TopBraid automatikusan elment OWL-be.
3. A TopBraid által elkészített OWL fájl megnyitható Protégé-ben.

Ez a munkafolyamat egy lehetőséget biztosít arra, hogy ontológiába átültesse az BrainXML fájl struktúráját, ezáltal egy metamodellt készítsen belőle.

4.2. UML alapú feldolgozás

Egy másik, vizuális megfeleltetési lehetőség a grafikus metamodellek osztálydiagramjainak összefésülése. Az Ecore felépítése hasonlít az ontológiára, léteznek eszközök, amelyek képesek EMF-ből egyből UML diagramot építeni. Amennyiben a felépített ontológiát EMF segítségével ábrázoljuk, akkor utána abból UML készítése már rutin feladat.

A NeOn által Ecore-ban elkészített OWL metamodell egy jó kiindulási alapot ad a megfeleltetés elvégzésére, viszont sajnos se a PersonalBrain-nek, se a BrainXML-nek efféle metamodellje nem áll rendelkezésre. Az XSD-ből UML osztálydiagram automatikus elkészítésére lehetőséget biztosít például az Eclipse-alapú *hyperModel* [11], valamint az Altova termékcsaládból [2] az *XMLSpy*.

A **hyperModel** képes XSD-ből UML-t generálni, de a teljes tartalmat egy osztályba generálja le, így ez a megoldás nem használható megfelelően a feladathoz, hisz nem építette fel a várt metamodell sémát.

Az **XMLSpy** az XSD megnyitása után lehetőséget nyújt arra, hogy UModel UML projectet generáljon, ami aztán az Altova UModel termékével megnyitható, és megjeleníthető vele a generált osztálydiagram.

A modell legenerálása ebben az esetben is egy osztályba történt, ám az osztály tagjai megnyithatóak, tehát megjeleníthető vele a tagok felépítése, és kapcsolatai. Viszont ebben az esetben sem használható fel megfelelően a generált modell, mert nem áttekinthető, és nem olvasható le egyszerűen minden kapcsolat.

5. fejezet

A fázisok közötti modelltranszformáció

A modellek közötti transzformáció kiindulópontja a metamodellek megszerzése/előállítása. Ez a korábbi fejezetekben leírtak szerint nem jelent gondot az OWL-nél (én a NeOn modelljét használtam), de a PersonalBrain ilyen modellt nem ad.

A lehetőségek áttekintése után arra jutottam, hogy az XSD alapú feldolgozás abból a szempontból nem járható út, hogy szemantikai szempontokat nem vesz figyelembe a transzformáció során, ezáltal nem ad pontos képet a PersonalBrain által ábrázolt tudásbázisról, és annak jelentéséről.

Tehát az osztálydiagram alapú megfeleltetést választottam, viszont a generált modell használata ebben az esetben sem járható út, hisz szintén nem tartalmaz szemantikai szempontokat, valamint nem generál áttekinthető, a kapcsolatokat illetően jól felhasználható metamodellt.

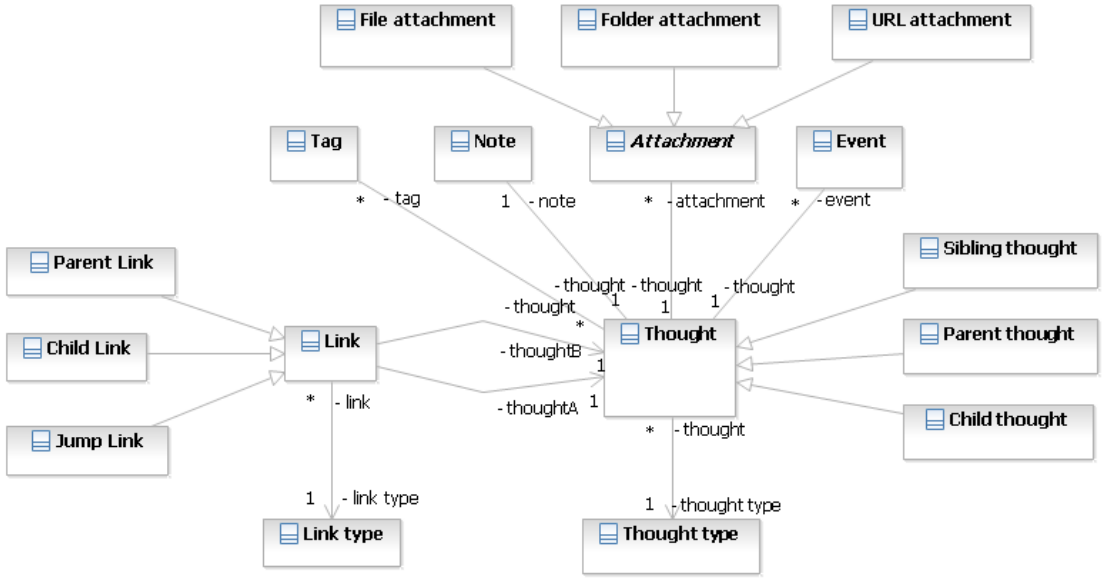
A tapasztalatok alapján végül saját magam készítettem el a PersonalBrain metamodelljét a rendelkezésre álló adatok alapján.

A PersonalBrain metamodelljének elkészítéséhez a PersonalBrain felhasználói útmutatójának [32] részletes tanulmányozása szükséges, hisz nem áll rendelkezésre egy előre elkészített és publikált metamodell. A felhasználói kézikönyv alapos átolvasása után megrajzoltam a PersonalBrain metamodelljét UML osztálydiagrammal ábrázolva (5.1. ábra).

Az elkészített modell azokat a fogalmakat, és a fogalmak közötti lehetséges kapcsolatokat ábrázolja, amelyekből egy PersonalBrain-ben felépített grafikus tudásbázis állhat. Erre a modellre azért van szükség, hogy áttekinthető legyen milyen fogalmak jelennek meg a PersonalBrain-ben, tehát milyen fogalmakat kell keresni a BrainXML feldolgozása során, valamint a megtalált fogalmak között milyen szemantikai kapcsolat van.

A BrainXML metamodelljét az XML dokumentum felépítése, és az XML dokumentumban található attribútumok alapján készítettem el, mert sajnos még az XML formátum részletes specifikációja sem állt rendelkezésre.

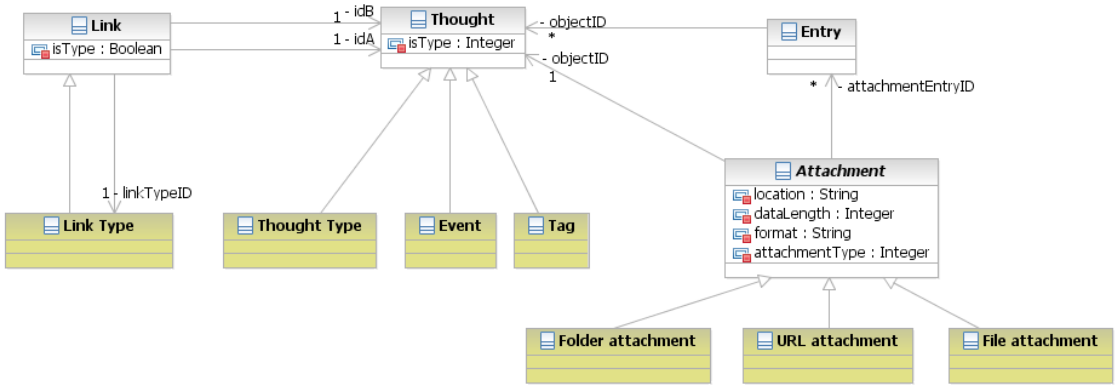
Az XML-ben minden entitáshoz tartozik egy *egyedi azonosító*, aminek a segítségével lehet az adott entitásra hivatkozni más entitásokból. A metamodellben megjelenő kapcsolatokat az XML-ben található egyedi azonosító segítségével történő hivatkozások alapján



5.1. ábra. *PersonalBrain metamodell.*

állapítottam meg. Az ábrán sárgával megjelölt osztályok nem jelennek meg külön tag-ként az XML-ben, hanem az ősosztály egy-egy attribútumának értéke alapján kapnak jelentést (F.2).

Az F.2 függelékben található szótárat elkészítettem öndokumentáló formában is, azaz a fogalmakat felépítettem ontológiában, és a megfelelő tulajdonságok valamint annotációk hozzáadásával a kapott tudásbázisból egy navigálható, HTML alapú dokumentáció készíthető az OWLDoc Protégé plugin segítségével. Ennek a generált dokumentációnak a használata gyakorlati alkalmazás során nagyon hasznos, hisz *jól átlátható*, *navigálható* képet ad a fogalmak tulajdonságairól.



5.2. ábra. *BrainXML metamodell.*

A metamodellek összefésüléséhez az ontológia oldaláról az Ecore-ban felépített OWL-t használtam, így grafikusan jól követhető a megfeleltetés logikája, valamint az osztályok között fennálló kapcsolatok is. Az OWL metamodell felhasználása során csak azokat az elemeket jelenítem meg, amelyek a metamodellek összefésüléséhez szükségesek, hisz az

OWL elemkészlete terjedelmes, ám az elemkészletnek csak a töredéke szükséges a metamodellek összefésüléséhez. A PersonalBrain oldaláról pedig az általam elkészített BrainXML osztálydiagramját valamint a PersonalBrain metamodelljét használtam.

5.1. A PersonalBrain összefésülése ontológiába

A metamodellek összefésülése elsősorban *szemantikai szempontok* alapján történt, hisz így biztosítható, hogy a PersonalBrain-ben felépített tudásbázis szemantikailag megfeleljen az ontológiában felépített tudásbázissal, ami a feladat célja. Miután az OWL metamodelljébe beépítettem a PersonalBrain és a BrainXML metamodelljét, továbbra is fennállnak azok a relációk, amik a BrainXML metamodelljében fennálltak. Tehát *tranzitív kapcsolatokon* keresztül minden elem megfelelően kapcsolódik a másikhoz. Így ami felírható BrainXML-ben, az felírható ontológiában is.

Az 5.3. ábrán látható az összefésült metamodell, ahol a sárgával jelölt elemek az OWL 2 metamodelljét jelentik, a pirossal jelölt elemek pedig PersonalBrain metamodelljének elemei, ezáltal az ábráról jól leolvasható a megfeleltetési táblázat (5.1. táblázat) is.

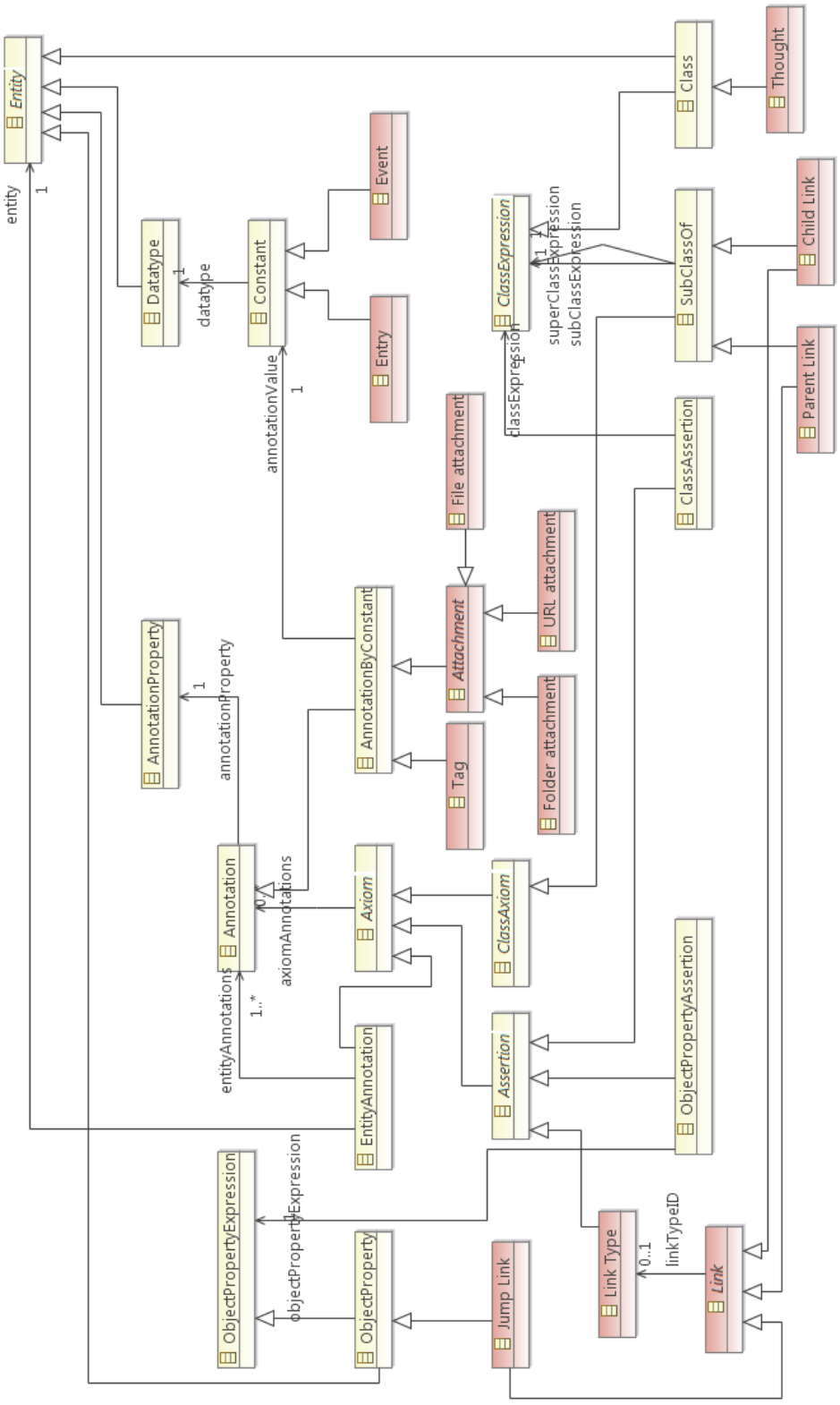
A táblázat első oszlopa az OWL elemkészletét jelöli, a második oszlop a PersonalBrain szókészletét, míg a harmadik oszlopban megtalálható egy rövid magyarázat a megfeleltetéshez.

5.1. táblázat. *Megfeleltetési táblázat*

OWL szókészlet	**PersonalBrain szókészlet**	**Megjegyzés**
Class	Thought	A fogalmak az ontológiában osztályként jelennek meg.
ObjectProperty	Jump Link	A nem hierarchiát kifejező linkek kapcsolattulajdonságként jelennek meg az ontológiában.
Assertion	Link Type	A kapcsolattulajdonság típusa Assertion segítségével jellemezhető.
AnnotationByConstant	Tag	A címke annotációként jelenik meg az ontológiában.
AnnotationByConstant	Attachment	A csatolmány annotációként jelenik meg az ontológiában.
Constant	Entry	Egy megjegyzés konstansnak tekinthető.
Constant	Event	Az esemény konstansnak tekinthető.
SubClassOf	Parent Link	Hierarchikus kifejezések a SubClassOf segítségével hozhatók létre.
SubClassOf	Child Link	Hierarchikus kifejezések a SubClassOf segítségével hozhatók létre.

A megfeleltetés során a *fogalmak feldolgozása* után a legfontosabb elem a *linkek kezelése*, hisz főként ezen múlik a tudásbázis építésének helyessége. A fogalmak OWL elemmel történő megfeleltetése egyértelműen elvégezhető, hisz a *fogalom* ugyan azt a szerepet tölti be a tudásbázisban, amit az ontológiában a *Class*.

A linkek feldolgozása már nem egyértelmű feladat. A transzformáció elvégzése során szükséges a linkek szétválasztására az irányuk szerint, hisz más feldolgozást igényel a *hierarchia* kezelése, valamint az *"oldal" irányú kapcsolatok* kezelése.



5.3. ábra. *PersonalBrain metamodellje beépítve az OWL 2 metamodelljébe.*

A hierarchia szintek felépítésére a SubClassOf OWL elem használata célszerű, míg oldal irányú kapcsolatok során az ObjectProperty használatával lehet meghatározni a két fogalom közötti kapcsolatot. Oldal irányú kapcsolat esetén van elsősorban szükség a link

típusának meghatározására, hisz ezzel lehet jellemezni a kapcsolatot.

A transzformáció során a fogalom típusának (Thought Type) feldolgozása egyelőre nincs elkészítve, mert a típus használatának információtartalma még nem tisztázott, így nem meghatározható a fogalom típus ontológiában betöltött szerepe.

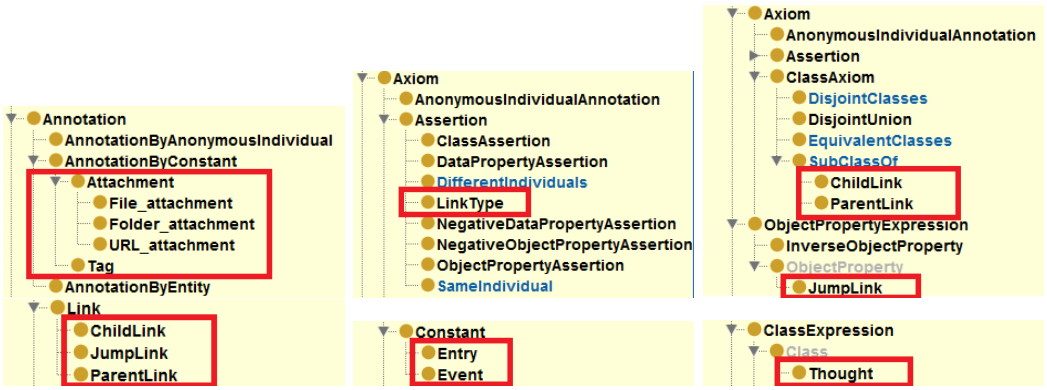
A PersonalBrain és az OWL metamodelljének összevetéséből egyértelműen látható, hogy az ontológia kifejező ereje sokkal nagyobb, mint a PersonalBrain kifejezőereje, hiszen az ontológia szókészlete tágabb, mint PersonalBrain szókészlete. Ezen felül az ontológiában megadható axiómák segítségével olyan kényszerek alkalmazhatóak, amelyeket a PersonalBrain nem képes kezelni.

5.2. Helyesség ellenőrzés

A megfeleltetés végén szükséges annak belátása, hogy a két metamodell valóban megfeleltethető egymásnak, és nem keletkeznek ellentmondások. A helyesség ellenőrzésének egyik módja az *objektum kalkulus* használta [35], amelynek a segítségével felírhatók szabályok az egyes osztályokra, és az azok közötti kapcsolatokra. Az objektum kalkulus során rendkívül fontos a *formális*, *precíz* leírás, azaz, hogy *biztonságos* legyen a felírt kifejezés. Egy kifejezés akkor biztonságos, ha pontosan meghatározza a kifejezésben szereplő objektumok hatáskörét, vagyis a domaint és a lehetséges értékeket.

Egy másik lehetőség *bizonyító automaták* használata, amelyhez szükséges az összefésült osztálydiagram ontológiába történő beépítése.

Miután a PersonalBrain metamodelljét beágyaztam az ontológia metamodelljébe, a szokásos ontológia eszközök adnak lehetőséget az ontológia vizsgálatára. Ezeket eredetileg a *sémák helyességének* vizsgálatára, illetve a *példányok sémához illeszkedésének* vizsgálatára dolgozták ki.



5.4. ábra. *PersonalBrain metamodellje beépítve az OWL 2 metamodelljébe.*

Az összefésült osztálydiagramot tartalmazó ontológia ez után Protégé-ben megnyitható, és a Protégé-be beépített Hermit Reasoner futtatásával megfigyelhető, hogy az ontológia tartalmaz-e hibákat, ellentmondásokat, esetleg egyszerűsítési lehetőségeket.

A PersonalBrain modell teljességét ezen megközelítés nem ellenőrizheti, hiszen arra csak az XML séma felderítésének teljessége garantálhatta. Azt természetesen kézzel ellenőrizni lehet, hogy minden elem megjelent-e a beágyazott ontológiában. Az esetleges ellent-

mondások mindenképpen *beágyazási hibára* utalnának. Ilyen lehetne az, ha egy fogalom finomítását ellentmondásosan tartalmazná az OWL, mint ontológia leíró nyelv, illetve a PersonalBrain beágyazott nyelv.

Az egyszerűsítés pedig azt jelentené, hogy vagy valamely fogalom nem került volna igazán beágyazásra, vagy pedig a beágyazás hibája miatt különböző PersonalBrain fogalmak egybeesnének.

A Hermit Reasoner a futtatás során nem jelzett sem ellentmondásokat, sem egyszerűsítési lehetőségeket az összefésült ontológiában, így feltételezhető, hogy az *összefésülés helyes volt.*

A bizonyító rendszer esetleges fogyatékosságai miatt fel nem fedett hibák elkerülése érdekében a kapott ontológia ellenőrzését egy másik bizonyító automatával is elvégeztem, amelynek a neve Pellet [27].

6. fejezet

Transzformációs program

A korábbiak szerint a PersonalBrain és az ontológia közötti leképezésre az ontológiába ágyazás egy közvetlenül implementálható specifikációt ad. Kiemelendő, hogy ez a formálisan ellenőrzött transzformációs specifikáció természetesen csak a fogalmak beágyazására vonatkozik, de a megvalósítását (PersonalBrain modellek importálása ontológiába) kézzel kell lekódolni.

Ennek megfelelően a már formálisan is ellenőrzött megfeleltetési szabályok alapján implementáltam a transzformációs alkalmazást, amely alkalmazás a PersonalBrain-ben felépített tudásbázist transzformálja ontológiába, illetve az ontológiából készít BrainXML-t, ami betölthető a PersonalBrain-be.

Én az első utat követtem, mert a közvetlen importálás túlságosan eszközhöz kötette volna az alkalmazást. Még az olyan elterjed eszközök esetében is, mint a Protégé, az ontológiatár kezelésére szolgáló interfészeket gyakran változtatják, és csak ad-hoc jelleggel szabványosítják.

6.1. XML alapú transzformáció

A teljesen kitöltött gondolattérkép és ontológia közti átjárás lényegében egy szintaxis vezérelt modelltranszformáció, esetleges információvesztéssel. Az információvesztés jellemzően az ontológiából történő transzformáció során következik be, hisz a gondolattérkép kifejezőereje jóval kisebb. A PersonalBrain beágyazása ontológiában viszont a korábbiak szerint veszteségmentes.

Az interjúkészítés *fokozatosan finomodó modelleket* eredményező *iteratív folyamat*, ahol szokásosan az interjúkészítő eszköz (esetünkben a PersonalBrain) vizuális annotációs mechanizmusát arra is használják, hogy az egyes modell elemek készültségi fokát is jelezzék. Ennek megfelelően az interjúkészítés során az aktuális modellben lehetnek készre jelölt részek, amelyek helyességét ellenőrizni kell, illetve félig kész, kitölteni szándékozott részek is. Ez a megközelítés nyitva hagyja azt a lehetőséget, hogy akár a modellépítés szakaszaiban is ellenőrizhető legyen a végfelhasználó számára a modell egyes részei, amíg a többi rész kifejtendő, félkész állapotban van.

További érdekes fejlesztési irány annak a vizsgálata, hogy hol lehet abbahagyni a modell

építését. Ez annyit jelent, hogy milyen módon lehet megállapítani azt a pontot, ahol a modell még konzisztens állapotban van, és a modell finomítása szüneteltethető. Ez azt igényli, hogy az OWL annotációs mechanizmusait is felhasználjuk specializált módon.

6.2. Elkészített program

Az importálásra több lehetőség is kínálkozik:

- a két XML formátum közötti transzformáció, amelynek előnye, hogy eszköz független,
- a PersonalBrain XML importálása valamely ontológiatárba, és az ott letárolt elem exportálása OWL/XML formátumba.

Bár a PersonalBrain és az RDF/OWL közt jól látható strukturált hasonlóság miatt elvben az XSLT-vel is megoldható lenne a transzformáció, de mivel a belső hivatkozási rendszer összetett, ezért a Java DOM (Document Object Model) technológiát választottam a feldolgozáshoz. Ez egyúttal lehetőséget ad arra, hogy később, a modell ellenőrzésekor előkerülő hibákat lehetőség szerint vizuálisan is megjelenítse az alkalmazás (például tovább nem bontott, formálisan eliminálható absztrakt osztályok megjelenése, vagy akár az olyan notációk érdemi feldolgozása, amelyek a feldolgozás során finomítandóak, ellenőrizendőek vagy befejezettnek tekinthetőek). Ezek mellett a Java segítségével megoldható a modell projekció is, amit XSLT-vel nem lehetne elkészíteni.

A PersonalBrain az ontológiának egy *kivetítését* biztosítja, nem tartalmazza az ontológia teljes tartalmát, hisz a kifejező ereje jóval kisebb. Az alkalmazás jelenleg két fő funkciót valósít meg:

- Ontológiában elkészített tudásbázisból BrainXML készítése.
- PersonalBrain-ből ontológia készítése, változáskövetés támogatásával.

A PersonalBrain-be gyárilag OWL importálót építettek, de az nem teljes értékű, csak a hierarchia szinteket képes megjeleníteni, egyéb információt és kényszereket nem vezet át a grafikus modellbe. Emiatt az alkalmazásom saját ontológia-BrainXML transzformációt valósít meg, ezáltal a hierarchia szintek kezelésén túl plusz információt visz a modellbe. A plusz információt egyelőre a címkék, valamint annotációk kezelése jelenti.

6.1. ábra. *Importálás bővítése*

Változáskövetés esetén lehetőség van egy félkész ontológia beimportálására a PersonalBrain-be, majd a vizuálisan megjelenített, egyszerűsített ontológián elvégzett változtatások átvezetésére az eredeti ontológiába.

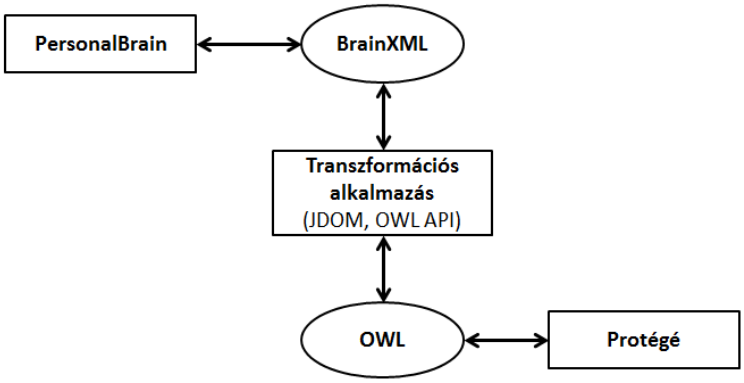
Utóbbi funkció azért szükséges, mert ezáltal az elkészített tudásbázis egy egyszerűsített kivetítése prezentálható a tárgyterületi szakértő felé, aki ellenőrizheti a modell szemantikai helyességét, valamint módosításokat végezhet el a modellen.

Módosítás esetén nélkülözhetetlen annak követése, hogy mi változott az eredeti ontológiához képest, hisz nem célszerű egy az egyben a PersonalBrain-ben módosított ábrát visszaalakítani egy új ontológiába, mert ebben az esetben információ veszne el, hisz importálás során csak a tudásbázis egy része kerül át a PersonalBrain-be.

Változáskövetés esetén megvalósítható az eredeti ontológia információtartalmának megőrzése, és csak a vizuálisan megjelenített modellen elvégzett változások legyenek átvezetve az eredeti ontológiába.

Az alkalmazás az ontológia létrehozására, valamint az ontológia feldolgozására a Javahoz kidolgozott OWL API könyvtárat [16] használja, amely teljes támogatást nyújt az OWL 2 formátum szerkesztéséhez.

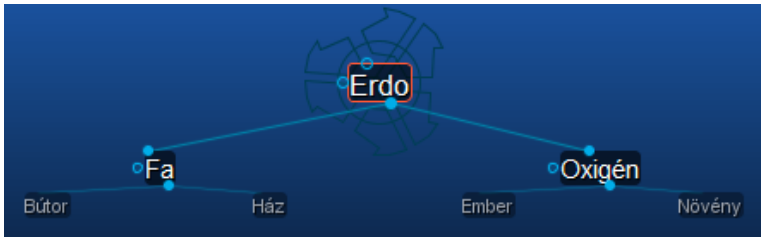
Az alkalmazás használatának folyamatábráját a 6.2. ábra mutatja be.



6.2. ábra. *Transzformáció folyamatábrája.*

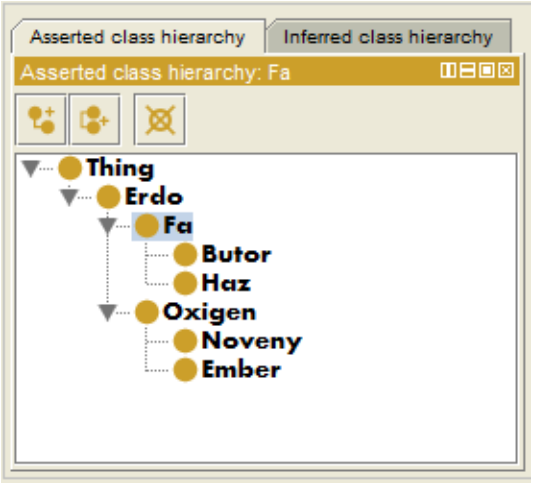
6.3. A program működése

A program használatához a PersonalBrain-ben elkészített ábrát (6.3. ábra) BrainXML-be kell importálni, mert a transzformációs program a PersonalBrain által generált BrainXML formátumot támogatja.



6.3. ábra. *PersonalBrain*

A transzformációs program által generált OWL fájlt megnyitva Protégé-ben (6.4. ábra) látható, hogy a PersonalBrain-ben megrajzolt ábrának megfelelően megjelennek a felrajzolt osztályok, és minden felrajzolt osztály a megfelelő hierarchia szintre került.



6.4. ábra. *Protege-ben megnyitva a generált fájl.*

Az összes térképszerű eszközhöz hasonlóan a fogalmi relációk alapértelmezése a "részfogalom", ez természetesen veszélyes, hiszen ha egy más kapcsolat bevezetésekor lemarad az élcímke, akkor szintaktikusan ugyan helyes, de értelmetlen diagram születik. Az 5.3 ábra térképén az erdő-fa erdő-oxigén reláció címkéjének elmaradása mind a fát, mind az oxigént egy speciális erdőfajtává tette.

6.5. ábra. *Helyes PersonalBrain ábra*

A tapasztalatok alapján a későbbiekben célszerű kiegészíteni az ontológiából PersonalBrain-be történő megjelenítést, az altípus kapcsolat explicit megjelenítésével (számomra ezt az ontológia nyilván megmutatja). Ez ugyanakkor jól mutatja azt is, hogy a formalizálásnak mi a jelentősége, hiszen az ontológia reasonerrel való ellenőrzése ezt megmutathatja.

7. fejezet

Esettanulmány

7.1. Bevezető

A szakdolgozat [31] alkalmával a banki szektor működésével kapcsolatos esettanulmányt készítettem el, mert a banki szektor működését több aspektusból kell vizsgálni (pénzügyi, jogi, szervezeti, informatikai), ezáltal a specifikáció kialakításához mindenképpen interdiszciplináris tudásfeltárás kell. A TDK dolgozatban ezt az esettanulmányt dolgozom fel újra, immár a TDK dolgozatban ismertetett formalizálási mechanizmussal, automatizált transzformáció segítségével.

7.2. Korábbi eredmények

A szakdolgozatban [31] az esettanulmány elkészítése során kiválasztottam a banki folyamatok közül egy olyan példát, amely alkalmas a feladat szemléltetésére. A kiválasztott példa a belföldi forint átutalás folyamata volt, hiszen ez a mindennapos életben is gyakran használt tranzakció, és egyúttal kellőképpen összetett ahhoz, hogy reprezentatív tömegű specifikációs elemet használjon. Ezt a folyamatot természetesen a bankok között, illetve a felhasználó felé jogszabályok szabályozzák, de a konkrét implementáció ezeken a kereteken belül banki belügy, azaz a fogalomkészlet is általános, jogilag szabatos fogalmak, és azoknak bankon belüli implementációjából áll.

A szakdolgozatban [31] összegyűjtött terminológia két csoportra osztható fel:

- törvények,
- működési szempontok/fogalmak.

A terminológiában szereplő törvények és rendeletek [12]:

Basel II ajánlás: a Bázel II-n alapuló 2006/48/EC direktívát a 2007. évi LI. törvény a hitelintézetekről és a pénzügyi vállalkozásokról szóló 1996. évi CXII. törvény, valamint egyes szakosított hitelintézetekről szóló törvények módosításáról, továbbá a 196/2007. (VII. 30.) Kormányrendelet a hitelezési kockázat kezeléséről és tőkekövetelményéről nevű jogszabályok ültetik át a magyar jogba [25]

devizanemek kezelését szabályozó törvény: 18/2009. (VIII. 6.) 2. melléklete

díjszabással kapcsolatban: ÁSZF (Általános Szerződési Feltételek)

könyveléssel kapcsolatos: Számviteli törvény

nyomtatványokkal kapcsolatos törvény: 18/2009. (VIII. 6.) MNB rendeletének 13. §

számlaértesítés esetén: 18/2009. (VIII. 6.) MNB rendeletének 39. §

tartalékrátát szabályozó törvény: 2001. évi LVIII. törvény 9. §

A terminológiában szereplő fogalmak és szakszavak [3]:

átutalás: bankszámlák közötti pénzmozgás, általában az indító számlatulajdonos rendelkezése alapján

átutalási nyomtatvány (PF1): a bank által kibocsájtott nyomtatvány, aminek a segítségével átutalási tranzakció indítható

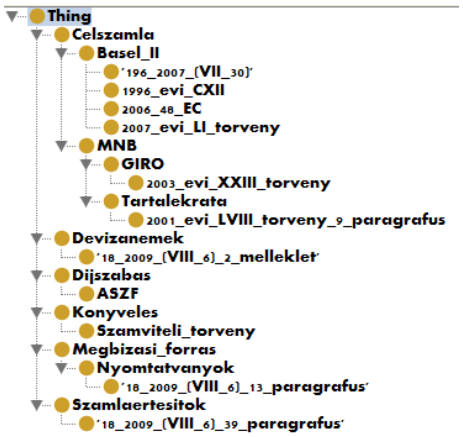
célszámla: az a bankszámla, amelyre az átutalás megérkezik

devizanem: valamilyen külföldi pénz, pl. euró vagy amerikai dollár, pl. euró alszámla esetében az mondható, hogy ilyenkor a számla devizaneme euró; a forint is egy devizanem

megbízási forrás: az a csatorna, amelyen keresztül az ügyfél a banki művelet teljesítését kérvényezte

számlakivonat: egy elszámolási időszak (általában hónap) forgalmainak papír alapú összesítése

tartalékráta: a kereskedelmi bankok az általuk gyűjtött betétek és egyéb források után - a tartalékráta által meghatározott százalékban - tartalékot kötelesek képezni a jegybanknál. A tartalékráta emelése szűkíti, csökkentése növeli a gazdaságban lévő pénzmennyiséget

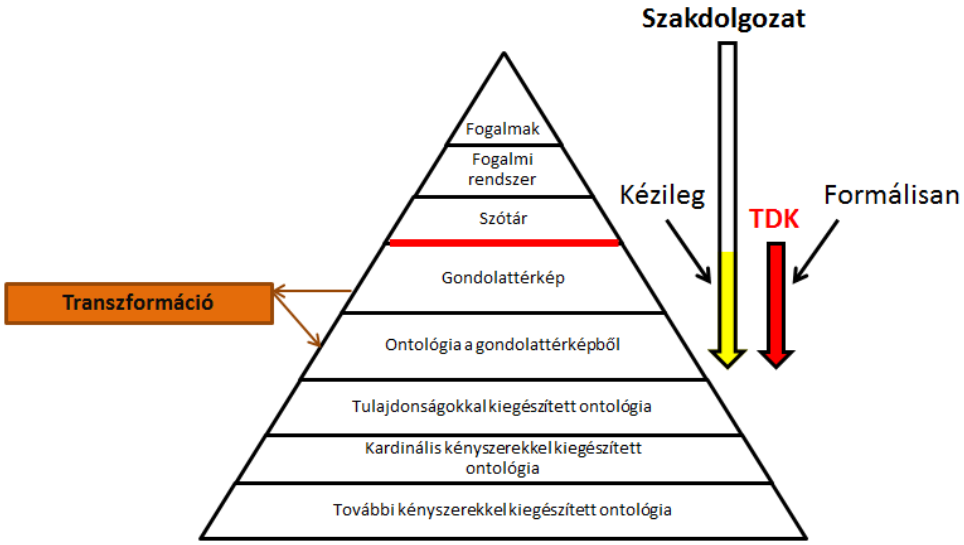


7.1. ábra. *Gondolattérkép felépítése ontológiában.*

A szakdolgozatban elvégzett munka a kézzel felépített informális tudásbázis átültetése ontológiába kézi transzformáció segítségével. Ez a módszer nem hatékony se idő, se hibamentesség szempontjából.

Az akkori transzformáció eredménye a 7.1. ábrán látható. A 7.1. ábra bal oldalán látható ontológia ábrázolja, hogy mely törvény mit szabályoz, a jobb oldali ontológia pedig a fontos fogalmak felépítését tartalmazza.

7.3. Új megközelítés



7.2. ábra. *Esettanulmány átdolgozásának szakaszai.*

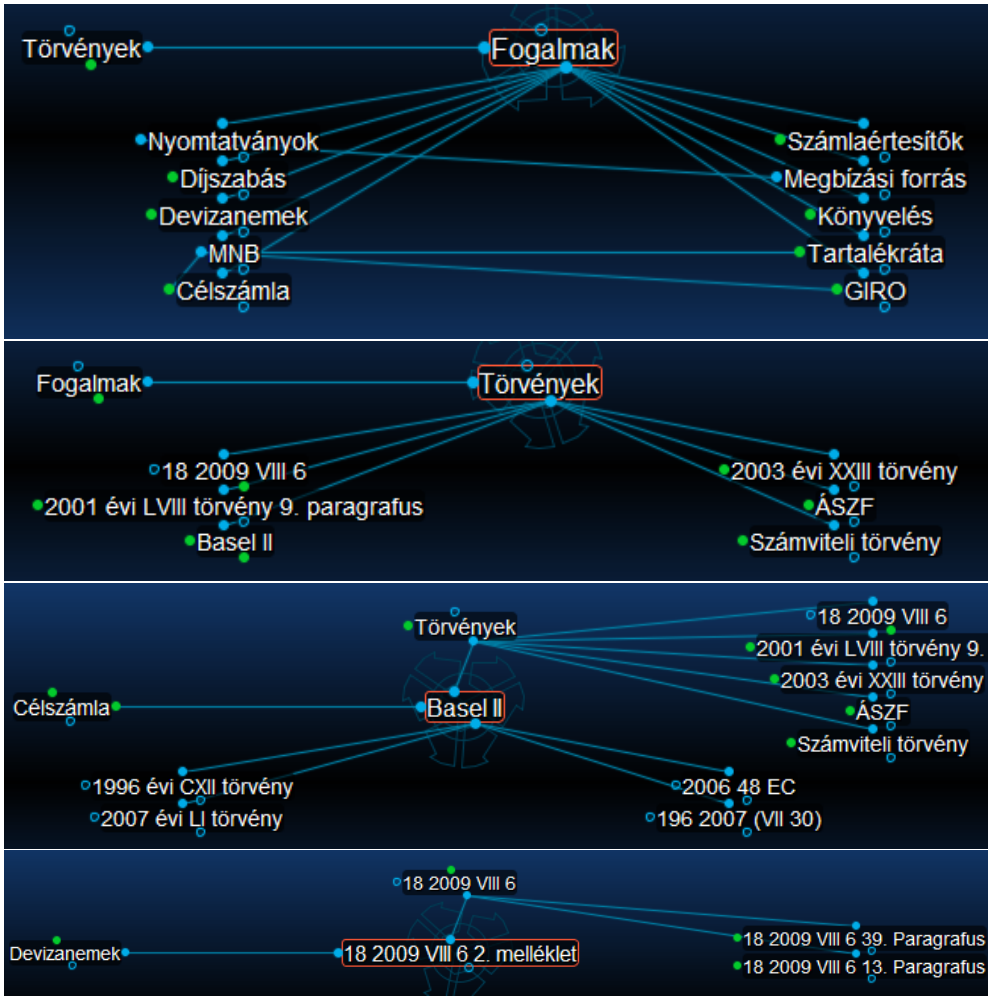
A 7.2. ábrán látható, hogy a szakdolgozat során a folyamat felépítése a kezdetektől a transzformáció utáni lépésig tartott. A TDK dolgozatban nincs szükség az interjúzás újbóli elkészítésére, a szakdolgozatban összeírt szótár megfelelően alkalmazható jelen esetben is. A TDK dolgozat elsősorban a formalizálás folyamatát tanulmányozza részletesen, a PersonalBrain és az ontológia közti formalizálásának lehetőségeit.

Az esettanulmány elkészítésének új megközelítése az alábbi:

- Terminológia felrajzolása PersonalBrain-be.
- PersonalBrain-ből BrainXML készítése.
- BrainXML feldolgozása.
- A transzformációs szabályok alapján automatizáltan ontológia építése.

Az új megközelítés lényege a *kézi módszerrel történő ontológia építésének* elkerülése. Manuális ontológia építés helyét az *automatizált ontológia építés* váltja ki.

Az automatizált ontológiaépítésben az előző fejezetben ismertetett transzformációs alkalmazás segít. Mivel az alkalmazás még fejlesztés alatt áll, ezért a funkciók korlátozottan érhetőek el benne, de az esettanulmány bemutatására már alkalmas.



7.3. ábra. *PersonalBrain-ben felrajzolt törvények és fogalmak.*

A 7.3. ábrán látható a PersonalBrain-ben felépített törvényeket tartalmazó tudásbázis. Az első képen a felmerülő fogalmak láthatóak, a második képen a szükséges törvények, a harmadik, negyedik képen pedig egy-egy példa arra, hogy az egyes törvények hogyan bonthatóak részekre, valamint hogyan kapcsolódnak a fogalomhoz. Az így felrajzolt ábrát el kell menteni BrainXML formátumba ahhoz, hogy az alkalmazás fel tudja építeni belőle az ontológiát.

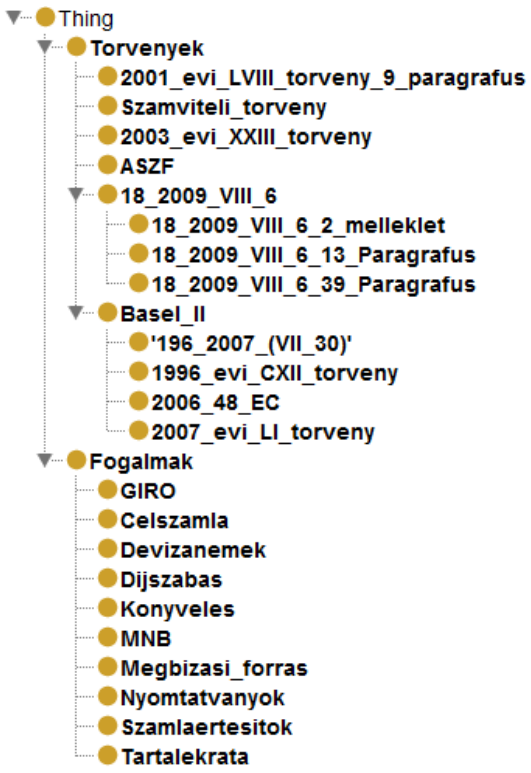
A transzformációs alkalmazásba betöltött BrainXML-ből OWL fájl készül, amit Protégében megnyitva látható a hierarchikus tudásbázis szerkezet (7.4. ábra).

7.4. Értékelés

7.4.1. Fejlesztési tapasztalatok

Helyesség ellenőrzés

A szakdolgozat során a hagyományos specifikációs technikákból indultam ki, és kézzel kódoltam. A TDK során többszörösen formálisan is megvizsgáltam a modelltranszformáció alapú specifikációt, így az teljesen egzakt és helyes volt. Egyúttal a transzformációs szabá-



7.4. ábra. *Protégé-ben felépített tudásbázis a fogalmakból és törvényekből.*

lyok megléte miatt kisebb részfeladatokkal lehetett implementálni a programot. Még egy ilyen kicsi példa is jól mutatja a modellalapú megközelítés előnyét:

- A szakdolgozatban elkészített transzformáció felépítése hibás volt, ennek a javítása a jelen esettanulmányban megtörtént. A hibát a nem megfelelően használt hierarchikus felépítés okozta, hisz nem voltak szisztematikusan szétválogatva a fogalmak, így a tudásbázis felépítse sem lehetett jó.

- A hiba forrása elsődlegesen a nem megfelelően átgondolt tudásbázis építése volt, hisz a hierarchia szintek használata helyett tipizált kapcsolatok használta javasolt az esettanulmány során. Azonban a megfelelő formalizálás és ellenőrzés hiányában a szakdolgozatban kézzel elkészített tudásbázis hiányosságaira és hibáira nem derült fény.

7.4.2. Alkalmazási tapasztalat

Időbeli ráfordítás

A kialakult technológia azt ígéri, hogy a tacit tudás feltárási folyamat melléktermékeként létrejön egy olyan gondolattérkép, ami legalább informálisan (szakértői review útján) ellenőrzött. A tapasztalatok szerint a transzformáció időigénye elhanyagolható, ezzel sikerült hidat teremteni a korábbi kézi modellépítés, és információkészítés alapján előálló térkép automatikus feldolgozására.

Az időbeli ráfordítás szempontjából az automatizált megoldás lényegesen gyorsabb megvalósításra adott lehetőséget. Míg kézi módszerrel a törvényeket tartalmazó ábrának felépítése nagyjából 3 percig tart (és közben fennáll az elgépelés veszélye, valamint egyéb hibák elkövetése), addig automatizált eszközökkel nagyjából 2 másodperc alatt elkészül. A mérések nem másodperc pontosak, hisz az ontológia építési idő manuális eszközök esetén nagyban függ az ontológiát építő személytől (eszköz ismeret, ontológia ismeret, gépelési sebesség), automatizált esetben pedig a futtatókörnyezet befolyásolhatja a futási időt, viszont a nagyságrendi különbség így is szembetűnő.

Az esettanulmányok összevetéséből látható, hogy rendkívül fontos feladat a megfeleltetési táblázat precíz elkészítése, hisz ennek hiányában a transzformáció szemantikai hibákat tartalmazhat, ami pontatlan tudásbázishoz vezet. Továbbá az eredmények tekintetében látható az is, hogy szükség van az automatizált feldolgozásra, mert sok emberi hiba kiküszöbölhető vele, emellett jelentős gyorsulást eredményez a feldolgozás során.

8. fejezet

Továbbfejlesztési lehetőségek

A témában további fejlesztési, kutatási lehetőségek vannak. Egy hasznos továbblépés lehet integrálni a specifikációtervezés folyamatába a tudásbázis automatikus formalizálását, így már a tervezés korai fázisában előkerülhetnek olyan pontatlanságok, ellentmondások, amelyek javítása a fejlesztési folyamat későbbi szakaszaiban költséges.

8.1. Algoritmikus fejlesztési lehetőségek

További fejlesztési lehetőség rejlik az alkalmazás fejlesztésében, hisz a megfelelően kialakított szóhasználati kényszerekkel a PersonalBrain kifejező ereje tovább nőhet, például amennyiben előre meghatározott link típusok használatát az ontológia egy-egy elemének dedikáljuk, akkor akár bonyolultabb kapcsolatrendszer is kiépíthető a grafikus megjelenítő segítségével.

Továbbfejlesztési irány lehet, a PersonalBrain-be beimportálható, az ontológiában szereplő kezdőfogalmak, így a notáció megtanulhatja azt, hogy mely fogalmak használhatóak, melyek azok a fogalmak, amik az importált ontológia részét képezik.

Számos PersonalBrain-ben felhasználható kérdés sablon kifejlesztésére van lehetőség. Például nagy számú ontológiatár létezik, amelyek a biztonság területével foglalkoznak, ezeknek az ontológiáknak a felhasználásával elkészíthető olyan kérdés sablon, ami biztosan lefedi minden részterületét a biztonságnak.

Jelen pillanatban az elkészített alkalmazásban a link típusokra a tartalmazást kivéve csak jelölés szintű megfogalmazás van, ez elvben összekapcsolható egy precízebb ontológiabeli megfogalmazással, amivel ellenőrizhetővé válik a link ellentmondás-mentessége.

Az elkészített transzformációs alkalmazás még fejlesztés alatt áll, nincs elkészítve a teljes funkcionalitása, így ennek befejezése is továbbfejlesztési lehetőség.

Az alkalmazásból jelenleg az alábbi funkciók hiányoznak:

- Hiányzik a link típusok feldolgozása.
- Hiányzik a fogalom típusok kezelése.
- Hiányzik a PersonalBrain Note ablak tartalmának teljes feldolgozása.

- Az csatolmányok és címkék egyelőre az ontológiában csak tipizált annotációként jelennek meg.
- A címkék célirányú feldolgozása (például: befejezett, ellenőrizendő, kiegészítendő címkék értelmezése).

8.2. Gyakorlati felhasználások

A témában gyűjtött tapasztalatok, valamint az elkészített alkalmazás gyakorlatban is felhasználásra kerül több futó projectben is.

Az egyik project az *Emerald* [9], amelynek célja a jogi szövegek feldolgozása. A jogi szövegben található hivatkozott jogszabályok felismerése, és ezek alapján tudásbázis építése, amelyben visszakövethető, hogy mely törvény mely más törvényekre mutat. Ebben az esetben a tudásbázis vizuális megjelenítésénél használható fel a kutatások során szerzett tapasztalat, azaz a tudásbázis grafikus környezetbe történő átültetésének módja, majd a grafikus felületen elvégzett módosítások visszavezetése az eredeti tudásbázisba.

A következő két projectben az interjúztatás sikerességén van a hangsúly, illetve az interjúztatás során összegyűjtött adatok formalizálásának lehetőségén, hisz minden további lépés ezekre az információkra épül.

Az egyik ilyen project az *e-Freight* EU project [8], ahol a nemzetközi szállítmányozásban használatos rendszer fejlesztése a cél, amely egy egyablakos felületet biztosít a felhasználónak, ahol minden szükséges információt megtalál, ami fontos az adott szállítmánnyal kapcsolatban, valamint azonnal képes a szükséges engedélyeztetések elindítására. Ebben a projectben nagy szerepe van a szállítmányozás folyamatának pontos feltárására, a különböző szállítmányokhoz szükséges engedélyek hiánytalan összegyűjtésére, illetve egyéb korlátozások precíz összegyűjtésére és felhasználására.

A másik, éppen aktuális project Magyarország egy vezető bankjának rendszerét érinti. A project célja a banki működés feltérképezése, a bankon belüli folyamatok megismerése, és a folyamatok közötti függőségek (adat, illetve időbeli) meghatározása. A banknak jelenleg nincs átfogó képe a saját rendszeréről, ezért a project egyik célja, hogy a feltárt tudás érthető, áttekinthető formában az ügyfélnek prezentálható legyen. A project végén az összegyűjtött adatok alapján a teljes rendszerről egy olyan tudásbázis készül, amely alapján felismerhető a szükségtelen redundancia, és ezáltal javítási, optimalizálási javaslatok tehetőek a rendszerrel kapcsolatban.

Függelék

F.1. Kérdés sablon

F.1. táblázat. *Kérdés sablon.*

Elemkészlet	**Kérdés**
Classes	Mik a fogalmak?
Datatypes	Mik a tudásbázisban ábrázolandó fogalmak értéktartományai?
Object Properties	Milyen fogalmak közti kapcsolatok vannak?
Data Properties	Milyen tulajdonságai, attribútumai vannak a fogalomnak?
Annotation Properties	Milyen egyéb típusú megjegyzések fűzhetőek a fogalomhoz?
Individuals	Mik a példányai a fogalomnak?
Property Expressions	Milyen tulajdonság kifejezések vannak?
Object Property Expressions	Milyen kapcsolat tulajdonságokra milyen kifejezések vannak?
Data Property Expressions	Milyen adat tulajdonsági kifejezések vannak?
Data Ranges	Milyen értékbeli korlátok tartoznak a fogalomhoz?
Intersection of Data Ranges	Milyen értéktartományok metszete lehet?
Union of Data Ranges	Milyen értéktartományok uniója lehet?
Complement of Data Ranges	Milyen értéktartományok komplementere lehet?
Enumeration of Literals	Milyen értékeket vehet fel pontosan?
Datatype Restrictions	Milyen értéktípusbeli megkötés van?
Class Expressions	Milyen fogalmakat és azok tulajdonságait érintő kifejezések vannak?
Intersection of Class Expressions	Melyek azok a példányok, amelyek minden felsorolt fogalomnak példányai?
Union of Class Expressions	Melyek azok a példányok, amelyek legalább egy felsorolt fogalomnak példányai?
Complement of Class Expressions	Melyek azok a példányok, amelyek nem példányai a megadott fogalomnak?

Enumeration of Individuals	Pontosan mely példányok lehetnek a fogalomhoz tartozó példányok?
Object Property Restrictions	Milyen megkötései vannak a másik fogalommal felépített kapcsolatnak?
Existential Quantification	A kapcsolat mely fogalmak példányaiból köthet össze valamennyit?
Universal Quantification	A kapcsolat csak mely fogalmak példányait kötheti össze?
Individual Value Restriction	Melyek azok a példányok, amelyek a megadott tulajdonsággal rendelkeznek?
Self-Restriction	A fogalom önmagával milyen kapcsolatban lehet?
Object Property Cardinality Restrictions	Más fogalmakkal felépített kapcsolatban milyen mennyiségi szabályok vannak?
Minimum Cardinality	Legalább hány példánynak kell szerepelnie a kapcsolatok között az adott kapcsolattípussal?
Maximum Cardinality	Legfeljebb hány példánynak kell szerepelnie a kapcsolatok között az adott kapcsolattípussal?
Exact Cardinality	Pontosan hány példánynak kell szerepelnie a kapcsolatok között az adott kapcsolattípussal?
Data Property Restrictions	A fogalomhoz kapcsolódó értékekre milyen megkötések vannak?
Existential Quantification	Milyen korlátok közül vehet fel értékeket?
Universal Quantification	Milyen korlátok közül kell felvennie értékeket?
Literal Value Restriction	Megegyezik-e az adott kifejezéssel a beállított érték?
Data Property Cardinality Restrictions	Milyen adatokhoz kapcsolódó számossági megkötések vannak?
Minimum Cardinality	Az adott adattal legalább hány fogalomnak kell rendelkeznie?
Maximum Cardinality	Az adott adattal legfeljebb hány fogalom rendelkezhet?
Exact Cardinality	Az adott adattal pontosan hány fogalom rendelkezhet?
Axioms	Milyen szabályok léteznek?
Class Expression Axioms	Milyen kapcsolat van két fogalom között?
Subclass Axioms	Milyen fogalmak származnak ebből a fogalomból?
Equivalent Classes	Mivel ekvivalens ez a fogalom?
Disjoint Classes	Milyen fogalmak használatát zárja ki ez a fogalom?
Disjoint Union of Class Expressions	Mely fogalomcsoportok zárják ki egymást?
Object Property Axioms	Fogalmak közötti kapcsolat kifejezéseket hogyan lehet jellemezni?

Object Subproperties	Milyen altulajdonságai lehetnek a kapcsolatnak?
Equivalent Object Properties	Mely kapcsolattulajdonságok ekvivalensek egymással?
Disjoint Object Properties	Mely kapcsolattulajdonságok zárják ki egymást?
Inverse Object Properties	Mi a tulajdonság inverz tulajdonsága?
Object Property Domain	A kapcsolat melyik fogalmak példányait kötheti csak össze?
Object Property Range	Ha egy példány az adott kapcsolaton keresztül kapcsolódik a fogalomhoz, akkor az a példány az adott fogalom példánya?
Functional Object Properties	Mely tulajdonságra és mely példányokra igaz, hogy pontosan egy másik példányhoz kapcsolódnak az adott tulajdonságon keresztül?
Inverse-Functional Object Properties	Mely példányokra igaz, hogy pontosan egy másik példányhoz kapcsolódik az adott tulajdonságon keresztül?
Reflexive Object Properties	Mely tulajdonságon keresztül kapcsolódik önmagához az adott példány?
Irreflexive Object Properties	Mely tulajdonságon keresztül nem kapcsolódhat önmagához az adott példány?
Symmetric Object Properties	Mely tulajdonság segítségével kapcsolódik egymáshoz kölcsönösen két példány?
Asymmetric Object Properties	Mely tulajdonságra igaz, hogy az ezáltal összekapcsolt példányok kapcsolata nem kölcsönös?
Transitive Object Properties	Mely kapcsolatok biztosítanak tranzitivitást?
Data Property Axioms	Milyen adattípusokat érintő szabályok vannak?
Data Subproperties	Az adathoz milyen altulajdonságok tartoznak?
Equivalent Data Properties	Mely adatok ekvivalensek egymással?
Disjoint Data Properties	Mely adatok megkülönböztetése szükséges?
Data Property Domain	Az adat honnan vehet fel értéket?
Data Property Range	Az adat milyen tartományban vehet fel értéket?
Functional Data Properties	Mely adatból rendelkezhet minden példány csak eggyel?
Datatype Definitions	Milyen adattípusok vannak?
Assertions	Milyen kijelentések tehetőek?
Individual Equality	Mely példányokról jelenthető ki, hogy ekvivalensek egymással?
Individual Inequality	Mely példányok különböztethetőek meg egymástól?
Class Assertions	A példány melyik fogalomnak a példánya?
Positive Object Property Assertions	Két példány milyen tulajdonságon keresztül kapcsolódik egymáshoz?

Negative Object Property Assertions	Milyen tulajdonságon keresztül nem kapcsolódhat egymáshoz a két példány?
Positive Data Property Assertions	A példány milyen adattípus tulajdonságon keresztül kapcsolódik az adott értékhez?
Negative Data Property Assertions	A példány milyen adattípus tulajdonságon keresztül nem kapcsolódhat az adott értékhez?
Annotations	Milyen megjegyzések fűzhetőek a fogalomhoz?

F.2. BrainXML szótár

- **Fogalom (Thought)**
  - A felhasználó által megadható tetszőleges szöveg.
  - A programban szövegdobozként jelenik meg.
  - Az XML kimenetben Thought tag-el jelölt rész, amin belül szerepel az isType tag, és ennek értéke 0.
- **Fogalom típus (Thought type)**
  - A felhasználó létrehozhat fogalom típusokat, amiket tetszőleges számú fogalomhoz rendelhet. Egy fogalomnak csak egy típusa lehet.
  - A programban a tipizált fogalom típusa akkor jelenik meg, ha a tipizált fogalom fölé kerül az egérmutató.
  - Az XML kimenetben Thought tag-el jelölt rész, amin belül szerepel az isType tag, és ennek értéke 1.
- **Esemény (Event)**
  - A felhasználó a fogalmakhoz eseményeket rendelhet.
  - A programban az események a Calendar menüpontban láthatóak.
  - Az XML kimenetben Thought tag-el jelölt rész, amin belül szerepel az isType tag, és ennek értéke 2.
- **Címke (Tag)**
  - A felhasználó a fogalmakhoz címkéket rendelhet. Egy címke több fogalomhoz is kapcsolódhat, és egy fogalomhoz több különböző címke is rendelhető.
  - A programban a címke a fogalom jobb alsó sarkánál jelenik meg szövegdobozként.
  - Az XML kimenetben Thought tag-el jelölt rész, amin belül szerepel az isType tag, és ennek értéke 3.
- **Kapcsolat (Link)**

- A felhasználó a fogalmak között kapcsolatot építhet. Egy kapcsolat két fogalmat köt össze. A kapcsolatnak 3 iránya lehet: gyerek (child), szülő (parent), testvér, ahol nem közös a szülő (jump).
- A programban a kapcsolat egy vonallal van megjelenítve, ahol a vonal a két fogalom között húzódik.
- Az XML kimenetben Link tag-el jelölt rész, amin belül szerepel az isType tag, és ennek értéke 0.
- Az XML kimenetben Link tag használatával kapcsolódik össze:
  - a fogalom a fogalom típusával:
    - meaning tag értéke 1;
  - a fogalom az eseménnyel:
    - meaning tag értéke 3,
    - az esemény a fogalom gyerekeként jelenik meg;
  - a fogalom a címkével:
    - meaning tag értéke 4,
    - a címke a fogalom gyerekeként jelenik meg a linkben.
- Az XML kimenetben két fogalom (idA, idB) közötti kapcsolat irányát a Link tag-en belül a dir tag jelöli:
  - dir értéke 1
    - idA a szülő, idB a gyerek;
  - dir értéke 2
    - idA a gyerek, idB a szülő;
  - dir értéke 3
    - idA és idB között testvéri kapcsolat van (nem közös a szülő).

- **Kapcsolat típus (Link type)**
  - A felhasználó a kapcsolathoz típust rendelhet. Egy típus több kapcsolathoz is hozzárendelhető, egy kapcsolathoz csak egy típus rendelhető.
  - A programban a kapcsolat típusa egy szövegdobozban jelenik meg, ami akkor látható, amikor az egérmutató a kapcsolatot reprezentáló vonal fölé kerül.
  - Az XML kimenetben Link tag-el jelölt rész, amin belül szerepel az isType tag, és ennek értéke 1. A típusos linknél a linkTypeID tag értéke annak a Link-nek a guid-ja ami a megfelelő típust reprezentálja.

- **Bejegyzés (Entry)**
  - A felhasználó a fogalmakhoz megjegyzéseket írhat.
  - A programban a fogalomhoz fűzött megjegyzés a Note ablakban jelenik meg.

- Az XML kimenetben minden megjegyzés Entry tag-el jelölt részbe kerül, a megjegyzés tartalma a body tag értéke. Az EntryObject tag objectID tag-jébe kerül annak a fogalomnak az id-ja, amelyik fogalomhoz tartozik az adott bejegyzés.

- **Csatolmány (Attachment)**
  - A felhasználó a fogalmakhoz csatolmányt rendelhet. Egy fogalomhoz több csatolmány tartozhat. A csatolmány többféle típusú lehet, de csak pontosan egy típusa van.
  - A programban a fogalomhoz rendelt csatolmányt a fogalom jobb oldalán megjelenő szimbólum ábrázolja.
  - Az XML kimenetben minden csatolmány Attachment tag-el jelölt részbe kerül. Minden csatolmányhoz tartozik egy Entry tag is.

- **Fájl csatolmány (File attachment)**
  - Az XML kimenetben az Attachment attachmentType tag értéke 2.
    * A format tag értéke a fájl kiterjesztése.
    * A dataLength tag értéke a fájl mérete byte-ban.

- **Mappa csatolmány (Folder attachment)**
  - Az XML kimenetben az Attachment attachmentType tag értéke 2.
    * A format tag értéke ".0".
    * A dataLength értéke 0.

- **URL csatolmány (URL attachment)**
  - Az XML kimenetben az Attachment attachmentType tag értéke 3.
    * A format tag értéke az URL TLD-je.
    * A dataLength tag értéke 0.

F.3. BrainXML kód részlet

F.3.1. lista. *Generált BrainXML kód*

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE BrainData SYSTEM "http://www.thebrain.com/dtd/BrainData1.dtd">
<BrainData>
    <Source>
        <guid>D17A4D66-7BC3-2F98-622A-CFB5661E9C78</guid>
        <name>Erdo</name>
        <personalBrainVersion>6074</personalBrainVersion>
        <telepathyVersion>10</telepathyVersion>
        <homeThoughtGuid>95BA80DB-3B15-F279-2108-C900F72FA9EF</homeThoughtGuid>
    </Source>
    <Thoughts>
        <Thought>
            <guid>95BA80DB-3B15-F279-2108-C900F72FA9EF</guid>
```

```
            <name>Erdo</name>
            <label></label>
            <isType>0</isType>
            <color>0</color>
            <accessControlType>0</accessControlType>
        </Thought>
        <Thought>
            <guid>5CCF6D0D-B182-0586-B059-22327B18F3C9</guid>
            <name>Fa</name>
            <label></label>
            <isType>0</isType>
            <color>0</color>
            <accessControlType>0</accessControlType>
        </Thought>
        <Thought>
            <guid>B5549980-B39B-3D06-2C81-D0ECE1548519</guid>
            <name>Oxigén</name>
            <label></label>
            <isType>0</isType>
            <color>0</color>
            <accessControlType>0</accessControlType>
        </Thought>
        <Thought>
            <guid>E8D2571C-EC5E-5CF7-4586-6B8C4A02E800</guid>
            <name>Bútor</name>
            <label></label>
            <isType>0</isType>
            <color>0</color>
            <accessControlType>0</accessControlType>
        </Thought>
        <Thought>
            <guid>7A9EF6AE-652A-0010-FF99-D5D429A071CD</guid>
            <name>Ház</name>
            <label></label>
            <isType>0</isType>
            <color>0</color>
            <accessControlType>0</accessControlType>
        </Thought>
        <Thought>
            <guid>8FE4D457-3C69-225D-3456-28A7C2072BB2</guid>
            <name>Növény</name>
            <label></label>
            <isType>0</isType>
            <color>0</color>
            <accessControlType>0</accessControlType>
        </Thought>
        <Thought>
            <guid>3BC59C1A-4DFD-8628-4BBB-38A0EA956AB2</guid>
            <name>Ember</name>
            <label></label>
            <isType>0</isType>
            <color>0</color>
            <accessControlType>0</accessControlType>
        </Thought>
    </Thoughts>
    <Links>
        <Link>
            <guid>6468866F-539F-7123-B729-46A1539D6078</guid>
            <idA>95BA80DB-3B15-F279-2108-C900F72FA9EF</idA>
```

```
            <idB>5CCF6D0D-B182-0586-B059-22327B18F3C9</idB>
            <dir>1</dir>
            <name></name>
            <labelForward></labelForward>
            <labelBackward></labelBackward>
            <isType>0</isType>
            <color>0</color>
            <thickness>0</thickness>
            <meaning>0</meaning>
            <linkTypeID></linkTypeID>
        </Link>
        <Link>
            <guid>BCC7FF50-C792-2FA5-F9D6-834A8CD777BF</guid>
            <idA>95BA80DB-3B15-F279-2108-C900F72FA9EF</idA>
            <idB>B5549980-B39B-3D06-2C81-D0ECE1548519</idB>
            <dir>1</dir>
            <name></name>
            <labelForward></labelForward>
            <labelBackward></labelBackward>
            <isType>0</isType>
            <color>0</color>
            <thickness>0</thickness>
            <meaning>0</meaning>
            <linkTypeID>A8E15EFD-A1D1-70BE-1B0F-323D1A7C1C2F</linkTypeID>
        </Link>
        <Link>
            <guid>F69B1DD4-7092-701E-E1F0-04D3518C5CB8</guid>
            <idA>5CCF6D0D-B182-0586-B059-22327B18F3C9</idA>
            <idB>E8D2571C-EC5E-5CF7-4586-6B8C4A02E800</idB>
            <dir>1</dir>
            <name></name>
            <labelForward></labelForward>
            <labelBackward></labelBackward>
            <isType>0</isType>
            <color>0</color>
            <thickness>0</thickness>
            <meaning>0</meaning>
            <linkTypeID></linkTypeID>
        </Link>
        <Link>
            <guid>7DF061EA-D154-0FB3-7057-4A2C5F3882B8</guid>
            <idA>5CCF6D0D-B182-0586-B059-22327B18F3C9</idA>
            <idB>7A9EF6AE-652A-0010-FF99-D5D429A071CD</idB>
            <dir>1</dir>
            <name></name>
            <labelForward></labelForward>
            <labelBackward></labelBackward>
            <isType>0</isType>
            <color>0</color>
            <thickness>0</thickness>
            <meaning>0</meaning>
            <linkTypeID></linkTypeID>
        </Link>
        <Link>
            <guid>B842147F-B893-B7A5-7E07-5A17588CDEB0</guid>
            <idA>B5549980-B39B-3D06-2C81-D0ECE1548519</idA>
            <idB>8FE4D457-3C69-225D-3456-28A7C2072BB2</idB>
            <dir>1</dir>
            <name></name>
```

```
            <labelForward></labelForward>
            <labelBackward></labelBackward>
            <isType>0</isType>
            <color>0</color>
            <thickness>0</thickness>
            <meaning>0</meaning>
            <linkTypeID>6B11739F-3FC0-8F29-156C-A954D017DD5C</linkTypeID>
        </Link>
        <Link>
            <guid>D02BCD87-D638-50A8-0B7D-A7AA9C4659EB</guid>
            <idA>B5549980-B39B-3D06-2C81-D0ECE1548519</idA>
            <idB>3BC59C1A-4DFD-8628-4BBB-38A0EA956AB2</idB>
            <dir>1</dir>
            <name></name>
            <labelForward></labelForward>
            <labelBackward></labelBackward>
            <isType>0</isType>
            <color>0</color>
            <thickness>0</thickness>
            <meaning>0</meaning>
            <linkTypeID>6B11739F-3FC0-8F29-156C-A954D017DD5C</linkTypeID>
        </Link>
        <Link>
            <guid>6B11739F-3FC0-8F29-156C-A954D017DD5C</guid>
            <dir>0</dir>
            <name>fontos számára</name>
            <labelForward></labelForward>
            <labelBackward></labelBackward>
            <isType>1</isType>
            <color>0</color>
            <thickness>0</thickness>
            <meaning>0</meaning>
            <linkTypeID></linkTypeID>
        </Link>
    </Links>
    <AttributeDatas/>
    <Entries/>
    <Attachments/>
</BrainData>
```

Ábrák jegyzéke

Táblázatok jegyzéke

Irodalomjegyzék

[1] Allora xml, dtd, xsd dokumentum konvertálás. http://www.hitsw.com/xml_utilites/xmlUtilities.html.

[2] Altova. http://www.altova.com/products.html.

[3] Bankkártya szótár. http://www.bankkartya.hu/?oldal=szotar.

[4] Brainxml-t meghatározó dtd. www.thebrain.com/dtd/BrainData1.dtd.

[5] Common Logic. http://www.obitko.com/tutorials/ontologies-semantic-web/common-logic.html.

[6] Common logic ISO szabvány. http://www.iso.org/iso/catalogue_detail.htm?csnumber=39175.

[7] Comprehensive transformations of xml schemas and xml data to rdf/owl. http://topquadrantblog.blogspot.com/2011/09/living-in-xml-and-owl-world.html.

[8] e-freight project. http://www.efreightproject.eu/.

[9] Emerald project. http://www.ml-cons.hu/.

[10] Formalizálás lépései. https://docs.google.com/viewer?url=http://www.omg.org/news/meetings/workshops/SOA_MDA_WS_Workshop_CD/06-3_Kendall.pdf&pli=1.

[11] hyperModel. http://xmlmodeling.com/hypermodel.

[12] MNB rendeletek. http://www.mnb.hu/A_jegybank/mnbhu_mnb_rendeletek.

[13] Neon. http://neon-toolkit.org/.

[14] Ontológia metamodell. http://sourceforge.net/projects/ontomodel/files/ODM/OWL%202%20Metamodel%20v1.0/.

[15] Owl 2 szintaxis dokumentáció. http://www.w3.org/TR/owl2-syntax/.

[16] OWL Api. http://owlapi.sourceforge.net/.

[17] OWL2 dokumentáció. http://www.w3.org/TR/2009/REC-owl2-profiles-20091027/.

[18] PersonalBrain. http://www.thebrain.com/.

[19] Protégé. http://protege.stanford.edu.

[20] RDF szókészlet. http://www.w3.org/TR/2004/REC-rdf-schema-20040210/.

[21] ReDeFer project. http://rhizomik.net/html/redefer/.

[22] Szemantikus web. http://www.w3.org/2006/Talks/0318-Budapest-IH/cikk.html.

[23] Topic Maps. http://en.wikipedia.org/wiki/Topic_Maps.

[24] Xsd - owl megfeleltetési táblázat. http://rhizomik.net/html/redefer/xsd2owl/.

[25] Zsámboki Balázs. A pénzügyi szabályozás hatása a banki tőkekövetelmények ciklikusságára és a pénzügyi stabilitásra. http://www.mnb.hu/Root/Dokumentumtar/MNB/Kiadvanyok/mnbhu_mnbszemle/mnbhu_szemle_cikkek/zsamboki.pdf, 2007.

[26] Tony Buzan. *How to Mind Map: The Ultimate Thinking Tool That Will Change Your Life*. Thorsons, 2002.

[27] Clark and Parsia. Pellet. http://clarkparsia.com/pellet/.

[28] Michael Denny. Ontology editor survey. http://www.xml.com/2004/07/14/examples/Ontology_Editor_Survey_2004_Table_-_Michael_Denny.pdf, 2004.

[29] Roberto García. A semantic web approach to digital rights management. http://rhizomik.net/html/~roberto/thesis/html/Methodology.html#XMLSemanticsReuse.

[30] Liu Lei Jin Longfei. Ontology Definition Metamodel, 2009.

[31] Tasi Katalin. Informális tudás formalizálása mda-hoz. http://diplomaterv.vik.bme.hu/Theses/Informalis-tudas-formalizalasa-MDAhoz, 2011.

[32] Fifi Klein. *PersonalBrain 6.0 User Guide*, October 2010.
URL: http://www.thebrain.com/support/docs.

[33] Dr. Izsó Lajos. Miller magic number seven.

[34] Maneval, Rhonda E Filburn, Monica J Deringer, Susan O Lum, and Glen D. Concept mapping. does it improve critical thinking ability in practical nursing students? http://www.ncbi.nlm.nih.gov/pubmed/21923002.

[35] Ping Wu Nabil Kamel and Stanley Y.W. Su. A pattern-based object calculus, 1993.

[36] Stuart J. Russell Peter Norvig. Mesterséges intelligencia modern megközelítésben. http://diplomaterv.vik.bme.hu/Theses/Informalis-tudas-formalizalasa-MDAhoz, 2005.