Debreceni Egyetem

Informatikai Kar

Diplomamunka

Szemantikus Webontológiák és alkalmazásuk

Témavezető:

Jeszenszky Péter

egyetemi adjunktus

Készítette:

Sólyom Márk

programtervező matematikus

Debrecen, 2009

Tartalomjegyzék

Bevezetés

Diplomamunkám célja megismertetni a szemantikus webontológiákban rejlő lehetőségeket, továbbá bemutatni egy saját ontológiát és az azt kezelő-rendszert.

Az ontológiám konkrét második világháborús hadjáratokat dolgoz fel, és olyan struktúrára épít, ami lehetővé teszi, hogy mások is újabb hadjáratokat dolgozzanak fel, vagy a meglévőket bővítsék, esetleg módosíthassák a tartalmukat (vizuális és szöveges tartalmat egyaránt). Az ontológia jelenleg is publikus, a www.swoogle.com szemantikus web kereső oldalon elérhető. A swoogle.com lehetőséget ad az ontológia továbbfejlesztésére más felhasználóknak is, illetve fórumokon történő vitákra.
Az ontológiát nagyon jó lehetne felhasználni a történelem-oktatásában, természetesen megfelelő programkörnyezettel támogatva, hogy az ontológiában szereplő tudásbázist feldolgozva lehetne szemléltetni a második világháborús hadjáratok eseményeit, pl.: digitális vetítőgéppel megjelenített térképeken a fontosabb városok kiemelve láthatóak lennének, rájuk kattintva bővebb leírást kaphatunk az ott zajló ütközetekről, stb. Továbbá nemcsak az oktatókat, tanárokat segítené, hanem a diákokat is, akik böngészhetik és felhasználhatják történelem tanulmányaikhoz ezt az ontológiát. A Hadtörténeti Múzeumot is érdekli az elkészült ontológia és a kezelő-rendszer is egyaránt, de még csak érdeklődés szintjén áll, anyagi támogatást nem tudnak ígérni hozzá.

Ezt az ontológiát egy webböngészőre épített kezelő-rendszer segítségével bárki számára egyszerűen felhasználhatóvá tettem, ahol különböző kérdéseket tehetünk fel a rendszernek, illetve végezhetünk kutatásokat, kereséseket.

A szemantikus webről általánosan

A szemantikus web elgondolás attól a Tim Berners-Lee-től származik, aki az egész World Wide Web és a hozzá kapcsolódó technológiák (URI, HTTP, HTML, stb.) atyja.

Az elképzelés alapvetően két alapötletre épül: a metaadatok[1] erőforrásokhoz kapcsolása, valamint a metaadatokon történő következtetés lehetősége. Az adat és metaadat sokszor nem választható szét élesen: ami egy szituációban adat, egy másikban lehet metaadat, és fordítva.

A szemantikus világháló elképzelés meglehetősen tágan értelmezi az internetes erőforrások fogalmát és alapkövetelményének tekinti, hogy metainformációt társíthassunk lényegében bármihez, ami egyedileg azonosítható. Ilyen például egy honlap vagy a honlap egy része, egy kép, egy videóanyag, egy tetszőleges állomány, egy hardvereszköz, vagy akár fizikai objektum is.

A metainformációk erőforrások kapcsolása mellett a szemantikus web másik lényeges alapgondolata, hogy következtetni kell tudni ezen metainformációk segítségével. Ez a gyakorlatban azt jelenti például, hogy valamilyen úton ki kell tudni deríteni, hogy például egy adott képen személyek szerepelnek, holott a metainformáció csak annyi volt, hogy Erwin Rommel és Von Thoma látható rajta. Azt senki sem állította, hogy tábornokok is vannak a képen.

A szemantikus web elképzelés valójában nem teljesen új. A weben már most is találhatunk mind implicit, mind explicit módon megadott metainformációkat, amelyek elősegítik az Internet szemantikus oldalának kiaknázását. Explicit módon, tudatosan metainformációként megadott leírásokkal is találkozhatunk.

A META nevű HTML-elem segítségével például metaadatot adhatunk meg honlapunkról. Ennek az elemnek két legfontosabb és legtöbbet használt attribútuma a *description* és a *keywords*. Előbbi segítségével honlapunk tartalmának összefoglalóját adhatjuk meg, amelyet egyes keresőrendszerek a találati listájukban mutatnak meg. Ez kétségkívül metainformáció, amely bár szabadszöveg, ember számára mégis sokatmondó és szövegalapú kereséssel egészen jól feldolgozható.

[1] Metaadat: olyan adat, mely egy masik adatról adat, pl.: egy könyv címe

Sajnos azonban a META elem csak nagyon korlátozottan használható. Egyrészt a specifiákció csak néhány attribútum használatát engedi meg és nem ad lehetőséget újak létrehozására. Másrészt sokszor nem elég kifejező a META elemmel leírt metaadat. Például a *keywords* attribútum segítségével megadhatjuk, hogy honlapunknak köze van a Második Világháborúhoz, de azt már nem, hogy pontosan mi is ez a viszony, hiszen nem mindegy, hogy a Második Világháborúval kapcsolatos képek vannak a honlapon vagy a résztvevő országok háborús áldozatainak statisztikai adatai. Az is igaz, hogy a META elemmel csak magáról a honlapról, mint egészről vagyunk képesek állításokat megfogalmazni. Így azt ugyan kijelenthetjük, hogy ki a honlap szerzője, de azt már nem, hogy a honlapon található egyik képet ki készítette.

Fontos látni, hogy számos, az Internettől némileg távolabb eső területen is alkalmaznak metainformációkat, például MP3 állományok esetén az *id3v1* és *id3v2* elemeket.

A szemantikus web irányzat esetében a nagy ötlet az, hogy ezen metainformációkat egységesen és strukturált módon köthessük tetszőleges erőforráshoz. A cél az, hogy ugyanúgy társíthassunk metainformációt egy Word állományhoz, mint egy MP3 formátumú zenéhez vagy egy honlaphoz.

A metaadatok hasznossága

A metainformációk használata egy általános eszköz arra, hogy jelentést adjunk különböző erőforrásokhoz. Amennyiben képesek vagyunk gépek számára feldolgozható és érthető módon leírni egy-egy erőforrás jellemzőit, óriási lépést tehetünk az intelligensebb kereshetőség felé. Metaadatok megadásával kereshetővé válhat a mély web, elérhetjük az adatbázisokban, a flashes honlapok tömegében tárolt információt, találatként olyan képeket kaphatunk vissza, amelyekről metaadatok nélkül sosem lennénk képesek automatikus eszközökkel kideríteni, hogy mit tartalmaznak.

Ehhez mindössze annyi szükséges, hogy az adatbázisok metaadatokat szolgáltassanak magukról, például: „ebben az adatbázisban Sony digitális fényképezőgépek specifikációi vannak”, vagy „a Wermacht páncélos állománya 1941-ben”. A metaadat részletezettsége különböző lehet.

Valójában a metaadatok minden olyan esetben segíthetnek, amikor az erőforrás – például egy honlap – igazi tartalma nem derül ki magából a honlapból egy keresőrobot számára. A metainformációk megadásával kapcsolatos egyik legfőbb feladat, hogy erre a célra egységes leírást használjunk.

A szemantikus világháló elképzelésének alapnyelve az RDF, melyet arra terveztek, hogy segítségével egyszerű módon fogalmazhassunk meg kijelentéseket a Web erőforrásairól (röviden: webforrásokról), például weblapokról.

Az RDF nyelv

Az XML jelentős lépést tett a webes tartalmak gépi feldolgozhatóságának irányába azzal, hogy szabványos adatcsere-formátumként az alkalmazások együttműködésének szintaktikus oldalát megfelelő módon biztosítja.

Az XML használatával sem lehetséges azonban az, hogy alkalmazások alkalmazásokkal kommunikáljanak úgy, hogy előtte ne kelljen egyeztetni az átvitt információ szemantikáját. Hiába az XML-formátum, ha az egyik fél félreérti a dolgot, és ezer forintot utal át százezer helyett, azért, mert az egyikük ezresekben számol, a másikuk nem.

A világháló következő generációja, a szemantikus web, pontosan azt célozza meg, hogy az átvitt, tárolt információ jelentése is egyértelmű legyen a felek számára. Amennyiben ez így van, akkor persze egy keresőrobot is „érteni” fogja bizonyos internetes erőforrások tartalmát. Azt szokták mondani, hogy az információt a gépek számára feldolgozható és érthető formában kell tárolni. Előbbire az XML, az utóbbi alapjainak megteremtésére pedig az RDF (Resource Description Framework – Erőforrás-Leíró Keretrendszer) szolgál.

Az RDF arra az elvre épül, hogy a leírásra kerülő dolognak több *tulajdonsága*, a tulajdonságoknak pedig értéke van, és hogy az erőforrások leírhatók a fentiekhez hasonló kijelentésekkel, amelyek specifikálják az erőforrások tulajdonságait és a tulajdonságok értékeit. Az RDF egy meghatározott terminológiát használ az ilyen kijelentő mondatok különböző részeinek a megnevezésére. Például a mondatnak azt a részét, amelyik azt azonosítja, akiről/amiről az állítás szól, *alanynak* nevezi (a példánkban ez a weblap). Azt a részt, amelyik az alany tulajdonságait, jellemzőit azonosítja (a példánkban *creator*, *creation-*

date, *language*), *állítmánynak*, és azt a részt, amelyik e tulajdonságok értékeit azonosítja, *tárgynak* nevezi. Az RDF nyelvhez szorosan kapcsolódik az URI fogalma.

Az URI-k és szerepük

Az RDF világában erőforrás minden, ami rendelkezik Általános Erőforrás-azonosítóval (Universal Resource Identifier), röviden URI-val. Az URI-k rövid literálok (karaktersorozatok), amelyek általában weben található erőforrásokat azonosítanak. Erőforrás többek között egy weblap, a weblap egy része, egy kép, egy tetszőleges állomány, hanganyag, erőforrások egy csoportja stb. Fontos, hogy az URI-k – és így az RDF – szempontjából egy erőforrás nem feltétlenül kötődik a webhez; akár egy asztalon lévő kávéscsésze is erőforrás, feltéve, hogy társul hozzá egy URI.

Az URI-knak alapvető szerepük van a szemantikus világháló elképzelésben. A legfontosabb dolog, hogy segítségükkel egyértelmű állításokat fogalmazzunk meg, hiszen az URI-k egyértelműen azonosítanak erőforrásokat. Bárhol is legyen két metaadat-leírás, ha ugyanazon URI-t használja, amely például egy személyt azonosít, akkor ugyanazon erőforrásról jelentünk valamit. Például egyrészt azt, hogy az illető szeme kék, másrészt, hogy magassága 175 cm.

Az URI-ra tekintsünk két példát:

- file:///c:/examples/vehicles.rdf
- http://www.origo.hu/index.html

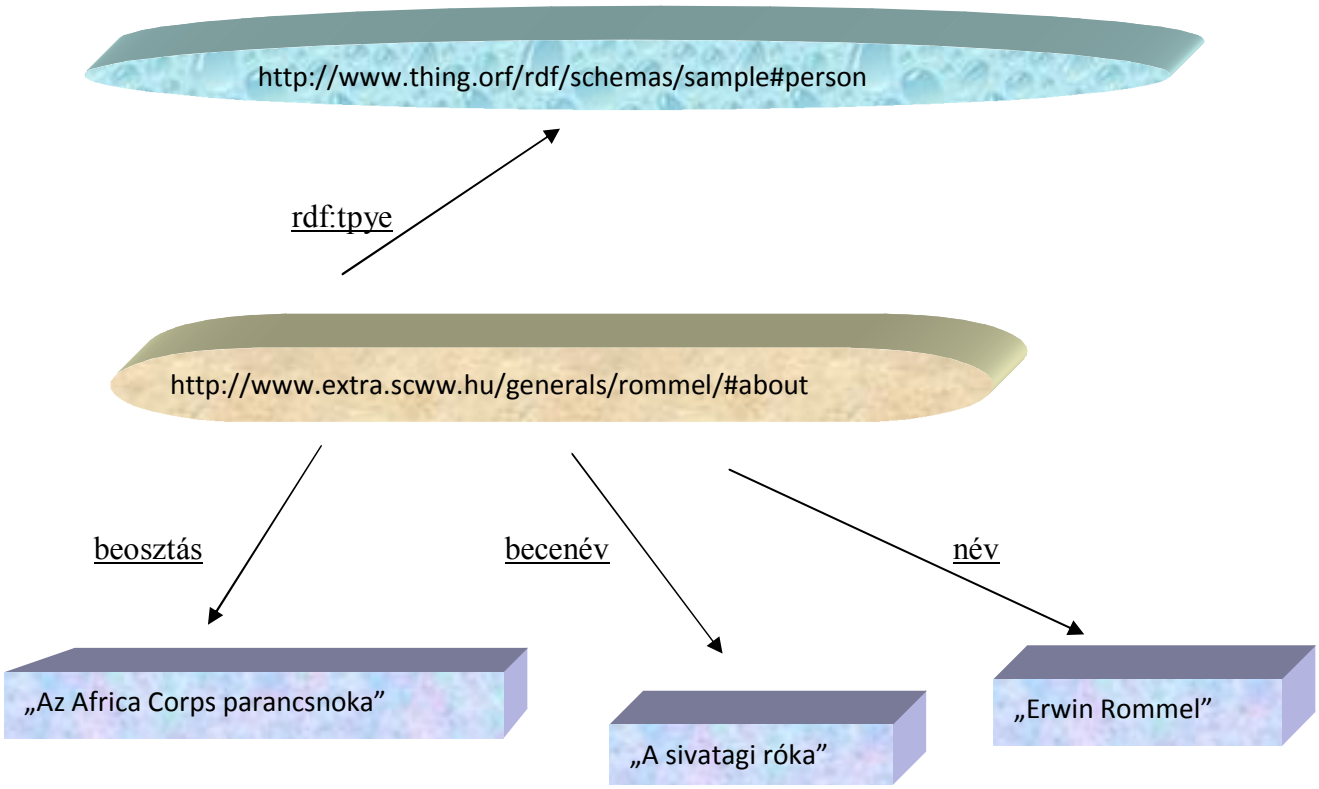
Az URI-knak két „fajtáját” különböztetjük meg. Az *„abszolút” URI-k* egyértelműen azonosítanak erőforrásokat. A *„relatív” URI-knak* csak egy adott környezetben van értelmük, nevezetesen akkor, ha rendelkezésre áll egy ún. bázis URI. Az, hogy mi ez a bázis URI, több dologtól is függhet, a szabvány pontosan definiálja ezeket, prioritásukkal együtt. Például, amennyiben egy adott dokumentum alkalmas arra, hogy definiálja a saját magán belül érvényes bázis URI-t, és definiálja is, akkor ez lesz a bázis URI. Ha nincs ilyen, akkor az az URI lesz a bázis, amely URI-n elérhető az adott dokumentum.

A bázis URI segítségével a „relatív” URI-kat ezek után mindig feloldjuk, és „abszolút” azonosítót készítünk belőlük. Szintaxisukat tekintve a „relatív” URI-kat legfőképpen az különbözteti meg az abszolút azonosítóktól, hogy hiányzik az ún. sémaazonosító az elejükről.

A sémaazonosítók kettősponttal vannak elválasztva az URI többi részétől és arra szolgálnak, hogy definiálják az URI által azonosított erőforrás elérési módját. Egy „relatív” URI feloldása közben a létrejövő abszolút URI a sémát a bázis URI-tól örökli.

Mind az „abszolút”, mind a „relatív” URI-khoz kapcsolódhat egy opcionális kiegészítő, amelyet a # jel választ el az URI törzsétől. Az RDF elképzelés alapötlete az, hogy az URI-val azonosított erőforrásokat tulajdonságok segítségével más erőforrásokkal vagy éppen közönséges literálokkal köti össze. Az összefüggéseket RDF-gráfok segítségével írhatjuk le.

Tekintsük az alábbi példát egy RDF-gráfra:



A gráf a következőket állítja:

- az erőforrás típusa http://www.thing.orf/rdf/schemas/simple#person
- neve Erwin Rommel
- beceneve „A sivatagi róka”
- beosztása az Africa Corps parancsnoka

A metaadatok leírhatósága érdekében az RDF sepcifikáció definiál egy halmazelméleti alapokon nyugvó adatmodellt, az RDF *adatmodellt*. A modellben az alábbi halmazok definiáltak:

- Erőforrások (Resources): az összes olyan elem halmaza, amire egy RDF-kijelentés vonatkozhat. A halmaz elméletileg minden elképzelhető entitást tartalmaz (mivel mindennek lehet URI-ja).
- Tulajdonságok (Properties): a halmaz elemeit tulajdonságoknak hívjuk, amik erőforrásokhoz kapcsolható jellemzők. A tulajdonságok maguk is RDF-erőforrások, azaz őket is URI-k azonosítják. Minden tulajdonságnak van jelentése, meghatározható, hogy milyen erőforráshoz kapcsolható és hogy milyen értéket vehet fel, valamint, hogy milyen viszonyban van más tulajdonságokkal.
- Literálok (Literals): elemi literálok, azaz karaktersorozatok.
- Kijelentések (Statements): a halmaz elemei hármasok vagy más néven kijelentések, amelyek alanyból, állítmányból és tárgyból állnak. Az alany tetszőleges RDF-erőforrás, a predikátum egy tetszőleges RDF-tulajdonság, míg a tárgy (más néven a tulajdonság értéke) tetszőleges RDF-erőforrás vagy literál lehet. Ennek megfelelően egy RDF-kijelentés nem más, mint három összetartozó, URI-val azonosított erőforrás vagy két erőforrás és egy literál.

A webontológia jellemzése

A szemantikus web elképzelés célja a világhálón elérhető információk automatikus feldolgozása. Az automatikus feldolgozáshoz elengedhetetlen, hogy az adatok jelentést nyerjenek. A szemantika definiálását célozzák meg a webontológiák.

Maga az ontológia eredetileg filozófiai szakkifejezés, a tudásalapú rendszerekben az ontológiák egy szakterület vagy közismereti terület fogalmait, ezek összefüggéseit definiálják. Egy ontológia az adott terület fogalomrendszerének olyan formális specifikációja, amely lehetővé teszi az automatikus következtetést.

A webontológiák a tudás megosztását teszik lehetővé a világhálón, míg a hagyományos tudásalapú rendszerek központosítottak.

Webontológia nyelvek

Az **RDF** elképzelés tudásreprezentációt tesz lehetővé, webes erőforrások írhatók le általa egyszerű kijelentések formájában, az *RDF Séma* pedig a szemantika definiálást célozza meg, bevezeti az osztályok fogalmát, lehetővé teszi az osztályok és tulajdonságok közötti kapcsolat leírását.

Az RDF szókészlet-leíró nyelve, az *RDF Séma*, nem más, mint az RDF szemantikai bővítménye. Olyan tulajdonságokat és osztályokat reprezentáló erőforrásokat definiál, amelyekkel tulajdonságokat és osztályokat lehet leírni, beleértve ezek kapcsolatait is természetesen. Az RDF szókészlet-leíró nyelv osztály- és tulajdonságfogalma némileg hasonló az olyan objektumorientált programozási nyelvek típus-rendszeréhez, mint pl. a Java. Az RDF abban különbözik sok ilyen rendszertől, hogy ahelyett, hogy az *osztályokat* definiálná azokkal a tulajdonságokkal, amelyek az egyedeit jellemzik, inkább a *tulajdonságokat* definiálja azokkal az erőforrás-osztályokkal, amelyekre ezek a tulajdonságok vonatkoznak.

Az RDF tulajdonság-centrikus közelítésének az egyik előnye az, hogy bárki számára lehetővé teszi egy erőforrás leírásának a bővítését – s ez nem más, mint a Web egyik architekturális alapelve. Az RDF szókészlet-leíró stratégiája ugyanis annak elismerésén alapszik, hogy sokféle technika lehetséges, amellyel az osztályok és tulajdonságok jelentése leírható.

Az *RDF Séma* szókészlet definiálása a http://www.w3.org/2000/01/rdf-schema# URI hivatkozással azonosított névtérben valósul meg, a továbbiakban az *rdfs* névtér előtaghoz ez a névtér név hozzárendelt. Kényelmi okokból a továbbiakban URI hivatkozások ábrázolása minősített nevekkel történik.

Az erőforrások csoportokra oszthatók, amelyeket *osztályoknak* hívunk. Az osztály tagjait az osztály egyedeinek vagy példányainak nevezzük. Az osztályok maguk is erőforrások. Ezeket RDF URI hivatkozásokkal azonosítjuk, és RDF tulajdonságok segítségével írjuk le. Az *rdf:type* tulajdonságot használhatjuk annak deklarálására, hogy egy erőforrás melyik osztálynak (RDF "típusnak") az egyede.

Az RDF megkülönbözteti az osztály fogalmát az egyedeiből álló halmaz fogalmától. Minden osztályhoz egyedek egy halmaza tartozik, amelyet az osztály kiterjedésének (extension) nevezünk. Két osztálynak lehet azonos az egyedhalmaza, ezek mégis lehetnek különböző osztályok.

Például az adóhivatal osztályként definiálhatja azoknak az embereknek a csoportját, akik mondjuk e dokumentum szerkesztőjével azonos területen laknak. A postahivatal pedig definiálhat egy másik osztályt, mely olyan embereket csoportosít, akiknek az irányítószáma azonos e dokumentum *szerzőjének* az irányítószámával. Könnyen előfordulhat, hogy ennek a két osztálynak azonosak a tagjai, és mégis lehetnek eltérő tulajdonságaik. Minimum abban a tulajdonságában eltérő a két osztály, hogy az egyiket az adóhivatalban, a másikat pedig a postahivatalban definiálták, tehát *fogalmilag* különböző osztályokról van szó (amelyeknek az azonosítója is különböző).

Egy osztály lehet *tagja* a saját osztálykiterjedésének, és lehet *egyede* is sajátmagának. Az erőforrásoknak az csoportja, amely az *RDF Séma* osztályaiból áll, maga is egy osztály, amelynek a neve *rdfs:Class*. Az adattípusok is osztályok. Egy adattípus-osztály egyedei az adattípus értékterének a tagjai.

Ha egy C osztály *alosztálya* egy C' osztálynak, akkor C minden egyede C'-nek is egyede. Az *rdfs:subClassOf* tulajdonságot használhatjuk annak kijelentésére, hogy egyik osztály egy másiknak az alosztálya. A *főosztály* kifejezést az alosztály inverzének tekintjük. Ha most úgy fogalmazunk, hogy a C' osztály *főosztálya* a C osztálynak, akkor is igaz marad, hogy C minden egyede C'-nek is egyede.

Az *RDF Séma* szókészlet osztályai:

- *rdfs:Resource*
    - Példánya az összes erőforrás
    - Az összes osztály alosztálya ennek az osztálynak
    - Az *rdfs:Resource* az *rdfs:Class* osztály példánya
- *rdfs:Class*
    - Példányai az osztályokat reprezentáló erőforrások
    - Az *rdfs:Class* az *rdfs:Class* osztály példánya
- *rdfs:Literal*
    - Példányai a tipizált és típus nélküli literálok
    - Az *rdfs:Literal* az *rdfs:Class* osztály példánya, valamint alosztálya az *rdfs:Resource* osztálynak
- *rdfs:Datatype*
    - Példányai adattípusok
    - Az *rdfs:Datatype* példánya és alosztálya az *rdfs:Class* osztálynak
    - Az osztály minden példánya alosztálya az *rdfs:Literal* osztálynak
- *rdf:XMLLiteral*
    - Példányai az XML literálok
    - Az *rdf:XMLLiteral* példánya az *rdfs:Datatype* osztálynak
    - Alosztálya az *rdfs:Literal* osztálynak
- *rdf:Property*
    - Példányai tulajdonságok
    - Az *rdf:Property* az *rdfs:Class* példánya

Az *RDF Séma* szókészlet tulajdonságai: a P' tulajdonság a P tulajdonság altulajdonsága, ha minden olyan alany-tárgy erőforrás párt, amelyet összekapcsol a P' tulajdonság, összekapcsol a P tulajdonság is. A P tulajdonság a P' tulajdonság főtulajdonsága, ha minden olyan alany-tárgy erőforrás párt, amelyet összekapcsol a P' tulajdonság, összekapcsol a P tulajdonság is. Mindkét tulajdonság tranzitív. Az *RDF Séma* nem definiál olyan tulajdonságot, amely főtulajdonsága az összes többi tulajdonságnak. A tulajdonságok nevei általában kisbetűvel kezdődnek.

Az *RDF Séma* tulajdonságai felsorolva:

- *rdfs:domain:* az *rdf:Property* osztály egyede, és annak kijelentésére használjuk, hogy egy adott tulajdonság csak azokra az erőforrásokra alkalmazható, amelyek egy vagy több megadott osztálynak az egyedei.
- *rdfs:range:* az *rdf:Property* osztály egyede, és annak kijelentésére használjuk, hogy egy tulajdonság értékei csak a megadott osztály(ok) egyedeiből kerülhetnek ki.
- *rdf:type:* az *rdf:Property* osztály egyede, és annak kijelentésére használjuk, hogy egy erőforrás melyik osztálynak az egyede.
- *rdfs:subClassOf:* az *rdf:Property* egyede, és annak kijelentésére használjuk, hogy az egyik osztály minden egyede egy másik osztálynak is egyede.
- *rdfs:subPropertyOf:* az *rdf:Property* osztály egyede, és annak kijelentésére használjuk, hogy minden olyan altulajdonság amelyik összekapcsolható az egyik tulajdonság segítségével, az összekapcsolható egy másik segítségével is.
- *rdfs:label:* az *rdf:Property* osztály egyede, és arra használjuk, hogy megadjuk vele egy erőforrás nevének ember által olvasható változatát.
- *rdfs:comment:* az *rdf:Property* osztály egyede, és arra használjuk, hogy megadjuk vele egy erőforrás ember által olvasható leírását.

Az *RDF Sémára* épülő világban egy tulajdonságot definiálunk annak a tükrében, hogy az milyen osztályok példányaira alkalmazhatjuk. Az osztályokat úgy definiálhatjuk, hogy megmondjuk, milyen jellemzővel rendelkező példányok az elemei. Az *RDF Sémához* hasonló tulajdonságközpontú nyelvek jobban illeszkednek a szemantikus világháló által megcélzott „bárki mondhat bármit” filozófiához.

Az *RDF Séma* kifejezőereje azonban nem elég, hiányosságai és a szemantikus világhálón felhasználható ontológianyelvek iránti növekvő igény vezetett el számos ontológianyelv kialakulásához.

Az ezredforduló idején a két legjelentősebb nyelv az OIL (Ontology Interface Layer) és a DAML-ONT (DARPA Agent Markup Language) volt. Előbbit a Manchesteri egyetem, utóbbit az Amerikai Védelmi Minisztérium (DoD) fejlesztette ki. Mindkét nyelvet leíró logikai tudás birtokában készítették, az OIL nyelvet például kifejezetten a FaCT[2] következtetőre szabták és ennek megfelelően a SHIQ[3] nyelvosztályt valósítja meg.

2001 végén született meg a DAML+OIL nyelv, amelyet a két nyelv fő fejlesztői egy közös európai és amerikai ad hoc bizottság keretében hoztak létre, és amely a DAML és OIL nyelvek előnyös tulajdonságait ötvözte. A DAML+OIL nyelvet benyújtották a W3C-irodához, hogy az alapjait képezze az akkor már tervezett hivatalos webes ontológianyelvnek, a *Web Ontology Language* (OWL)-nek.

[2] FaCT: ontológiák konzisztencia-ellenőrzésre szolgáló logikai osztályozó rendszer

[3] SHIQ: az egyik legnagyobb kifejezőerejű leíró logikai nyelv

Az OWL, mint webontológiai nyelv

Az OWL rövidítés nem elírás és nem is véletlen, hanem egy szóvicc. Az *owl* kifejezés angolul baglyot jelent. A baglyok ugyebár bölcsek, ezért esett erre a készítők választása.

Az OWL nyelv 2004. február 10-én lett hivatalos W3C-ajánlás. Ekkor váltak ajánlássá az RDF és az RDF séma nyelvek is. Ez nem a véletlen műve, hiszen egymásra épülő technológiákról van szó. Az RDF egy általános keretrendszer, mely segítségével egyszerű módon fogalmazhatunk meg kijelentéseket a Web erőforrásairól, mint ahogy azt már korábban olvashattuk. Az RDF séma egy RDF-szintaxison alapuló pehelysúlyú ontológianyelv. Az OWL ezt az irányvonalat követi és egy nehézsúlyú, RDF-alapú ontológianyelvként funkciónál.

Az OWL tehát a DAML+OIL nyelv egy kicsit módosított és átszabott változata, amelyet olyan alkalmazások esetére javasol a W3C, amikor az RDF séma által nyújtott képességek nem elegendőek. Az OWL-t tekinthetjük egy URI halmaznak az RDF sémához hasonlóan. A halmazba tartozó URI-k olyan erőforrásokat azonosítanak, amelyek segítenek minket egy terminológiai rendszer elkészítésében. Ezen erőforrások jelentése tisztázott és legfőképpen szabványos, segítségükkel leírhatjuk osztályok diszjunktságát, ekvivalenciáját, tulajdonságok tranzitivitását, kardinalitását (számosságkorlátozás) stb.

A világháló egy osztott környezet, az OWL-nek támogatnia kell az információk összegyűjtését osztott forrásokból, ezt részben lehetővé teszi, hogy az ontológiák összekapcsolhatóak egymással (például importálhatják egymást).

Az OWL az RDF sémához hasonlóan él a nyíltvilág-feltételezéssel. Egy OWL nyelvű ontológiát készítő személy számára ez például azt jelenti, hogy egy általa „definiált” osztályhoz bárki más megadhat további tulajdonságokat. Az OWL nyelv monoton, azaz minden olyan következtetést, amelyet képesek voltunk levonni egy bizonyos konzisztens állapotban, le kell tudnunk akkor is, ha olyan új információt adunk hozzá a rendszerhez, amellyel az konzisztens marad. Ennek megfelelően OWL-ben törölni sosem tudunk állításokat, azokat sem, amiket esetleg mások a tudtunk nélkül adtak hozzá a tudásbázishoz.

Az OWL nem él az egyedinév-feltételezéssel (*Unique Name Assumption*), ez ugyanis nem lenne megfelelő webes környezetben, ahol egyes erőforrásokra sokféleképpen is hivatkozhatunk. Az egyedinév-feltételezés hiányában tehát különböző URI is azonosíthatja ugyanazt a példányt, amely nem az alapértelmezett eljárás a legtöbb leíró logikai következtetőrendszernél.

Az OWL résznyelvei

Az OWL kifejezőerő szempontjából több résznyelvet definiál. Ezek rendre az OWL Full, az OWL DL és az OWL Lite. Az, hogy egy konkrét OWL-leírás melyik résznyelvbe esik, attól függ, hogy mely OWL-konstrukciókat és hogyan használja fel.

Az OWL Full (a teljes nyelv) a leíró logikákkal ellentétben nem választja el élesen az egyedeket, illetve az adattípusokhoz tartozó értékeket tartalmazó halmazokat. Annak, hogy egy elem benne lehet mindkét halmazban, számos nagyon fontos következménye van.

Az OWL Full továbbá semmilyen megkötést nem támaszt az OWL konstrukciók használatával szemben. Minden RDF-konstrukció használható minden olyan módon, amit az RDF megenged. Alapvető példa lehet, hogy OWL Full esetén egy osztályról minden további nélkül kijelenthetjük, hogy példánya egy másik osztálynak. Ennek semmi akadálya, mindössze két megfelelő RDF-hármasra van szükségünk:

{[URI1], [rdf:type], [owl:Class]}

{[URI1], [rdf:type], [t:Ember]}

Az első hármas kijelenti, hogy az URI1 URI-val azonosított erőforrás OWL-osztály, a második azt, hogy ugyanezen erőforrás egy ember. Ezt egy RDF séma esetén is megtehettük volna, látszik azonban, hogy a következtetés szempontjából ez komoly kérdéseket vet fel, és éppen ez az oka annak, hogy például az OWL DL nyelv nem engedi meg ezt a fajta használatot. OWL DL-ben igaz az, hogy az OWL-osztályok valódi részhalmazai az RDF séma osztályainak, és így például azon osztályok, amelyek példányok is egyben, nem OWL DL osztályok. OWL Full esetén azonban az owl:Class ekvivalens az rdfs:Class erőforrással, azaz az OWL és RDF séma osztályfogalma egybeesik.

Az OWL Full engedékenységének természetesen megvan a maga előnye. Gyakorlati tapasztalatok alapján tudjuk, hogy az emberi elme nagyon szívesen használ a fentihez hasonló

metamodellezési konstrukciókat a világ leírására. Például a Boeing-747-es utasszállító vagy B-17 Flying Fortress bombázógép erőforrásra egyrészt szeretnénk úgy tekinteni, mint repülőgépek egy halmazára. Ennek a két osztálynak a példányai konkrét repülőgépek, azok, amelyeket az egyes légitársaságok használnak, illetve az Amerikai Légierő a Második Világháború idején használt. Ezekről kijelentéseket tehetünk, leírhatjuk állapotukat, utolsó karbantartásuk idejét, harci bevetéseik számát, hány utast szállítottak, stb. Másrészt azonban ezekre az erőforrásokra szeretnénk úgy tekinteni néha, mint a repülőgéptípusok osztály egy példányára.

Az OWL FULL nyelv nem feleltethető meg egyetlen leíró logikai nyelvnek sem. Az OWL Full nyelvű dokumentumokon való következtetéshez elsőrendű következtetési képességek kellenek, és ebben az esetben is számos trükkre van szükség ahhoz, hogy egyáltalán elsőrendű logikai állításokká konvertáljuk az OWL-dokumentumot.

Az OWL DL nyelv minden OWL-konstrukcióval rendelkezik, de használatára bizonyos megkötések érvényesek. Az OWL DL megköveteli, hogy az osztályokat, adattípusokat, egyedtulajdonságokat, adattípus-tulajdonságokat, annotációs tulajdonságokat, ontológia-tulajdonságokat, egyedeket, értékeket és a beépített osztályokat és tulajdonságokat azonosító URI-k páronként különbözőek legyenek. Ennek megfelelően egy erőforrás nem lehet egyszerre osztály is és egyed is, egy tulajdonságról mindig pontosan lehet tudni, hogy egyedeket egyedekkel vagy egyedeket értékkel köt össze stb.

Míg az OWL DL minden OWL-konstrukciót használhat, amit az OWL Full (legfeljebb korlátozott módon), addig az OWL Lite résznyelvből bizonyos konstrukciók teljesen hiányoznak. Az OWL Lite alapötlete, hogy az RDF sémánál kifejezőbb ontológianyelv legyen, de olyan, ami csak egy kis lépést jelent az alkalmazásírók szempontjából az „igazi” ontológianyelvek irányába. Az elképzelés az, hogy a létező RDF sémát feldolgozni képes alkalmazások idővel az OWL Lite dokumentumokat is meg fogják érteni és mintegy ugródeszkát használva innen lépnek tovább a még kifejezőbb OWL-résznyelvek felé.

Az OWL-osztályleírások fajtái

Az OWL hatfajta osztályleírás konstrukciót különböztet meg annak megfelelően, hogy milyen módon hoztuk létre, s milyen módon írtuk le azokat, az osztályleírás konstrukciók a következőek:

I. megnevezett osztály (named class)
II. felsorolásos osztály (enumeration class)
III. tulajdonságkorlátozással megadott osztály (property restriction)
IV. osztályok metszeteként megadott osztály (intersection)
V. osztályok uniójaként előállított osztály (union)
VI. egy osztály komplemenseként előállított osztály (complement)

A *megnevezett osztályok* az egyedüliek, amelyekhez társul URI. A többiek névtelen osztályokat definiálnak, amelyek megkötéseket tesznek arra vonatkozóan, hogy mely objektumok lehetnek példányaik. Az OWL előre definiál két megnevezett osztályt. Az *owl:Thing* egy szuperosztály, OWL DL esetén minden egyed beletartozik ebbe az osztályba, és ennek megfelelően az *owl:Thing* minden OWL-osztály szülőosztálya. A másik ilyen beépített osztály az *owl:Nothing*. Ennek az osztálynak nincsen egyetlenegy példánya sem, és ennek megfelelően az össze OWL-osztály alosztálya.

Felsorolásos osztályt a következőképpen adhatunk meg: az *owl:oneOf* tulajdonság segítségével egy osztályt úgy konstruálhatunk meg, hogy konkrétan megadjuk, mely példányok tartoznak bele. Az *owl:oneOf* tulajdonság értelmezési tartománya az *owl:Class* osztály, értékkészlete egy RDF-lista vagy más néven kollekció. Ez utóbbi megadásához szinte mindig az *rdf:parseType* attribútumot használjuk, amely segítségével nagyon kényelmesen adhatunk meg RDF-kollekciót. Tekintsük az alábbi példát:

Az alábbi példában megadhatjuk a Föld kontinenseit tartalmazó *felsorolásos osztályt*:

```
<owl:Class>
  <owl:oneOf rdf:parseType=”Collection”>
        <owl:Thing rdf:about=”#Afrika” />
        <owl:Thing rdf:about=”#Amerika” />
        <owl:Thing rdf:about=”#Ausztrália és Óceánia” />
```

```
        <owl:Thing rdf:about="#Ázsia" />
        <owl:Thing rdf:about="#Európa" />
        <owl:Thing rdf:about="#Anktartisz" />
    </owl:oneOf>
</owl:Class>
```

Az *owl:oneOf* konstrukció egyáltalán nem használható az OWL Lite nyelvben.

A *tulajdonságkorlátozással megadott osztályok* olyan névtelen osztályok, amelyekbe tartozó példányok egy megadott tulajdonságkorlátozásnak tesznek eleget. Egy tulajdonságkorlátozás megnevez egy tulajdonságot, majd az értékére vagy a számosságára vonatkozó megkötést tesz.

Az ilyen osztályok általános alakja a következő:

```
<owl:Restriction>
<owl:onProperty rdf:resource="P" />
    a tulajdonság értékének vagy számosságának korlátja
</owl:Restriction>
```

A „P" a megnevezett tulajdonság URI-ja, míg az *owl:Restriction* az *owl:Class* alosztálya.

Értékkorlátok

Ismerkedjünk meg az *értékkorlátokkal*, melyeket az *owl:Restriction* osztályok létrehozásához használhatunk fel. Értékkorlátok a következőek lehetnek: *owl:allValuesFrom* , *owl:SomeValuesFrom* , *owl:hasValue* tulajdonság.

Az *owl:AllValuesFrom* tulajdonság jobb oldala egy OWL osztályleírás vagy egy *owl:dataRange*, azaz értékek halmaza. Az előbbi esetben a korlát jelentése az, hogy az osztályba azon példányok kerülnek, amelyekre igaz, hogy az összes hozzájuk kapcsolódó P tulajdonság jobb oldalán csak a megadott osztályleírásba (egészen pontosan az osztályleírás által meghatározott halmazba) tartozó egyedek állnak.

Nézzük meg az alábbi példát:

```
<owl:Restriction>
    <owl:onProperty rdf:resource=”#tengerpart/>
    <owl:allValuesFrom rdf:resource=”#Kavicsos”/>
</owl:Restriction>
```

Példánkban egy olyan osztályt definiáltunk, amelybe azon egyedek (mondjuk országok) tartoznak, melyeknek minden tengerpartja kavicsos, pontosabban minden tengerpartja beletartozik a „Kavicsos” osztályba. Fontos, hogy a definíció értelmében azon egyedek is idetartoznak, akiknek nincs is tengerpartjuk.

Az *owl:allValuesFrom* konstrukció az OWL Lite nyelvben csak korlátozottan használható. OWL Lite esetén csak megnevezett osztály, illetve adattípus azonosítója állhat az *owl:allValuesFrom* jobb oldalán.

Az *owl:someValuesFrom* tulajdonság nagyon hasonlít az *owl:allValuesFrom* tulajdonságra. Az *owl:someValuesFrom* jobb oldala egy OWL-osztályleírás vagy egy *owl:dataRange*. A korlát jelentése a következő: az *owl:Restriction* osztályba azon példányok kerülnek, amelyekhez van egy olyan P tulajdonság, amelynek jobb oldalán a megadott osztályleírásba tartozó egyede áll. Ha tekintjük az owl:allValuesFrom korlátra hozott példánkat módosítva:

```
<owl:Restriction>
    <owl:onProperty rdf:resource=”#tengerpart/>
    <owl:someValuesFrom rdf:resource=”#Kavicsos”/>
</owl:Restriction>
```

Van olyan ország, ami példánya ennek osztálynak, akkor annak van olyan tengerpartja, ami példánya a „Kavicsos” osztálynak. Fontos, hogy azon példányok, amiknek nincs tengerpartja, nem lesznek példányai a fenti osztálynak. Az *owl:someValuesFrom* konstrukció az OWL Lite nyelvben szintén csak korlátozott módon használható.

Az *owl:hasValue* korlát jobb oldala egy egyed vagy egy érték. A korlát jelentlése, hogy az osztályba azon példányok kerülnek, amelyekhez van legalább egy olyan P tulajdonság, amelynek jobb oldalán a megadott egyedekkel vagy értékkel szemantikailag ekvivalens egyed vagy érték található.

Két egyed szemantikailag ekvivalens, ha ugyanazon URI azonosítja őket, vagy ki lett jelentve róluk, hogy azonosak. Két érték szemantikailag ekvivalens, ha szintaxistól függetlenül ugyanazt az értéket reprezentálják (például a 2.5 és a 2.50 érték szemantikailag ekvivalens). Példaként tekinthetjük a következőt:

```
<owl:Restriction>
    <owl:onProperty rdf:resource=”#tengerpart/>
    <owl:hasValue rdf:resource=”#Aranypart”/>
</owl:Restriction>
```

Ebben az osztályba azok az egyedek kerülnek, amelyeknek tengerpartja „Aranypart”. Az *owl:hasValue* konstrukció nem használható OWL Lite nyelvben.

Számosságkorlátok

Az *owl:Restriction* osztályok létrehozásához felhasználhatunk háromféle számossági korlátot is. A számossági korlátok segítségével olyan osztályokat definiálhatunk, amelyekbe tartozó egyedekre a hozzájuk kapcsolódó egyedek darabszámára tehetünk megkötést.

Míg az RDF-ben semmi sem köti meg, hogy egy erőforrás hány RDF-állítás alanya, azaz hány él indul ki belőle az RDF-gráfban, vagy másképpen: hány tulajdonság kapcsolódik hozzá. Ez inkább előny, mint hátrány, hiszen például így válik lehetővé a „bárki mondhat bármit” elv. Az OWL esetében is ugyanez a helyzet, a számosságkorlátozások csak arra vannak hatással, hogy bizonyos erőforrások mely osztályokba tartozzanak, arra nem, hogy globálisan hány él és milyen él indulhat ki az erőforrásból.

Az *owl:maxCardinality* tulajdonság jobb oldala egy érték, egészen pontosan egy nemnegatív egész. Az *owl:maxCardinality* korlát jelentése, hogy az *owl:Restriction* osztályba azon példányok kerülhetnek, amelyekhez legfeljebb a megadott számú olyan P tulajdonság tartozik, amelyek jobb oldala szemantikailag különböző. Az alábbi példával definálható egy kis ország hadserege, ha legfeljebb 3 hadosztályból áll:

```
<owl:Restriction>
    <owl:onProperty rdf:resource=”#hadosztály(division)”>
    <owl:maxCardinality rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
    >3</owl:maxCardinality>
```

```
    </owl:onProperty>
</owl:Restriction>
```

Az *owl:maxCardinality* OWL Lite esetén csak azzal a korlátozással használható, hogy 0 vagy 1 értéket adjuk meg benne.

Az *owl:minCardinality* korlátozás jobb oldala nemnegatív egész, jelentése: az osztályba azon példányok kerülnek, amelyekhez legalább megadott számú olyan P tulajdonság tartozik, amelyek jobb oldala szemantikailag különböző. A korlát segítségével definiálható egy nagyobb ország hadserege, ha legalább 90 hadosztályból áll:

```
<owl:Restriction>
    <owl:onProperty rdf:resource="#hadosztály(division)">
    <owl:minCardinality rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
    >90</owl:minCardinality>
    </owl:onProperty>
</owl:Restriction>
```

Az *owl:Cardinality* jobb oldalán egy nemnegatív egész áll. Jelentése, hogy az *owl:Restriction* osztályba azon példányok kerülhetnek, amelyekhez pontosan a megadott számú olyan P tulajdonság tartozik, amelyek jobb oldala szemantikailag különböző.

Az *owl:Cardinality* egy szintaktikus édesítőszer valójában, amely kiváltható az *owl:minCardinality és owl:maxCardinality* együttes használatával.

Metszet (intersection)

A negyedik fajta osztályépítő művelet osztályleírások metszetéből képez egy új osztályt. Erre az *owl:intersectionOf* tulajdonság szolgál, amely egy *owl:Class* osztálybeli példányt köt össze egy osztályleírásokat tartalmazó listával. Az így definiált osztály azon egyedeket tartalmazza, amelyek példányai az összes, listában szereplő osztályleírásnak. Az OWL Lite korlátozza a metszetkonstrukció használatát. A megadott osztályleírások listájában ilyenkor csak megnevezett osztályok és/vagy tulajdonságkorlátozással megadott osztályok szerepelhetnek.

Unió

Az *owl:unionOf* tulajdonság segítségével osztályleírások uniójaként hozhatunk létre új osztályt. Az *owl:unionOf* egy *owl:Class* osztálybeli példányt köt össze egy osztályleírásokat tartalmazó listával. Az így definiált osztály azon egyedeket tartalmazza, amelyek példányai legalább az egyik, a listában szereplő osztályleírás által meghatározott osztálynak.

A következő példában egy olyan osztályt láthatunk, amelynek példányai rendelkeznek tengerparttal, vagy Ázsiában vagy Európában találhatóak:

```
<owl:Class>
   <owl:unionOf rdf:parseType=”Collection”>
         <owl:Class rdf:about=”#Tengerpart”/>
         <owl:Class rdf:about=”#Ázsia”/>
         <owl:Class rdf:about=”#Európa”/>
   </owl:unionOf >
</owl:Class>
```

Az unióképzés nem használható az OWL Lite nyelvben.

Komplemens

Az utolsó fajta OWL osztályleírás egy osztály komplemenseként definiál egy új osztályt. Az *owl:complementOf* tulajdonság egy *owl:Class* osztálybeli példányt köt össze egyetlen osztályleírással. Az *owl:complementOf* egy olyan osztályt ír le, amelynek példányai pontosan azok, amelyek nem példányai a jobb oldalon megadott osztályleírásnak.

Az alábbi példában azon repülők osztályát adjuk meg, amibe nem tartoznak a négymotoros bombázógépek egyedei:

```
<owl:Class>
   <owl:complementOf>
         <owl:Class rdf:about=”# Négymotoros bombázó” />
   </owl:complementOf>
</owl:Class>
```

Az komplemensképzés nem használható az OWL Lite nyelvben.

OWL-axiómák

Az osztályleírások képezik az alapját annak, hogy axiómákat fogalmazhassunk meg. Axiómát háromféle OWL-tulajdonság segítségével képezhetünk, mindhárom tulajdonság értelmezési tartománya és értékkészlete is egy-egy osztályleírás lehet (az eddig látott tulajdonságok egyikének sem volt osztályleírás az értelmezési tartománya). A három tulajdonság a következő:

1. *rdfs:subClassOf*
2. *owl:equivalentClass*
3. *owl:disjointWith*

Az *rdfs:subClassOf* segítségével kijelenthetjük, hogy az alanyként megadott osztályleírás interpretációja részhalmaza a tárgyként megadott osztályleírás értelmezésének[4].

Az *owl:equivalentClass* segítségével kijelenthetjük, hogy az alanyként megadott osztályleírás interpretációja pontosan ugyanaz, mint a tárgyként megadott osztályleírás értelmezése. Azaz megadhatjuk, hogy a két osztályleírás példányhalmaza ugyanaz. Az OWL sokszor ad lehetőséget arra, hogy az *owl:equivalentClass* tulajdonságot elhagyjuk, amennyiben a környezetből kiderül, hogy implicit módon erről van szó.

Az *owl:disjointWith* segítségével pedig kijelenthetjük, hogy az alanyként megadott osztályleírás interpretációjának és a tárgyként megadott osztályleírás interpretációjának nincsen közös eleme.

Valójában persze mind az ekvivalencia, mind a diszjunktság explicit módon történő kifejezése nagyon hasznos, hiszen a legtöbb esetben névtelen osztályleírásokat használunk, amelyek esetén a kétirányú tartalmazás leírása nehézkessé válik. Ha ragaszkodunk ahhoz, hogy alosztály-tulajdonságokkal fejezzük ki az ekvivalenciát, akkor megoldás lehet az, hogy az ilyen osztályoknak is nevet adunk, de ez nagy mértékben ront(hat)ja ontológiánk olvashatóságát és tömörségét.

Az OWL-axiómák fontos eszközei a különböző ontológiák közötti kapcsolatok megteremtésének is. Segítségükkel kijelenthetjük például, hogy bizonyos, a saját

[4] A „részhalmaza” kapcsolatban megengedjük az egyenlőséget, azaz a nem valódi részhalmazokat is

ontológiánkban szereplő osztályok ekvivalensek mások osztályaival, így távoli ontológiákat kapcsolhatunk össze a sajátunkkal.

OWL-tulajdonságok fajtái

OWL-ben megkülönböztetünk *egyedtulajdonságokat* (object properties) és *adattípus-tulajdonságokat* (datatype properties). Az *egyedtulajdonságok* egyedeket kötnek össze egyedekkel. Az adattípus-tulajdonságok egyedeket kötnek össze értékekkel.

Az egyedtulajdonságokat az *owl:ObjectProperty*, az adattípus-tulajdonságokat az *owl:DatatypeProperty* osztály példányaként adjuk meg. OWL DL-ben mindkét osztály az rdf:Property osztály alosztálya, amelyek ráadásul diszjunktak. OWL DL-ben elképzelhető olyan RDF-tulajdonság, amely se nem OWL-egyed-, se nem OWL-adattípus-tulajdonság.

OWL Full esetén az egyedtulajdonságok osztálya megegyezik az összes RDF-tulajdonság osztályával. Az adattípus-tulajdonságok ilyenkor alosztályai az egyedtulajdonságoknak.

Az OWL nyelv változatos lehetőségeket kínál osztályleírások megadására. Tulajdonságok esetén nem ez a helyzet. Nem tudunk tulajdonságokat más tulajdonságokkal definiálni. Az egyetlen lehetőségünk, hogy kijelentsünk egy tulajdonságról, hogy létezik, erre az OWL megnevezett osztályának konstrukciójával analóg szerkezetet használhatunk.

Tekintsünk meg két példát:

```
<owl:ObjectProperty rdf:ID="tábornok"/>
<owl:ObjectProperty rdf:ID="életkora"/>
```

A tábornok egyedtulajdonság embereket köt össze emberekkel, az életkora adattípus-tulajdonság pedig embereket számokkal.

Tulajdonságaxiómából azonban már sokfélét találhatunk az OWL nyelvben. Ezeket négy csoportba osztjuk az alábbiaknak megfelelően:

1. RDF séma konstrukciók
2. ekvivalencia és inverz
3. globális számosság-megkötések

4. tulajdonság-karakterisztikák

Az első csoportba, az RDF séma által nyújtott konstrukciók közé tartozik a tulajdonság-altulajdonság viszony kifejezésére szolgáló *rdfs:subPropertyOf* szerkezet, valamint az értelmezési tartományra és értékkészletre vonatkozó korlátozások. A második csoportba tartozó konstrukciók segítségével egy tulajdonságról kijelenthetjük, hogy ekvivalens egy másikkal, illetve hogy az inverze egy másik tulajdonságnak. A globális számosság-megkötések a tulajdonságra nézve globálisan megkötik, hogy jobb, illetve bal oldalukon állhat-e egynél több egyed, illetve érték. Végül a tulajdonság-karakterisztikák segítségével kijelenthetjük egy tulajdonságról azt, hogy tranzitív és/vagy szimmetrikus.

Egy tulajdonság példányai alatt – az RDF sémához hasonlóan – párokat értünk.

Tulajdonságkarakterisztikák

Egy relációnak számos logikai jellemzője lehet, amelyek közül az OWL a szimmetria és a tranzitivitás leírását támogatja.

Szimmetria: a szimmetrikus tulajdonságokra az igaz, hogy két erőforrás között mindkét irányban fennállnak. Másképpen, amennyiben az (a,b) pár példánya a tulajdonságnak, akkor biztosan a (b,a) pár is. Példa szimmetrikus tulajdonságra a *testvére* tulajdonság, hiszen ha tudjuk Gézáról, hogy Zsófi testvére, akkor Zsófiról is tudjuk, hogy ő bizony Géza testvére.

A szimmetrikus tulajdonságok az *owl:SymmetricProperty* osztály elemei, amely osztály az *owl:ObjectProperty* osztály alosztálya. Ennek megfelelően az OWL DL és az OWL Lite nyelvek esetén csak egyedtulajdonságokról jelenthetjük ki, hogy szimmetrikusak.

Nézzünk példát szimmetrikus tulajdonság leírására OWL-ben:

```
<owl:SymmetricProperty rdf:ID=”testvére”>
   <rdfs:domain rdf:resource=”#Ember”/>
   <rdfs:range rdf:resource=”#Ember”/>
</owl:SymmetricProperty>
```

A szimmetria valójában úgy is felfogható, hogy egy tulajdonságról kijelenthetjük, hogy az ekvivalens a saját inverzével.

Tranzitivitás: egy tulajdonságról azt mondjuk, hogy tranzitív, ha igaz rá, hogy amennyiben mind az (a,b), mind a (b,c) pár példánya, akkor az (a,c) pár is példánya. Jó példa tranzitív tulajdonságra az őse viszony, mert valaki ősének az őse az illető őse. Hasonlók mondhatók el általában a rész-egész viszonyt kifejező tulajdonságokról is. Például amennyiben tudjuk, hogy a küllő része a keréknek, valamint a kerék része a biciklinek, akkor az is igaz, hogy a küllő része a biciklinek.

A tranzitív tulajdonságokat az *owl:TransitiveProperty* osztály segítségével adhatjuk meg. Ezen osztály az *owl:Property* osztály alosztálya, így az OWL DL és az OWL Lite nyelvek esetén csak az egyedtulajdonságok lehetnek tranzitívak. Az ok ugyanaz, mint az előzőekben: érték nem lehet alanyi pozícióban.

Tekintsük meg az alábbi példát:

```
<owl:TransitiveProperty rdf:ID=”része”>
   <rdfs:domain rdf:resource=”#Dolog”/>
   <rdfs:range rdf:resource=”#Dolog”/>
</owl:TransitiveProperty>
```

Az OWL DL és az OWL Lite nyelvek esetén tranzítív tulajdonságokra, azok szülőtulajdonságaira vagy ezek inverzeire semmiféle számosságkorlátozást nem adhatunk meg.

Példányok

Példányok leírására az OWL nem vezet be semmilyen, az RDF-től eltérő jelölést vagy koncepciót. Az OWL nem él az egyedinév-feltételezéssel, azaz több URI is azonosíthatja ugyanazt az erőforrást. Ennek megfelelően fontos annak a lehetőségnek a biztosítása, hogy bizonyos URI-król kijelenthessük, hogy azok valójában ugyanazt az erőforrást reprezentálják, vagy éppen ellenkezőleg, különbözőt. Egy OWL-következtetőnek mindkét lehetőséggel számolnia kell mindaddig, míg konkrétan nem derül ki két URI viszonya.

Az OWL Full nyelvben minden erőforrás példány (az *owl:Thing* osztály ekvivalens az *rdfs:Resource* osztállyal), ezért az alábbi konstrukciókat arra is használhatjuk, hogy például osztályok közötti azonosságot írjunk le. Ez lényegesen különbözik az osztályok közti

ekvivalencia leírásától, mert itt ténylegesen az osztályok által reprezentált fogalmakat feleltetjük meg egymásnak.

Az *owl:sameAs* tulajdonság segítségével kijelenthetjük két URI-val azonosított példányról, hogy azok valójában ugyanazok. Az owl:sameAs konstrukciót nagyon gyakran arra használjuk, hogy segítségével azonosítsuk a különböző ontológiákban definiált, de azonosnak tekintendő fogalmakat és tulajdonságokat. Ehhez azonban már OWL Full nyelvet kell használnunk, hiszen az osztályokat és tulajdonságokat példánynak kell tekintenünk.

Példányok közötti különbséget kétféleképpen adhatunk meg:

- *owl:differentFrom*: két példány különbözőségének megadására szolgál
- owl:AllDifferent: egy példányokat tartalmazó kollekció elemeiről jelenthetjük ki, hogy páronként különbözőek

OWL-adattípusok

Az OWL nyelvben adattípusokat az RDF-ben megszokott módon használhatunk. A konkrét adattípusok (a típusos literálok) az *rdfs:Lit*eral osztály alosztályai. Egy RDF-állítás tárgyának típusát az *rdf:datatype* attribútum segítségével adhatjuk meg, amelynek értéke az adattípust azonosító URI.

Az alábbiakban kijelenthetjük, hogy az #bakancsom (amely a Cipő osztály példánya) egyed mérete 45:

```
<Cipő rdf:ID=”bakancsom”>
   <mérete rdf:datatype=”&xsd;int”>45</mérete>
</Cipő>
```

Az adattípust azonosító URI a szabvány ajánlása szerint az XML sémát azonosítja, de ez nem kötelező jellegű.

Az OWL evolúciója

- *OWL 1.0 [3]:*
  - A legelső OWL szabvány, 2004. február 10-én vált hivatalosan is ajánlássá a W3C által. Magáról a szabványról az előző fejezetben olvashatunk részletesebben. Ez a fejezet az 1.0-tól történő fontosabb eltéréseket és változtatásokat magába foglaló szabványokkal foglalkozik.
- *OWL 1.1 [4]:*
  - Ez a szabvány 2006. december 19-én vált hivatalosan W3C munkatervvé. Az *OWL 1.1* kiterjeszti az 1.0-ás szabvány által definiált OWL nyelvet. Az 1.1-es verzió absztrakt szintaxisa hasonlít az 1.0-ás ajánlás absztrakt szintaxisához, azonban visszafelé nem kompatibilis. Ennek az az oka, hogy az 1.0-ás szabvány absztrakt szintaxisából eredendő számos problémát nem lehet megoldani a visszafelé kompatibilitás feláldozása nélkül.

    Az 1.1-es szabvány szerint készült ontológiákat és elemeiket IRI-vel (International Resource Identifiers) azonosítjuk, ami tulajdonképpen egy speciális URI. Az *OWL 1.0*-ás ajánlás URI-kat használt az egyes elemek azonosításához, ahhoz hogy megőrizzük a kompatibilitásunkat az 1.0-val, URI-t használunk, ami egy IRI hivatkozás tulajdonképpen, legyen az éppen teljes vagy rövidített IRI.

    Az 1.1-es verzió névtér prefixe a következőképpen fog kinézni: http://www.w3.org/2006/12/owl11#.

    Az 1.1-es ajánlás szerint készült ontológiák rendelkeznek egy annotációs halmazzal, ahol a halmaz elemeinek az értéke két fajta lehet:

    - konstans (nemcsak stringek)
    - az ontológiák entitásai: ami azt jelenti, hogy az ilyen annotációk egyértelművé teszik azt, hogy nemcsak egyszerű konstansok szerepelhetnek értékként, hanem egy entitás az adott ontológiából vagy éppen néhány más ontológiából.

    OWL 1.0-ás szabványban az *owl:imports* egy speciális annotáció volt, ami lehetővé tette, hogy egy ontológiát beimportáljon egy másik ontológiába. Ezzel szemben az 1.1-es verzióban az importálások nem annotációk, nincs beépített *owl:imports* annotációs tulajdonság. Minden ontológia tartalmaz egy feltehetően üres import deklarációs halmazt.

- Tekintsünk át néhány fontosabb újdonságot az 1.1-es szabványból:
  - Új leíró logikai konstrukciók: *SROIQ*, mely növeli a leíró logika kifejező erejét. Néhány új lehetőség:
    - *minősített számosság korlátok* (qualified cardinality restrictions)
    - *reflexív, szimmetrikus, antiszimmetrikus, diszjunkt tulajdonságok* (csak egyszerű tulajdonságokra használhatóak)
  - Az adattípusok kifejezőképességének kiterjesztése: ez azt jelenti, hogy a felhasználó saját adattípust hozhat létre, hasonló módon, mint ahogy ezt Protege-ben is teheti. Ez lehetőséget ad arra, hogy új adattípusokat lehet definiálni a *SubClassOf* konstrukcióval, nézzünk rá egy példát:
    SubClassOf (Adult DataSomeValuesFrom(age DatatypeRestriction (xsd: integer 1 minInclusive "18"^^xsd:integer)).
  - Meta modell kiterjesztése és annotációk: *OWL 1.1*-ben egy név használható bármely egyed vagy osztály, vagy tulajdonság esetén. Ehhez azonban biztosítani kell azt, hogy egy egyed nevének a használata ne legyen hatással egy osztály nevének a jelentésére, ezt a módszert „punning”-nak hívják.

- *OWL 2.0 [5]*:
  - A W3C OWL munkacsoportja publikálta 2008. október 10-én. Az OWL munkacsoport tagjai arra törekedtek, hogy visszafelé kompatibilis legyen az 1.0-val, ha át akarunk konvertálni egy OWL 1.0 ontológiát OWL 2.0 ontológiává, akkor annyit kell tennünk, hogy az 1.0-ban készült dokumentumot át kell helyezni 2.0-ba, a konverzió automatikusan megtörténik.

  - Vizsgáljuk meg a 2.0-ás ajánlás újdonságait:
    - Ugyanazzal az URI-val egynél több tulajdonság-típus azonosítása már nem megengedetté vált. Az URI szerepét egy ontológiában az ontológiában elérhető deklarációk határozzák meg.

- A funkcionális szintaxisban a terminális[5] szimbólumok egyszerűen el lettek távolítva, helyette egyszerűbb terminális szimbólumokat használ, például: *ObjectSomeValuesFrom* és *DataSomeValuesFrom* megszorítások helyett *SomeValuesFrom* terminális szimbólumot használ.

- Az OWL nyelv ki lett terjesztve kulcsaxiómákkal, továbbá „Felső" és „Alsó" adattulajdonságokkal.
- Kibővült a szabvány által támogatott adattípusok halmaza.

- Annotációk annotációi megengedettek lettek a nyelv számára, az annotációk értékeinek típusai:
  - Konstansok, nemcsak sztringek, bármilyen OWL 2.0-ás konstans megengedett
  - Entitások
  - Névtelen egyedek
- Névtelen példányokkal bővült a nyelv
- Az ontológiák verzió-kezelése módosult: ha egy ontológiának több verziója van, akkor létezik egy közös IRI, amivel minden verzió elérhető.
- Az importálások kezelése egyszerűsítve lett
- Számos eddigi hibát és mulasztást javít

[5] Terminális szimbólumok: az OWL nyelv elemei

Webontológiák lehetséges alkalmazásai

Elsőként jogosan tehetjük fel a kérdést, hogy miért használjunk és készítsünk webontológiákat? A válasz egyszerű: sokkal hatékonyabb webes alkalmazásokat fejleszthetünk általuk. Az ontológiák nagy jelentőséggel bírnak a jelenleg is kialakulóban lévő intelligens világháló szempontjából.

A webontológiák nagyon jó eszközt nyújtanak ahhoz, hogy egy terület (tudományos vagy egyéb), egy szakmai ágazat ismereteit és fogalmait összegyűjtsük, rendszerezzük, egy megfelelő hierarchiát állítsunk fel. Ilyen területek lehetnek például az „intelligens adatbázisok", elektronikus kereskedelem, kartográfia, stb.

Az ontológiák nem csak feljavítják meglévő alkalmazásainkat, hanem újakat is teremthetnek. Ha valaki a nászútját szeretné megtervezni, katalógusokat, menetrendeket, szállodák listáit, a lehetséges országok éghajlatának megfelelő időjárást, a látnivalókkal kapcsolatos információt fog keresni a weben. Ezután útitervet fog készíteni, hogy milyen városba fog megérkezni, hogyan jut el oda, kiválasztja, hogy melyik apartmanban vagy szállodában szeretne lakni, és elintézi az összes szükséges foglalást. Egy szemantikus web alkalmazás képes lesz arra, hogy összerakja az egyes szempontoknak megfelelő adatokat, és egyetlen eredménnyé alakítsa át őket.

Az ilyen típusú alkalmazások elkészítéséhez nagyban hozzájárul az RDF, DAML+OIL és az OWL tovább fejlesztése.

Tekintsünk meg néhány példát az új alkalmazásokra:

- Webportálok: tipikusan olyan konkrét webhelyek, ahol nagyobb mennyiségű információt osztanak meg a felhasználókkal egy adott témakörben, mint például egy második világháborús portál (http://www.masodikvh.hu/), ahol részletesebben foglalkoznak az akkori haditechnika és hadszínterek bemutatásával. A webhelyek tartalmát általában a portált rendszeresen látogató felhasználók bővítik. Témák szerint megjelölik a portál tartalmát, ez a folyamat metatag-ek segítségével történik.

Az egyes témák puszta indexelése nem elegendő ahhoz, hogy minden információhoz hozzájussanak a közösség tagjai, amire szükségük van. Lehetőség nyílik a webportálok számára, hogy ontológiát definiáljanak. Az ontológia segítségével egységes terminológiát teremthetünk, és új fogalmakat hozhatunk létre, mint a „szerző", „személy", „fejezet". Az újonnan definiált fogalmak segítségével kijelenthetjük, hogy minden szerző személy és minden fejezetnek van szerzője. Így ezek a következtetések segítik a felhasználókat, hogy olyan találatokat kapjanak kereséseikre, amelyeket a hagyományos keresők segítségével nem kaptak volna meg, vagy a nagyon sok találat közül nem találták volna meg a megfelelőt.

Az ontológiákra építő webhelyekre konkrét példaként az Agent Portált érdemes megemlíteni, melyet az első ontológiákra épülő e-üzleti portálnak neveznek, de jó példa még az OntoWeb is, mely egy ontológiakutató weboldal.

- Mobil-kooperatív számítástechnika (Ubiquitous Computing) [2]: a számítástechnika kialakulófélben lévő új irányzata. Az új irányzat jellemzője, hogy kisméretű, hordozható, vezeték nélküli eszközöket használnak nagyobb mennyiségben. A széles körben elterjedt technológia azonban speciális hálózati architektúrát igényel, hogy zavartalan és biztonságos kommunikáció jöjjön létre az egyes eszközök között, és a hálózatot automatikusan be tudja konfigurálni (hasonlóan a Wifi technológiához).

  Az ilyen „ad hoc" hálózatok technológiájának alappillére a szolgáltatások automatikus felismerése, melyeket a különböző készülékek nyújtanak (mobiltelefonok például). Ez azt jelenti, hogy ezeknek a szolgáltatásoknak a jellemzőit össze kell gyűjteni, szabványosítani kell (ilyen szabvány például a Microsoft UPnP és a SUN JINI rendszere). Így a szolgáltatásokat leíró rendszerek függnek a szabványoktól.

A mobil-kooperatív számítástechnika nagy problémája, hogy a különböző gyártmányú készülékeknek kommunikálniuk kell egymással. Olyan szolgáltatást kell nyújtani, ami biztonságos kommunikációt tesz lehetővé. Egy megfelelő ontológianyelv használatával képesek leszünk leírni a készülékek jellemzőit, a hozzájuk történő csatlakozás módját, technikai követelményeket és a korlátozásokat. Ezek a jellemzők meghatározzák az egyes készülékek számára, hogyan jelenjenek meg a mobi-kooperatív hálózatai konfigurációban.

Saját webontológia bemutatása

„A Második Világháború Nagy Csatái”

(Great Battles of Second World War)

A webontológiám témája a második világháború nagy csatáinak, hadjáratainak ismertetése **hadtörténeti szempontból**, politikai ideológiáktól mentesen. Egyfajta emléket állítva azoknak, akik életüket vesztették vagy éppen túléltek a világháborút. Ez egyben azt is jelenti, hogy igyekeztem néhány visszaemlékezéssel (naplókból, emlékiratokból) fűszerezni az ontológiámat. Úgy gondolom ennek a történelmi eseménynek a következtében alaposan megváltozott világunk, ezért érdemes foglalkozni vele, hogy levonhassuk a megfelelő konklúziót a jelenükre és a jövőnkre vonatkozóan.

Mára nagyon sok információ, statisztikai adat (számadat) áll a rendelkezésünkre a legkülönbözőbb forrásokból a témát illetően, ezek közül – ahogy korábban is említettem – hadtörténeti tematika szerint építettem fel az ontológiámat. A nagymennyiségű ismeret és adat között nehéz eligazodni sokszor. Ekkor jött az ötlet, hogy az ismereteket jó lenni rendszerezni és átláthatóvá tenni, egy kézzel fogható struktúrát adni nekik. Ha elolvasunk egy cikket vagy tanulmányt ebben a témában, akkor tipikusan a következőkkel találkozhatunk:

- „IV. légiflotta alá rendelt XI. légihadtest – tulajdonképpen légideszant erő – és a VIII. légihadtest” – 1941. májusa, Merkúr hadművelet: Kréta ellen indított német támadás [Forrás: http://en.wikipedia.org/wiki/Battle_of_Crete]

- „A támadásban 190 hadosztály, ebből 27 páncélos, illetve gépesített vett részt. A Barbarossa hadművelet három irányba írta elő az előrenyomulást.” – 1941. júniusa, Szovjetunió megtámadása [Forrás: http://hu.wikipedia.org/wiki/Második_világháború]

- „A franciaországi hadjárat (Fall Gelb)” – 1940. májusa [Forrás: Jónás Edit, Rónaszegi Éva - A Második Világháború Krónikája]

- „A novokramatorszki nehézipari gépgyár például 1941. szeptember 29-én kapta meg a parancsot, hogy szerelje szét gyártósorait. Öt napon belül a német bombázók támadásai közepette valamennyi gépét vagonokra pakolták, az utolsó napon pedig az üzem 2500 munkása 30km-t menetelt a legközelebbi működő vasúti pályához, hogy a gépeket követve vonatra szállhasson. Ugyanakkor az üzemeket sikerült szinte hihetetlen sebességgel újraindítani. December 8-án, tíz héttel az után, hogy a harkovi tankgyárat az Urál-hegységbeli Cseljabinszkba telepítették, már le is gyártotta az első 25 darab T-34-es harckocsit." [Forrás: http://hu.wikipedia.org/wiki/Második_világháború]

- „A közismerten kemény orosz télben jobban vizsgáztak a szovjetek fegyverei is: miközben a német Schmeisser géppisztolyok hamar befagytak, a lényegesen egyszerűbb szerkezetű orosz Spagin (PPS) géppisztolyok mindenféle időjárásban használhatók maradtak, és sem a nedvesség, sem a homok vagy más szennyeződés nem ártott nekik. A dieselmotoros T-34-es közepes tankok is felülmúlták a német Pz-III-as és Pz-IV-es tankokat" – 1941. ősze [Forrás: Földi Pál – A „Barbarossa" hadművelet]

- „A szovjet hadsereg hosszas előkészítés után 1943. január 12-én indította meg az Osztrogrozsg-Roszos hadműveletet. A hadművelet a sztorozsevoi (urivi) hídfőből indította el a 40.-ik szovjet hadsereg. A túlerő mértéke: zászlóalj 2,7:1 tüzérség 5:1 páncélos: 1,3:1." [Forrás: http://hu.wikipedia.org/wiki/2._magyar_hadsereg]

- „1943. január 12-én, −30°C fokos hidegben, (a hadtest naplója -42 fokot rögzít) erős harckocsi-támogatással megindult a szovjet támadás az arcvonal északi részén (az *urivi hídfő*ből kitörve déli irányban)."

  [Forrás: http://hu.wikipedia.org/wiki/2._magyar_hadsereg]

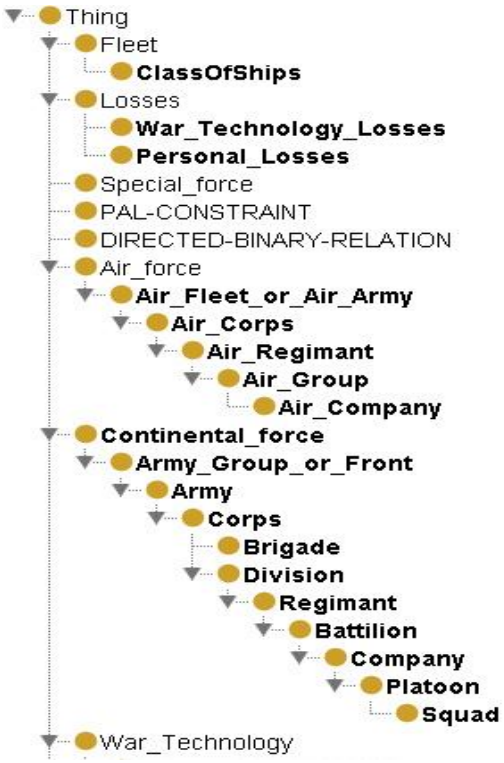
- „Még Chamberlain is Churchill mellé állt, aki a Parlamentben kijelentette: *Harcolunk a tengerpartokon, a kikötőkben, a mezőkön és a városok utcáin és harcolunk a dombok között és a hegyekben. Soha sem adjuk meg magunkat, és ha, amit egy percig sem hiszünk, a sziget, vagy annak nagy része le lenne igázva,*

akkor birodalmunk tengeren túli részein a brit hajóhad segítségével folytatják a harcot." – 1940. nyara [Forrás: Joachim Peiper – Banditák nyolc óránál]

Az első példában rögtön találkozhatunk a légiflotta, légihadtest, stb. fogalmakkal, melyek első olvasatra nem mondanak igazából semmit, hiszen nem tudjuk pontosan mit jelentenek, mekkora erőt vagy mennyiséget képviselnek.

Az ontológia felépítése

Az olyan fogalmakat, mint a hadtest, hadosztályt, stb. hierarchikusan igyekeztem felépíteni, pl.:



1. Részlet a webontológiából

A szárazföldi erőkön belül a hadseregcsoport vagy front áll a legmagasabb szinten, ezt követi a hadsereg, majd a hadtest és így tovább. Ezáltal már láthatóvá válnak a szintek, hogyan is épül fel egy szárazföldi haderő a rajtól az ezredeken át a hadseregcsoportig. A megnevezett egységjelöléseket minden harcoló ország hadereje használta. A szárazföldi

erőkkel egyetemben több fő kategóriát is felállítottam:

- Légierő: azok az országok, melyek rendelkeztek harcigépekkel, számuktól és feladatuktól függően különböző nagyságú egységekbe tagolták, a repülő századoktól a légihadtesteken át egészen a légiflottákig, például a német légierőnél 1942. nyarán a 6. német hadsereget támogató 4. légiflotta 1200 repülőgéppel rendelkezett a „Kék" hadművelet során. A légierő egységeinek harcerejét és nagyságát első sorban a bevehethető gépek száma képviselte, nem a személyi állomány létszámának nagysága. Egyéb tényezőként természetesen a harci tapasztalatot és a gépek típusait is figyelembe kell venni.

- Szárazföldi erők: a háborúban részt vett országok mindegyike rendelkezett valamilyen szárazföldi erővel. Egy adott ország lakosságától függően tudta felállítani saját hadseregét, amit azután megfelelő haditechnikai eszközökkel kellett felszerelnie, a nagyobb, erősebb gazdasággal rendelkező országoknak ez nem jelentett akkora problémát, mint a kisebb, iparilag fejletlenebb országoknak. Vegyük Magyarországot például, a 2. magyar hadsereg állománytáblája szerint 207000 fővel (a munkaszolgálatosokat és a kisegítő, támogató fegyveres erők létszámát is beleszámítva) rendelkezett 1942. július 3-án, azonban a felszerelés elég hiányos volt, és kis számban fordult elő gépesített egység, vontató eszköz. Az utánpótlás akadozott, így kisebb harcerőt képviselt egy ugyan kisebb létszámú ám jobban felszerelt hadsereggel szemben. A II. magyar hadsereg abban az időben három gyaloghadtesttel, vagyis 9 hadosztállyal, összesen 27 ezreddel (ezek könnyű hadosztályok[6] voltak) vonult ki a szovjet frontra.

- Különleges erők: bizonyos országok felállítottak és kiképeztek úgy nevezett „különleges erőket", melyek nagyobb harcértékkel rendelkeztek, mint az „átlagos", reguláris hadsereg csapatai. Ilyenek voltak a német haderőnél bevetett SS-alakulatok, ejtőernyősök (fallschirmjager-ek), vagy az amerikai ranger[7]-ek,

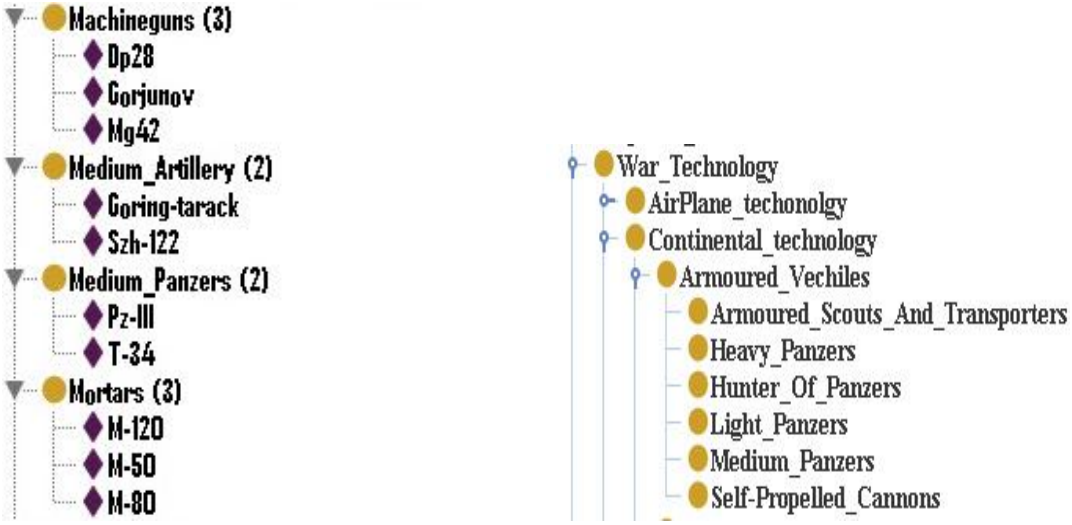
[6] Könnyű hadosztály: kevesebb emberrel rendelkezett, mint egy normál teljesen feltöltött hadosztály, ez a meghatározás csak a 2. magyar hadsereg alakulataira vonatkozik

[7] Ranger: különleges amerikai katonai egység

angol kommandók, francia idegenlégiósok, stb. Ezek az egységek különlegesen nehéz, a hagyományos egységek számára lehetetlennek tűnő feladatokat hajtottak végre, például: egy német ejtőernyős egység elfoglalta a „bevehetetlennek” tartott Eben-Emael belga erődöt 1940. május 10-én, 1944. június 6-án a Normandiai partraszállás során az Omaha partszakasznál a 2. ranger zászlóaljnak sikerült véres veszteségek árán áttörni a német védelmi rendszeren. Ezek az alakulatok sokszor igen nagy árat fizettek feladataik végrehajtása során, például német ejtőernyős hadosztályok Kréta 1941-es elfoglalásakor 14000 főből körülbelül 4500 főt vesztettek halottakban, sebesültekben, foglyokban és eltűntekben.

- Szabadcsapatok: a háború során sok országban a civil lakosság is csatlakozott a harcokhoz, harcoló alakulatokhoz, partizánharcokhoz, felkelésekhez. Az ilyen csapatok nem reguláris katonai egységek voltak, nem részesültek kiképzésben, nem hivatalos formában szerveződtek. Férfiak, nők, gyermekek, öregek és fiatalok egyaránt részt vettek fegyveres harcokban. Különösképp olyan országokban, mint a volt Szovjetunió lángolt fel a partizánharc, mivel Németország elég kegyetlenül és durván bánt a helyi lakossággal, sok atrocitás történt, sok civilt deportáltak, kényszermunkára vittek. Ezért már a Barbarossa hadművelet kezdetekor sokan csatlakoztak a partizánalakulatokhoz. A „szabadcsapatok” kategóriájába tartoznak a milíciák is, például Sztálingrádban (ma Volgográd) 1942-ben megalakult egy 75000 fős milícia a város védelmére.

A következő példákban olyan típusjelöléseket láthatunk, mint T-34, PPS, Pz-III - egy-egy haditechnikai eszközt képviselnek -, és amelyek igazából elsőre semmitmondóak, de megfelelő helyre beillesztve őket az ontológiában már informatívak lesznek. Leírást készíthetünk hozzájuk akár egy konkrét webes erőforrást megadva, akár konkrét leírást adhatunk meg annotációként, társíthatunk hozzájuk képeket. A haditechnikai eszközöket is igyekeztem egy helyre csoportosítani:



2. A haditechnikák egy része

Az ábrán látható, hogy például a T-34 egy közepes harckocsi jelölésére szolgál, ami a szárazföldi haditechnikai eszközök, páncélozott járművek kategóriájában, a közepesen páncélozott harckocsik között található.

A fentebb olvasott típusokat főbb kategóriákba soroltam:

- Repülőgép technológiák: a különböző típusú, egyéni feladatkörrel rendelkező repülőgép-kategóriákat foglalja magában, mint például:
    - La-5: rendkívül megbízható szovjet vadászgép.
    - FW-200C3-U4 („Kondor”): strapabíró szállítógép, 1942 telén a német légierő a Sztálingrádnál bekerített 6. német hadsereget ilyen gépekkel próbálta ellátni utánpótlással.
    - He-219 („Uhu”): az angol bombázópilóták által rettegett éjszakai elfogóvadászgép, az angolok szerencséjére csak 289 gép készült el a háború alatt.
    - Ju-87D-1 („Stuka”): a német villámháború jelképévé vált taktikai zuhanóbombázó, a gép közismert hangját a kimerevített futómű tetté igazán félelmetessé.
    - B17-Flying Fortress: a legeredményesebb amerikai, négymotoros stratégiai bombázó, minden fronton harcolt, az erőd jelzést azért kapta, mert úgy tervezték, hogy vadászkíséret nélkül is meg tudja védeni magát.

- Szárazföldi technológiák: az ontológia egyik legnépesebb csoportja, a harcoló országok szárazföldi haditechnikai eszközeinek széles spektrumát tartalmazza, mint például:
    - Tigris-E6: a leghíresebb német nehéz harckocsi a Második Világháború során, a szövetséges csapatok úgy tartották, hogy öt Sherman tank kell egyetlen Tigris tank ellen, hogy esély legyen a harckocsi elpusztítására.
    - Kar98-K: alapvető gyalogsági fegyver volt a német hadseregnél, pontos, megbízható puska, csupán a tűzgyorsasága volt alacsony.
    - Flak-88: légelhárító ütegként kezdték rendszeresíteni a német haderőknél, azonban 1940 májusában 48 brit harckocsit lőttek ki a francia Arras városánál 88-as ágyúkkal, innentől kezdve kiváló páncéltörőfegyverként is szolgáltak.
    - 41M-Király: magyar fejlesztésű és gyártmányú géppisztoly volt, sok tekintetben felülmúlta a német MP-40 és az orosz PPS géppisztolyokat, a magyar katonák kedvelt harceszköze volt.

- Haditengerészeti technológiák: a háborúban résztvevő országok flottáinál szolgált eszközök típusainak gyűjteménye, mint például:
    - HMS Hood: brit csatahajó, a flotta büszkesége volt, míg 1941 májusában el nem süllyesztette a német Bismark csatahajó.
    - U-96: az egyik legeredményesebb német tengeralattjáró, melynek történetét meg is filmesítették (Das Boat).

- Különleges technológiák: a háborúban bevetett speciális haditechnikai eszközöket foglalja össze, mint például:
    - V-1: az első számú német megtorlófegyver, mely egy robotrepülőgép volt tulajdonképpen, kifejlesztése során három berepülőpilóta is életét vesztette, maga a fegyver számos brit és belga állampolgár életét követelte.
    - MK-I Radarrendszer: 1940-ben állították fel a britek Anglia légterének védelmére, mely nagyon hasznos rendszernek bizonyult és nagyban segítette az angol vadászgépkötelékeket a német bombázóegységek ellen.

Egy hadművelet felépítése

Az eddig felsoroltak csak kisebb részei voltak a teljes ontológiának. Nézzük meg, hogyan épül fel egy teljes hadművelet!

Az egyes hadműveletekhez szükséges részleteket, információkat, dátumokat, stb. fontos összegyűjteni és egy egységbe foglalni. Fentebb olvashattuk a „Barbarossa" hadműveletet az egyik idézetben. Jogos a kérdés, hogy miről is szól ez a „Barbarossa" haditerv tulajdonképpen.

A Wermacht[8] feladata volt a terv megvalósítása, mely az akkori Szovjetunió lerohanását tűzte ki célul. 1941. június 22-én hajnali 3 óra 15 perckor, 153 német és 29 szövetséges hadosztály négymillió katonájával, 3500 harckocsival és 4000 repülőgéppel, 3500 km-es arcvonalon (vagyis a közös határ teljes szélességében) indult a Barbarossa-hadművelet a Szovjetunió ellen, a megnemtámadási szerződést megszegve. A hadüzenetet reggel 4-kor adta át Ribbentrop a szovjet nagykövetnek, a német követ pedig Molotovnak. Napóleon is ezen a napon támadta meg Oroszországot 1812-ben.

A hadjárat világnézeti harcot jelentett a bolsevizmus ellen, célja az "élettér", a mezőgazdasági terület, nyersanyag, rabszolga-munkaerő megszerzése, hangoztatta a hitlerista propaganda.

A németekkel román, olasz, szlovák, finn és magyar csapatok is harcba szálltak. Sztálin a brit figyelmeztetések, a kémjelentések s a német légi felderítés határsértései ellenére nem hitt a támadásban, nem engedte hadseregének a határokon történő mozgósítását.

A Vörös Hadsereg nem mozdult, nehogy provokációra gondoljanak a németek, így légiereje nagyrészt a földön semmisült meg (több ezer harcirepülő), a légiuralmat kivívó német hadsereg meglepetésszerű támadása nyomán, technikai fölényük révén gyorsan nyomultak előre. Sztálin, aki 1938-as koncepciós pereivel megtizedelte tábornoki karát (száz vezénylő tábornokából körülbelül hetvenet kivégeztetett, idegileg összeomlott, napokig nem szólt a nyilvánossághoz.

Július 9-én Minszk környékén 328 ezer szovjet katona került katlanba, majd esett fogságba, augusztus 5-én Szmolenszknél újabb 310 ezer. A szeptember 8-án körülzárt Leningrádot 900 napig ostromolták - eredménytelenül. Hitler a nyugati villámháborút akarta megismételni, ám csapatait (főleg a páncélos erőket) a két szárnyon lévő hadseregcsoportok

[8] Wermacht: Németország fegyveres védereje

között osztotta szét. A német Közép Hadseregcsoport október 2-án Moszkva ellen indult, ennek eredményeképpen a brjanszki és vjazmai csatákban 673 ezer hadifoglyot ejtettek, ám későn indultak az orosz főváros ellen, így az időjárás fokozatosan rosszabbra fordult. Az oroszok, kihasználva a nagy terület előnyeit, lassan magukhoz tértek, „Tél" tábornok csapásai folyamatosan csökkentették a támadó német ékek erejét. Centralizálták a katonai vezetést, így Sztálin az eddiginél is nagyobb hatalomra tett szert.

A német légierő által fenyegetett területeket kiürítették, az Urálon túlra telepítették a hadiipart. Nagy Honvédő Háborút hirdettek, az feladott szovjet területeken a „felperzselt föld taktikáját" alkalmazták, sok helyen partizánalakulatokat szerveztek. A kimerült német hadosztályok ugyan elérték Moszka külvárosát, még be is hatoltak a városba, de visszaverték őket. Ekkorra végképp elfogyott a lendület a német támadásból, a szovjet légierő kivívta a légifölényt Moszka körzetében, és a szárazföldi erőket pedig friss szibériai hadosztályokkal töltötték fel. December 5-én nagy erejű ellentámadást indítottak és több száz kilométerrel hátrébb vetették vissza a német csapatokat. Így végképp szertefoszlott a villámháborús terv, a Barbarossa kudarcba fulladt.

3. Barbarossa hadműveleti terv, 1941. 06. 22.

Rögtön láthatjuk, hogy egy ilyen hadműveletnek mindig van egy fedőneve, amivel azonosítani lehet. Mindig van célja, célpontjai, amik fényében máris realizálható a hadművelet „miértje”. Olvashattuk, hogy hatalmas hadseregcsoportokkal indítottak offenzívát, melyeknek saját célpontjaik volt. A térképen ezek remekül látszanak, mely seregtest milyen irányban támadott.

Egy hadművelet esetén ismernünk kell a fontosabb eseményeket, dátumokat, helyszíneket, hiszen ezek mentén láthatjuk az életútját. Mikor is a haditervből konkrét hadművelet lesz. A haditervben megadják a szükséges alapadatokat, mint célok, rendelkezésre álló fegyveres erők, haditechnikai eszközök, utánpótlás, várható időjárási körülmények, támadási időpontok, menetterv. A nagy kérdés az, hogyan valósult meg a haditerv a valóságban. A rendelkezésünkre álló történelmi források alapján össze kell gyűjteni a fontosabb eseményeket, melyeket kiemeltem és felhasználtam például a „Fall Barbarossa” kidolgozása során, ilyenek:

- „1941. szeptember 26. Kijev eleste”
- „1941. november 16. Krím-félsziget elfoglalása”

Így lépésről-lépésre végig kísérhetjük a hadjárat eseményeit, menetét. Az alábbi példán a Barbarossa hadműveletet reprezentáló fontosabb események listáját láthatjuk:

4. A Barbarossa hadművelet főbb eseményei

Katonák és visszaemlékezések

Fontos a csatákban részt vett személyek szerepeltetése, akik részben a parancsnokok közül kerülnek ki, részben pedig elismert közkatonák, egyéb tisztek és altisztek, civilek megjelenítése. Az emberek szeretnek hősökre vagy nagy vezetőkre emlékezni és így fennmarad jó pár név a történelem vérzivataros századaiból. Minden személynek külön története van.

A vezérkari tábornokok dolgozták ki különös gonddal a haditerveket, az ő döntéseiken ezrek vagy százezrek élete, ágyúk, tankok, repülők, hajók és járművek ezreinek sorsa múlott. Előfordult, hogy egy-egy hadjáratot egyetlen névvel kötöttek össze, például: Rommel – Tobruk elfoglalása, stb.

A személyeket két fő szempont alapján kategorizáltam:

- Rangok szerint: a tiszthelyettesi tizedestől a főtiszti tábornok és marsallig bezárólag gyűjtöttem össze a rangokat.
- Rangtól független kiemelkedő teljesítményű katonák: az utókor számára sok visszaemlékezésben és régi jelentésekben maradtak fenn olyan emberek nevei, akik különösen nehéz és veszélyes feladatot teljesítettek, kockáztatták saját életüket bajtársaik megmentése közben, esetleg kiemelkedő harci tevékenységeket folytattak. Néhány példát hagy említsek:
    - V.G.Zajcev törzsőrmester: a 284. szibériai lövészhadosztály mesterlövészeként 242 igazolt kilövést ért el a sztálingrádi harcokban, az orosz katona jelképévé vált, megkapta a Szovjetunió hőse érdemrendet.
    - Erich Hartmann vadászpilóta: a világ mai napig is legeredményesebb vadászpilótája, 352 légigyõzelmet ért el, túlélte a háborút és a szibériai hadifogságot is.
    - Kosztka Vilmos százados: 1942. január 13-án az ötszörös túlerővel támadó 40. szovjet hadsereg alakulataival szemben hosszú órákon át tartotta Uriv község nyugati kijáratát, megmentve ezzel több más magyar egységet. Példát mutatott a bátorságból és elszántságból a hihetetlenül nehéz körülmények közepette.

Utoljára hagytam a személyes visszaemlékezések és naplók bemutatását. Egy hadművelet eseményeit és csatáit nem lehet jól illusztrálni a résztvevők visszaemlékezéseik, a harctéri jelentések nélkül. Nézzünk meg egy-két szívbe markoló visszaemlékezést:

- „Igyekeztünk mindennel fűteni. A földeken elvétve találtunk kukoricagórét. Tudvalevő, ezt nagyon szeretik az egerek. Belefészkelnek. Amint szétkapkodtunk egy-egy ilyen kupacot, hogy vigyük a tűzre, kiperegtek az egerek. Egyet-kettőt tudtak mozdulni a havon, s megfagytak. Hát ilyen a 40 fokos hideg..."

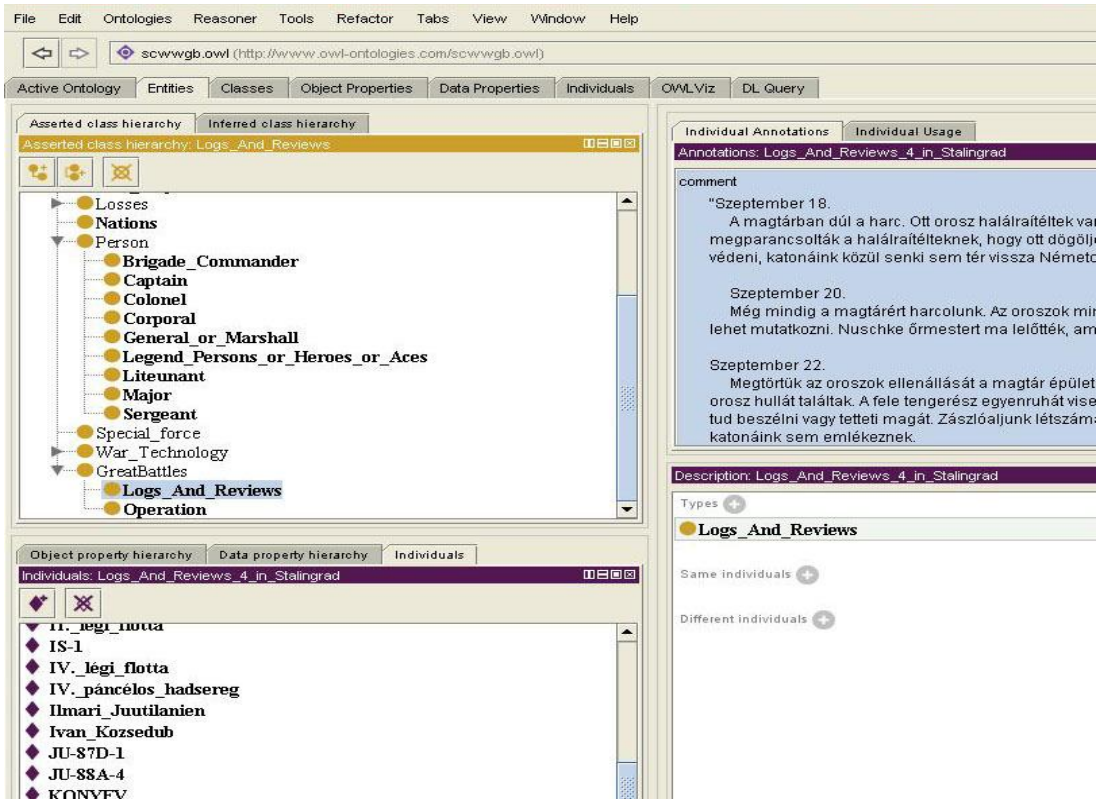
  – Sára Sándor filmjéből idézet

- „A magtárban dúl a harc. Ott orosz halálraítéltek vannak; a zászlóaljparancsnok azt mondja: "A komisszárok megparancsolták a halálraítélteknek, hogy ott dögöljenek meg a magtárban". Ha Sztálingrád minden épületét így fogják védeni, katonáink közül senki sem tér vissza Németországba. Ma levelet kaptam Elzától, győzelemmel vár haza..." – V. Csujkov tábornok visszaemlékezése a sztálingrádi harcokról

Protege

Nagy segítséget adott a **Protege** [4] ingyenes és nyílt forrású szoftver, mely lehetővé tette ontológiám elkészítését. Az ontológia eredetileg *OWL 1.0*-ás szabvány alapján készült, melyet a **Protege 3.0** támogatott. A Protege alapvetően tudásalapú alkalmazások létrehozására és fejlesztésére hivatott létrejönni. A Protege igyekszik implementálni a tudásalapú modell lehetőségeit, tartalmazza a létrehozástól kezdve a vizuális megjeleníthetőséget, a különböző megszorításokat, logikai viszonyokat, stb., nagyon sokféleképpen manipulálhatjuk ontológiánkat.

Könnyen bővíthető az internetről letölthető bővítményekkel, mint például a Graphviz, mely segítségével gráfok formájában jeleníthető meg az ontológia. Egyedire szabható a grafikus felület előre definiált stílusokkal. Minden fontosabb menüpontot külön fülekre (tab-okra) lehet bontani és különálló ablakokban lehet megjeleníteni. Az osztályokat, tulajdonságokat, egyedeket könnyen, gyorsan lehet szerkeszteni, újat létrehozni, törölni vagy annotálni. Néhány kattintás segítségével lehetőség nyílik az osztályok, tulajdonságok közötti viszonyok, relációk beállítására. Az alábbi ábrán láthatóak a legfontosabb funkciói:



5. Protege 4.0 beta főmenüje

6. A „Great Battles of Second Word War" ontológia főbb osztályainak hierarchiája

A *Rubicon* webontológia kezelő-rendszer

A **Rubicon** első ránézésre egy átlagos honlapnak tűnik, különböző típusú információkat szolgáltat a honlapot böngészők számára, mint például képeket a Második Világháborúról, linkeket nyújt más hasonló témájú weboldalakhoz, stb., de egy lényeges extra szolgáltatással rendelkezik, ez pedig nem más, mint a szemantikus keresés támogatása, így már nemcsak egy egyszerű honlap csupán. Ehhez a szolgáltatáshoz a „Great Battles of Second World War" című ontológia biztosítja a nélkülözhetetlen tudásbázist. Rögtön felmerül a kérdés, hogy miért is szükséges feldolgozni ezt az ontológiát. A válasz nagyon egyszerű, egy felhasználó számára (legyen bármilyen szintű) nehéz átlátni az ontológiát tartalmazó dokumentumot, és ebben a formában nehezen „emészthető". Az ontológiára támaszkodva azonban a kezelő-rendszer válaszokat tud adni a felhasználó kérdéseire az adott hadjáratokkal kapcsolatban.

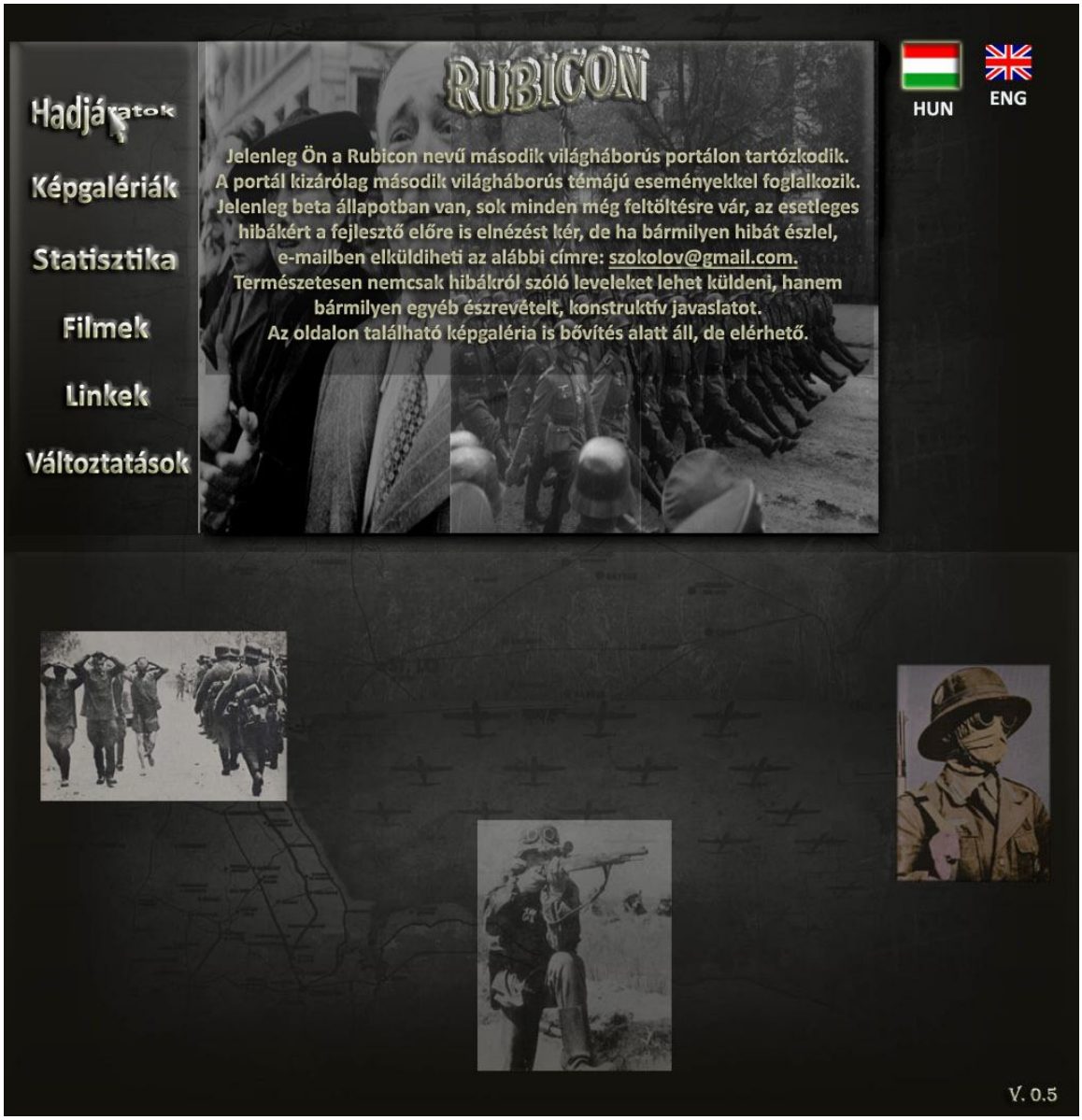
Jelenleg még a **Rubicon** nem publikus, csak belső szerveren elérhető.

A **Rubicon** bemutatását két részre osztottam:

1. Felhasználói szemszögből ismertetem a kezelő-rendszer funkcióit, a grafikus interfészét képekkel illusztrálva.
2. Fejlesztői szemszögből pedig elemzem a kezelő-rendszert, milyen technológiával készült, milyen nehézségek merültek fel, hány rétegen keresztül kommunikál a rendszer, stb.

A Rubicon felhasználói szemmel

A **Rubicon** futtatásához két dologra van szükség csupán: internetre és egy böngészőre (*Mozilla Firefox* és *Microsoft Internet Explorer* a támogatott böngészők). Ha ezek a feltételek adottak és sikerült elérni a honlapot, akkor a következő kép fogadhat:

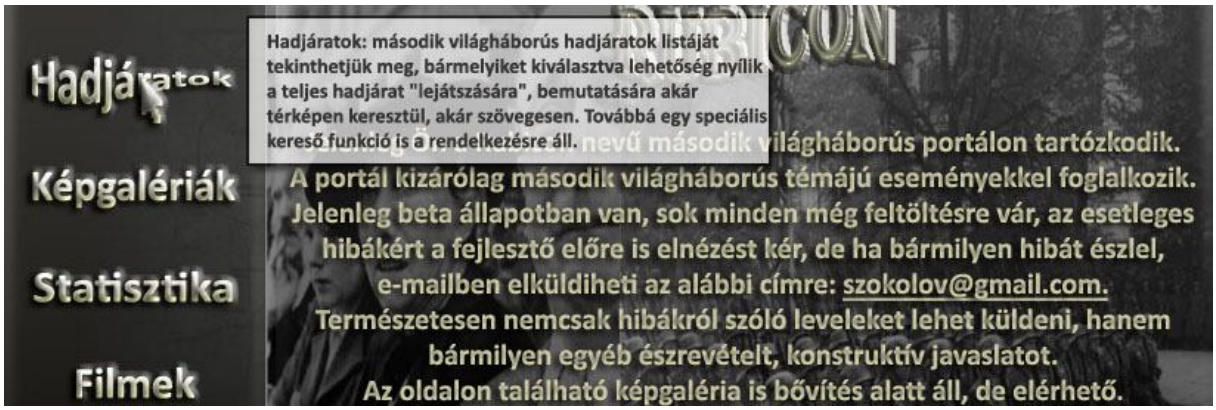


7. A Rubicon főoldala

A felhasználót alapból a főoldal fogadja, ahol alapvető információkat kap a honlapról, megtekintheti az egyes menüpontokat, melyek alatt elérheti az egyes szolgáltatásokat.

Elsőként nézzük meg a GUI[9] felépítését, komponenseit. A végfelhasználó számára fontos, hogy egy jól átlátható, könnyen kezelhető felületet kapjon. Ha egy opciót sokáig kell keresgetnie, esetleg nem is találja meg, az nem létezik a számára. Egyes GUI tervezők azt is ellenőrzik, hogy az adott oldalon mennyit kell kattintani, az egér mindkét gombját kellett-e használni, stb., így tökéletesítve a grafikus felületet a felhasználó számára.

Ezen a honlapon csak a balgomb használata szükséges a böngészgetéshez. A főoldalon látható, hogy több komponensre van felosztva a GUI, külön helyen találhatóak a gombok, a honlap ismertetése, a kiválasztott nyelv, ezek a grafikus felület felső részét foglalják el, míg az alsó rész alapból néhány képet mutat és a verziószámot. Amikor kiválasztunk egy opciót, a megfelelő gombra kattintás után megjelenik számunkra a kiválasztott menüpont a GUI alsó felén, lecserélve az előző tartalmat. Ha egy gomb fölé visszük az egérkurzort, akkor megváltozik a gomb kinézete, jelezve ezzel, hogy éppen az adott gomb lesz az aktuálisan kiválasztható opció, továbbá mikor megállunk egy gomb felett, akkor aktiválódik az úgy nevezett *hover*[10] esemény, aminek a hatására egy kis ablak jelenik meg, ami rövid, vázlatos leírást ad a megtekinteni kívánt menüpont tartalmáról. A következő ábrán erre láthatunk példát:



8. A hover eseményre felugró, súgóablak

A súgóablak segít a gyorsabb navigálásban, kiszűrhető hamar, hogy mely funkciók érdekelnek. Most már el tudjuk érni az egyes menüpontokat, ezek után vizsgáljuk meg, milyen lehetőségeket nyújtanak számunkra.

[9] GUI (**G**raphical **U**ser **I**nterface): grafikus felhasználói felület

[10] Hover: egy grafikus komponens felett megjelenő egérkurzor hatására kiváltódó esemény

A Rubicon szolgáltatásai

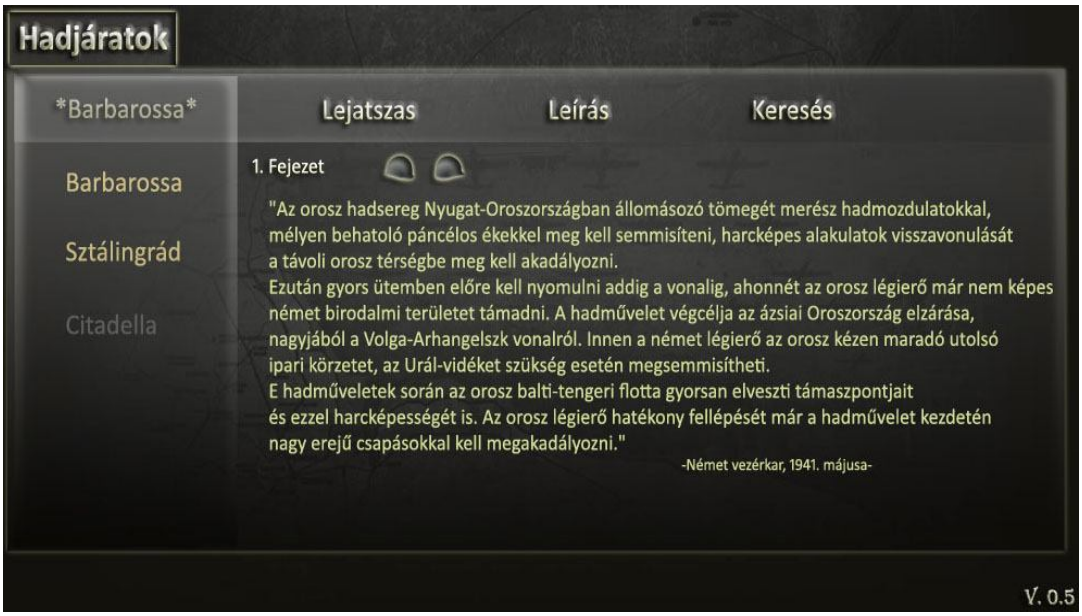
I. *Hadjáratok*: ez az opció az egyik legérdekesebb része a kezelő-rendszernek, ugyanis lehetőség nyílik a hadjáratok részletes elemzésére három további alopció segítségével. A megvizsgálandó offenzívát egy listából választhatjuk ki. A kiválasztott elem *-gal megjelölve fog megjelenni. A nem kiválasztható, de már hamarosan aktiválásra kerülő elemek leszürkítve lesznek. A listának még nincsenek görgető funkciói, mivel egy-egy hadjárat teljes elkészítése nagyon időigényes (sok háttérmunka szükséges egy komplett offenzíva összeállításához), így lassan fog bővülni a lista. Tekintsük meg az egyes almenüpontokat egyesével:

   a. *Lejátszás:* a kiválasztott hadjáratról egy interaktív kisfilmet láthatunk, amit bármikor szüneteltethetünk, megállíthatunk, újraindíthatunk tetszőleges időpontban. Egy kétszeresen nagyítható hadműveleti térképen láthatjuk az egységek mozgását, és hogy milyen várost foglaltak éppen el. A térkép természetesen méretarányos, az egyik fél (általában a támadófél) mindig pirossal jelenik meg, míg a másik fél kék színnel (általában a védekező fél). Nem ország szerint kapják a színeket, ez azt jelenti, hogy az egyik fél a *Tengelyhatalmak*[11] közé, míg a másik a *Szövetségesek* közé tartozott. Az egységek téglalapjain látható az elnevezésük. Méretben eltérnek az egyes téglalapok az egység típusától függően, a legnagyobb méretű a *hadseregcsoport*, illetve a *front*, a *hadseregen* kívül nincs alacsonyabb szintű egység, ami megjelenhetne a térképen:

[11] Tengelyhatalmak: 1936. november 1-jén jött létre egy szövetség elsőként Németország és Olaszország között (innen ered a „Berlin-Róma tengely” elnevezés), melyhez később hét ország csatlakozott.

9. A Rubicon hadjáratok menüpontjának lejátszás almenüpontja

b. *Leírás*: a kiválasztott offenzíváról kaphatunk fejezetekre bontott leírást, melyek között előre-hátra lapozhatunk. Kronológiai sorrend szerint olvashatjuk a fejezeteket.



10. A Rubicon hadjáratok menüpontjának leírás almenüpontja

c. *Keresés*: a keresés a legösszetettebb funkciója a weboldalnak. Kétféle keresést hajthatunk végre:

   i. Egyszerű szöveg alapú keresés: az összes tartalomra vonatkozik, a keresés eredménye egy felugró ablakban jelenik meg. A találatok listáját tetszőlegesen lehet lapozni.

   ii. Speciális keresés: kulcsszavakon alapul. Ez azt jelenti, hogy meg kell adnunk combobox-ok segítségével, hogy mit is szeretnénk keresni. A keresés hatásköre a kiválasztott hadjáratra vonatkozik csupán. Alapból hat fő keresési kategóriát lehet megadni, azokon belül egy listából lehet a konkrét alkategóriát kiválasztani. Az egyes kategóriákat tetszőlegesen lehet kombinálni. A keresés eredménye szintén egy felugró ablakban fog megjelenni. A találatok listáját tetszőlegesen lehet lapozni.

   Tegyük fel, hogy a *Barbarossa* hadjárat során felmerült szovjet emberveszteség halottjait szeretnénk megtudni (elnézést, hogy a példa ilyen szomorú és drasztikus), ehhez ki kell választanunk a szovjet felet, mint *résztvevő felet*, majd beállítani az *emberveszteségek* feltételek közül az elesetteket. A feltételek kiválasztásának sorrendje tetszőleges.

   A speciális keresés felett szintén lehet súgóablakot találni, ami röviden leírja a funkció használatát. A keresési feltételek alapból üresek, vagyis a „nincs kiválasztva” felirat jelenik meg. Tekintsük meg ezt az előbbi példát képeken is, előbb a keresési szempontok megadást láthatjuk, majd a keresés eredményét:

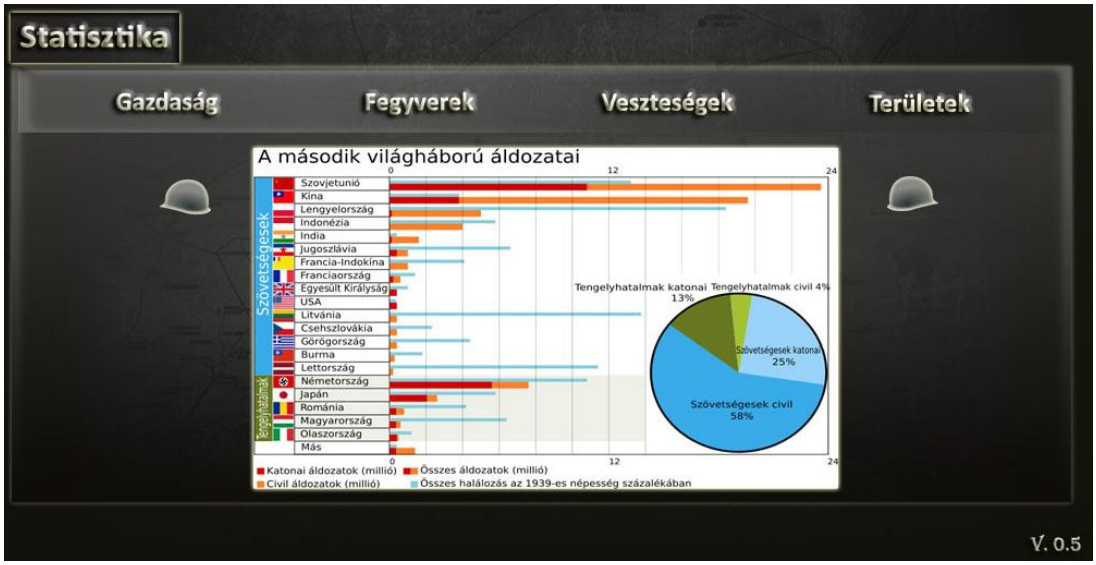
11. A Rubicon hadjáratok menüpontjának keresés almenüpontja

12. A Rubicon hadjáratok menüpontjának keresés almenüpontjának egy felugró ablaka

II. *Képgaléria*k: a galériák opció elengedhetetlen része egy ilyen témájú honlapnak. Egyenlőre öt kategóriára felosztva csoportosítottam a rendelkezésemre álló képeket. Ami biztosan bővülni fog mind tartalmilag, mind új kategóriával. Esetleg ha szükséges, alkategóriákkal lehet még szélesíteni a galéria spektrumát. Természetesen a galéria nem tartalmaz tiltott önkényuralmi jelképeket, hiszen ez az oldal politika-mentes, a Második Világháborút a konkrét eseményein keresztül igyekszik az érdeklődő elé tárni. Minden képet lehet nagyítani, így közel az eredeti méretben láthatóak, nyilván a nagyobb 1600*1200-as vagy attól is nagyobb felbontású képeket nem célszerű megjeleníteni. Az alábbi ábrán láthatóak lesznek ezek a kategóriák és maga a galéria menüpont, ahol egyszerre hat képet látható (a sisak textúrájú gombokkal pedig lapozni lehet a képek között):

13. A Rubicon képgaléria

III. *Statisztikák:* a weblap témájából eredendően szükségesnek éreztem vizualizálni azt a rengeteg sok statisztikai adatot a háborús veszteségektől kezdve, a legyártott tankok számán, a megtermelt búza mennyiségén, egy-egy ország területi változásain keresztül jól össze lehet foglalni egy-egy diagrammal. Egyenlőre négy féle statisztikából válogathatunk, szintén lehetőség van a lapozásra és a statisztikák kinagyítására. Az alábbi képen jól látszik, hogyan is néz ki ez a menüpont:

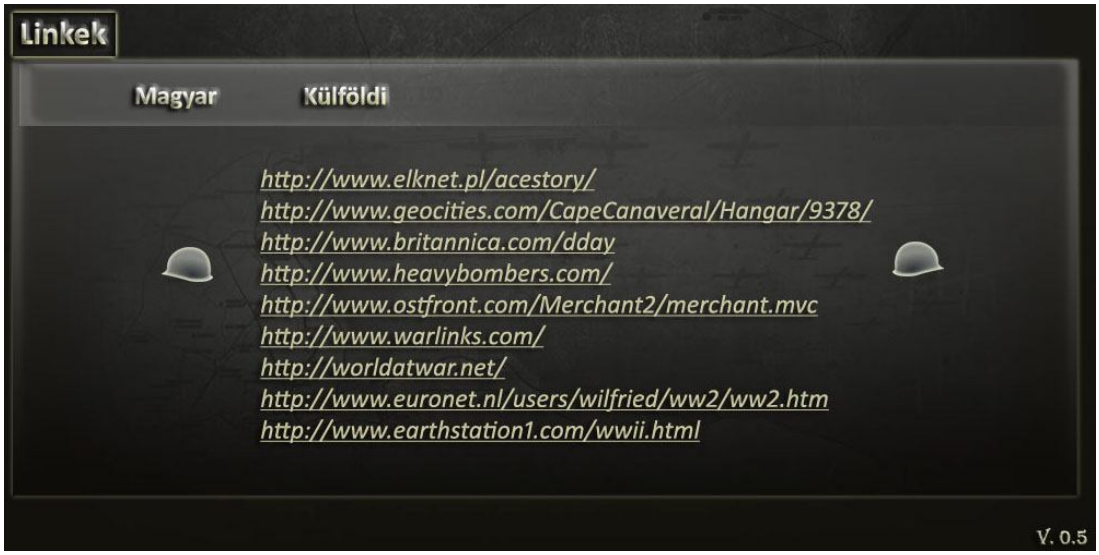


14. A Rubicon statisztika menüpontja

IV. *Filmek*: ezen opción keresztül különböző dokumentum, illetve mozifilmek címei érhetőek el, ezek linkek csupán, maguk az alkotások itt a honlapon nem elérhetőek. Filmek menüpont:

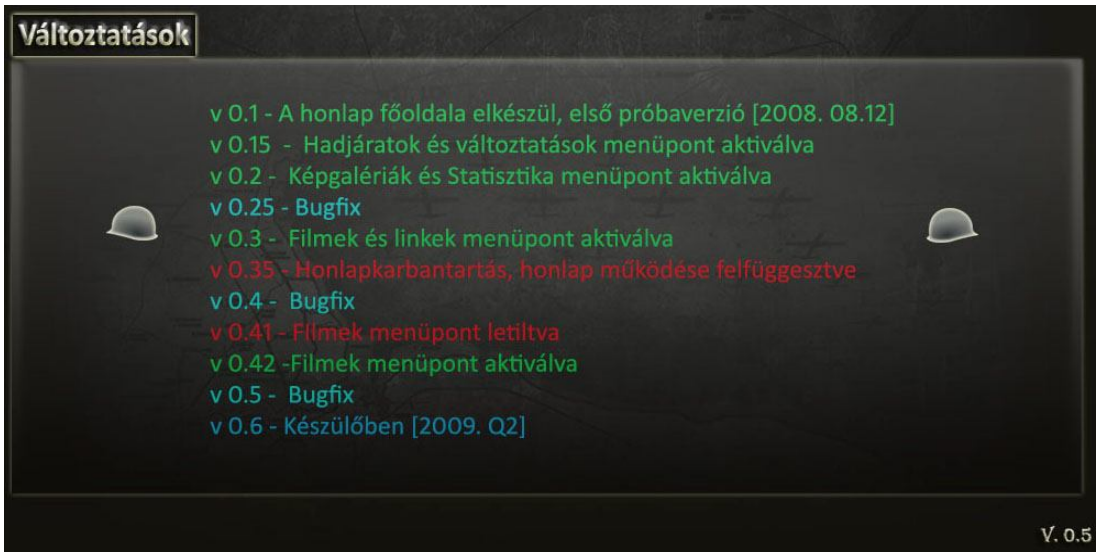
15. A Rubicon filmek menüpontja

V. *Linkek*: a linkek menüponton keresztül más Második Világháborúval kapcsolatos weblapok elérhetőségei közül böngészhetünk, külön csoportosítva a külföldi oldalakat és a magyar oldalakat.



16. A Rubicon linkek menüpontja

VI. Változtatások: ez az opció sok honlapról hiányzik sajnos, pedig ez is hasznos információ, hiszen ebből lehetne látni, hogyan fejlődött ki az adott weblap és hogy még milyen újításokra, módosításokra, karbantartásra lehet számítani a közel jövőben. A különböző üzeneteket különböző színnel jelöltem, hogy átláthatóbbak legyenek.



17. A Rubicon változtatások menüpontja

Amit a felsorolásba nem vettem már bele, az a nyelvválasztó funkció, amely a weboldal jobb felső sarkában található. Ha a felhasználó nem beszéli az alapból beállított magyar nyelvet, akkor átválthat angol nyelvűre, vagy egy újabb verzióban akár más nyelvre is.

Tervezett újítások, bővítések

Ami hiányzik a **Rubicon**-ból első ránézésre, az a *fórum*. Jelenleg nincs lehetőség vitákra, az esetleges kérdések megbeszélésére, így nincs közösségépítés. Nem lehet saját képeket feltölteni, megosztani. A későbbiekben jó lenne, ha egy felhasználó saját profilt tudna létrehozni, utána képeket feltölteni, fórumozni, stb.

Új multimédiás menüpontot képzeltem el, ahol zenéket, indulókat, jármű- és fegyverhangokat, videókat lehetne elérhetővé tenni.

A tartalmi bővítést fix időközönként érdemes elvégezni, új hadjáratot felvenni/bővíteni a jelenlegieket, még több statisztikát összegyűjteni, nagyobb és jobban kategorizált képgalériát készíteni. Új keresési feltételeket lehetne megadni, ami esetleg új felületet, más stílusú menüt von maga után.

Még több kényelmi funkciót szeretnék biztosítani a felhasználók számára, mint például extra gombok elhelyezése, hogy gyorsabban lehessen váltani az egyes funkciók között. Ilyen lehetne a hadjáratok menüpont esetében az ugrás a megadott fejezetre.

A hadjáratok opciói közül a lejátszás alopciót szeretném bővíteni néhány újdonsággal, mint például egy-egy összecsapás megjelenítésével (légitámadásoknál egy rövid videót mutatna, hogy a bombázógépek lecsapnak a célpontjaikra, páncélosütközeteknél szintén egy rövid videót láthatnánk a harcoló tankokról), statisztikai adatok jelenhetnének meg, ha egy egység fölé viszem a kurzort (hány hadosztálya van egy hadseregnek, mennyi a személyi állománya, fegyverzete). Ezáltal még színesebben lehetne bemutatni egy hadjárat történetét.

A Rubicon fejlesztői szemmel

Ebben a fejezetben szeretném ismertetni a **Rubicon** webes alkalmazás fejlesztése során használt technológiákat, a fontosabb fájlokat, könyvtárszerkezetét, az alkalmazás felépítését, az egyes rétegek közötti kommunikációt, a kezelő-rendszer feltöltését adatokkal.

A kezelő-rendszer felépítése

Elsőként lássuk a könyvtárszerkezet felépítését a szerveren (a főbb könyvtárak és fájlok láthatóak csak):

Rubicon (**R**oot **D**irectory)

- classes
  - includes
    - *campaignScript.scr*
    - *mainStyle.css*
    - *popup.js*
    - *specialStyle.css*
  - *campaign.php*
  - *changes.php*
  - *datas.php*
  - *gallery.php*
  - *links.php*
  - *menu.php*
  - *mainWindow.php*
  - *movies.php*
  - *search.php*
  - *tokenizer.php*
- lang
  - ENG
    - *eng.loc*
  - HUN
    - *hun.loc*

- semantic_media
    - includes
    - libs
    - specials
- storage
    - changes
    - gallery
        - general
        - persons
        - weapons
        - vehicles
        - posters
    - links
    - movies
- *index.php*

A logikailag összetartozó részeket igyekeztem egy könyvtárszinten csoportosítani. A *classes* mappában szerepeltetett fájlok felelnek az alkalmazás működtetéséért, a menüpontok kezeléséért, stb. Az *includes* alkönyvtárban szereplő .css fájlok a honlap stílusáért felelősek, a *popup.js* pedig a felugró ablakot kezeli, az *campaignScript.scr* pedig a hadjáratok szimulálásához szolgáltat utasításokat.

A *lang* könyvtárban a nyelvi lokalizációs fájlok találhatóak, ezeket tudja feldolgozni az alkalmazás, attól függően, hogy milyen nyelv lett beállítva, az állományokban osztályonként csoportosított sztring-azonosító értékpárok szerepelnek. A beállított nyelvnek megfelelő sztringeket használ a kezelő-rendszer, minden sztringre egy egyedi azonosítóval lehet hivatkozni. Az azonosítóval ellátott sztringek az egyes fájlokba vannak elhelyezve. Egy ilyen nyelvi fájl a következőképpen néz ki:

...

[gallery.php]

1 = Életképek

2 = Személyek

3 = Fegyverek

4 = Járművek

5 = Plakátok

[links.php]

…

Ha a *gallery.php* kódjában egy sztringet meg akarok jeleníteni a képernyőn, akkor egy „$ID|” sztringet kell elhelyeznem a kódban, ahol az ID egy konkrét egész érték. A program az ID alapján behelyettesíti a megfelelő sztringet a lokalizációs fájlból. Ha új sztringet akarunk létrehozni, akkor csak a lokalizációs fájlba kell felvenni ezt a sztringet egy megfelelő azonosítóval. A több helyen előforduló, azonos sztringeket egyetlen közös csoportba soroltam a *commonStrings.php*-ba. Így a duplikációt sikerült kiküszöbölnöm.

A *semantic_media* mappa egy ingyenes („free license”) alkalmazás-csomagot tartalmaz, aminek a segítségével fel tudtam dolgozni a webontológiámat, és ki tudtam nyerni a szükséges adatokat a kereső funkcióhoz. Ezt a csomagot részletesebben egy későbbi fejezetben fogom ismertetni.

A *storage* könyvtárban a honlap tartalmi része található úgy, mint a képgaléria, azon belül alkategóriák szerint rendezve a képfájlok, a linkek állománya és filmek listáját tartalmazó fájl. Könnyen bővíthető új kategóriákkal, új tartalmakkal.

Az *index.php* a főoldal megjelenítéséért felelős kódot tartalmazó fájl.

A kezelő-rendszer feltöltése adatokkal

A fejlesztési idő felét a világháborús adatok, képek, statisztikák, naplók tanulmányozása, összegyűjtése vette igénybe. Fontos volt, hogy a források minél pontosabbak legyenek. Sokat segítettek a háborús veteránokkal és a hadtörténetben jártas személyekkel történő beszélgetések. Az ilyen visszaemlékezések és az élőszóban hallott történetek tették lehetővé az egyes hadjáratok, csaták színesebb, részletesebb bemutatását.

Kíváncsi voltam, hogyan élték meg, hogyan látták a különböző nemzetek a Második Világháború eseményeit, csatáit, borzalmait. Törekedtem arra, hogy külföldi irodalmakat, folyóiratokat, könyveket és dokumentumfilmeket is megtekintsek. Így a résztvevő országok szemszögéből láthattam a világháború részleteit. A szöveges alapú forrásaimat könyvekből, folyóiratokból, honlapokról szedtem össze. A képeket pedig az interneten kutattam fel.

Alkalmazott technológiák

Az alkalmazás fejlesztése során alkalmazott főbb technológiákat szeretném röviden ismertetni ebben a fejezetben. A felsorolásra kerülő technológiákat a *Wamp Server* mellett használtam, ami biztosította a PHP5, Apache 2 és a MySQL támogatást. Ez a szoftver Windows operációsrendszereken képes feltelepíteni egy komplett webszervert.
Tekintsük át a kezelő-rendszer esetében alkalmazott fontosabb technológiákat:

- PHP
    - Az alapvető funkciók működtetése.
    - A keresés eredményének szerializációja.

- Javascript
    - Felugró ablakok kezelése.

- Semantic MediaWiki (SMW) [6]
    - OWL ontológiák importálása, feldolgozása.
    - Következtetés az OWL ontológiák alapján felépülő tudásbázisból.
    - SQL támogatás.
    - Keresés eredményének exportálása.

- *CampaignScript*
    - A hadjáratok bemutatásáért felelős utasításokat magában foglaló szkriptnyelv.

PHP technológia alkalmazása

Vizsgáljuk meg a PHP-ban elkészített osztályokat, csupán néhány főbb osztályra térek ki.

1. MainWindow osztály: a felhasználói felületen látható grafikus komponensek gyűjtőosztálya, ami egy általános interfészt nyújt az egyes menüpontok számára, melyek bővítik az osztályt.
   A honlap felülete két része lett osztva, a háttér, az alapvető menüpontok mindig láthatóak, viszont az alsó rész frissül, ha valamelyik opciót kiválasztottuk vagy az opción belül valamilyen műveletet hajtottunk végre.
   A felső részen csak a nyelvválasztás esetén frissül be új tartalommal. Fontosabb attribútumok és függvények:
   a. Attribútumok:
      i. data: adatkezelő osztálypéldány.
      ii. menuInstance: az aktuálisan kiválasztott menü.
      iii. selectedLanguage: a kiválasztott nyelv.
      iv. localizationFile: lokalizációs fájl.
   b. Metódusok:
      i. init(): inicalizálja az osztály adattagjait.
      ii. initDBconnection(): adatbáziskapcsolat létrehozását végzi, ha nem sikerül, akkor figyelmeztető üzenet jelenik meg a honlapon, hogy „technikai problémák miatt az oldal nem üzemel”.
      iii. makeLocalization(): beolvassa, majd feldolgozza a lokalizációs fájlt és bárhol, ha a kódban hivatkozás található egy sztringre, akkor ID alapján lecseréli.
      iv. functionOnEvent(): egy menüpont kiválasztásakor kiváltodó eseményt kezeli le, továbbá a honlap alsó részét frissíti a megfelelő tartalommal, és értéket ad a menuInstance változónak.
      v. onAnimate(): a gombok aktív és inaktív textúráinak váltogatásáért felelős függvény.

2. CampaignMenu osztály: a „hadjáratok opciót” kezelő függvényeket tartalmazza, a Menu osztály leszármazottja. A két legösszetettebb funkciót kezeli, a hadjáratok interaktív lejátszását és a keresést. Fontosabb attribútumok és függvények:
    a. Attribútumok:
        i. subMenu: az aktuálisan kiválasztott almenüpontot tárolja.
        ii. selectedCampaign: az aktuálisan kiválasztott hadjáratot tárolja.
        iii. campaignList: a megtekinthető hadjáratok listáját tartalmazza.
    b. Metódusok:
        i. initCampaign(): inicializálja a CampaignMenu osztály adattagjait először alapértelmezett értékekkel. Ha már van utoljára beállított érték, akkor arra az értékre fogja beállítani a CampaignMenu osztály változóit.
        ii. playCampaign(): a megtekinteni kívánt hadjárat lejátszását indítja el. Lepéldányosít egy Campaign osztályt, és értékül adja a subMenu változónak. A Campaign osztályt egy későbbi pontban fogom részletezni.
        iii. showCampaignByDate(): a subMenu változónak egy CampaignDescription osztálypéldányt ad értékül. Az aktuálisan kiválasztott hadjáratról nyújt tájékoztatást kronológiai sorrendben, fejezetekre bontva. A fejezetek tartalmát az ontológia alapján építi fel.
        iv. initSearch(): a search osztályt példányosítja le, és erre állítja be a subMenu változót. A keresést és a speciális keresést kezeli. A search osztályt a következő pontban fogom részletezni.
3. Search osztály: a hadjáratok menüpont keresés funkcióját implementálja. Kétféle keresést tesz lehetővé, az első egy sima kulcsszavak nélküli keresés, a második pedig egy speciális, kulcsszavas. Mindkét keresés a „Great Battles of Second World War” ontológia által nyújtott tudásbázisra épül. A keresés megvalósítását a *Semantic MediaWiki* programcsomag tette lehetővé. A programcsomagot, mint alkalmazott technológiát és a keresés megvalósítását egy későbbi pontban fejtem ki. A keresési feltételek kulcsszavait XML állományból olvassa be. Fontosabb attribútumok és metódusok:

a. Attribútumok:
    i. searchCondition: a felhasználó által megadott keresési feltételeket tartalmazó sztring.
    ii. searchConditions: az összes lehetséges kulcsszót tartalmazó tömb.
    iii. searchResult: a keresés eredményét tároló tömb.
b. Metódusok:
    i. comboOnChange(): egy adott combobox elemkiválasztását implementálja.
    ii. initCombos(): combobox-ok létrehozását és feltöltését végzi.
    iii. showResults(): a keresés eredményét jeleníti meg egy felugró ablakon a searchResult tömb alapján.
    iv. startSearch(): ha a searchCondition sztring értéke nem üres sztring, akkor elindítja a keresést.

4. Campaign osztály: megjelenít egy méretarányos térképet, amin a hadjáratban résztvevő országok egységeit és a fontosabb városokat láthatjuk eltérő színezéssel. A harcoló egységek és városok külön objektumként (kétdimenziós koordinátákkal rendelkeznek) szerepelnek. A harcoló csapatok és a városok adatai XML állományokból töltődnek be. Az egyes egységeket valós időben mozgatja a rendszer, a megadott célpontok felé, szimulálva ezzel a csapatmozgásokat a térképen. Szabályos időközönként frissül a térkép, olyan hatást próbál kelteni a program, mintha tábornokokként figyelnék csapataink mozgását. A mozgatás esemény alapú, a *CampaignScript.scr*-ből betöltött és feldolgozott fájl utasításai hajtódnak végre. (Például ha egy egység rendelkezik célponttal (target), akkor támadásba lendül. Mikor elérte azt, újabb célpontot kaphat.) A *CampaignScript*-et egy későbbi fejezetben fogom bemutatni. Fontosabb attribútumok és függvények:
    a. Attribútumok:
        i. city: a bemutatott hadjárat városainak adatait tartalmazó tömb.
        ii. cpgScript: a *CampaignScript.scr* feldolgozott adatait tárolja.
        iii. unit: a bemutatott hadjárat során szereplő egységek adatait magában foglaló tömb.

b. Metódusok:
    i. initCities(): inicializálja a city tulajdonságot.
    ii. initMap(): beállítja a térképet, rajta az egységek kiinduló állását.
    iii. initUnits(): inicializálja a unit változót.
    iv. updateMap(): frissíti a csapatok pozícióját a térképen.
    v. updateUnits(): frissíti az egységek célpontját.

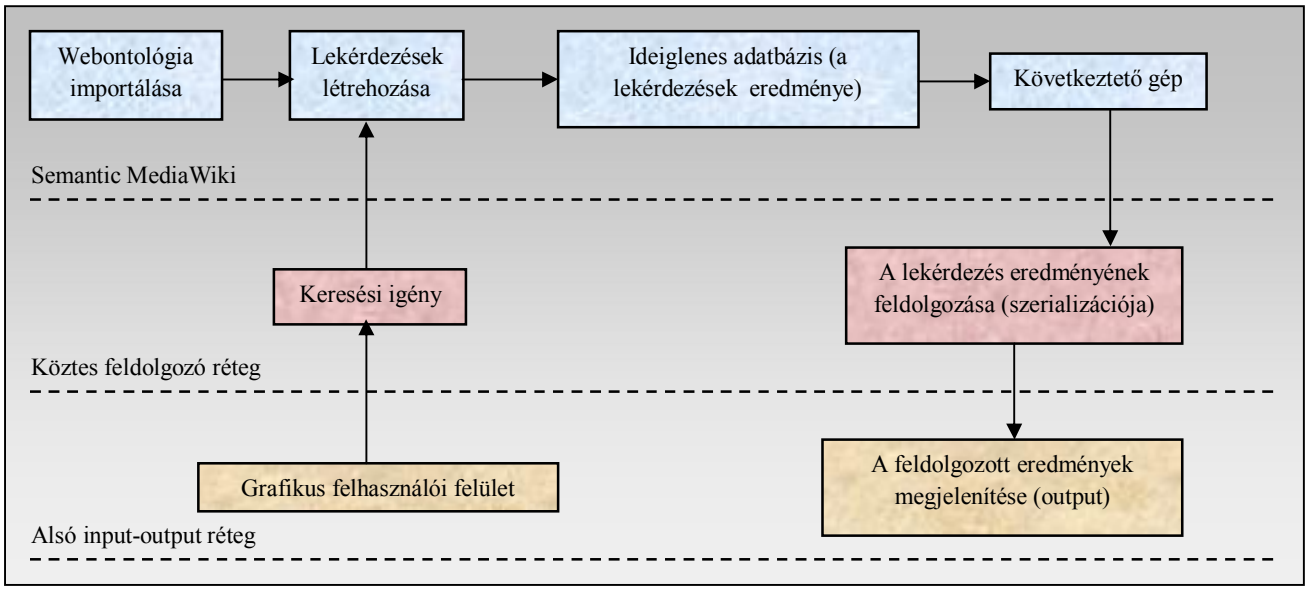
5. Data osztály: az alapvető adatbázisműveleteket, az XML állományok, továbbá a lokalizációs fájl és egyéb állományok feldolgozását megvalósító osztály. Fontosabb tulajdonságok és metódusok:
    a. Attribútumok:
        i. connection: adatbáziskapcsolat azonosítója.
        ii. dbName: az alkalmazás által használt állandó adatbázis neve.
        iii. host: az adatbázisrendszer elérési útvonala.
        iv. password: az adatbázisrendszer felhasználójának jelszava.
        v. user: az adatbázisrendszer felhasználója.
        vi. XMLParser: XML állományok feldolgozóját tartalmazó változó.
    b. Metódusok:
        i. connectToDB(): az adatbázissal létrehozza a kapcsolatot.
        ii. executeDBQuery(): végrehajt egy adatbázislekérdezést és visszatér a lekérdezés eredményével.
        iii. parseXMLFile(): feldolgoz egy paraméterként kapott XML fájlt és egy többdimenziós tömbként visszaadja a feldolgozás eredményét.
        iv. processLocFile(): beolvassa és feldolgozza a lokalizációs fájlt.
        v. processScriptFile(): egy .scr szkript fájlt dolgoz fel.

A Semantic MediaWiki technológia bemutatása

A PHP-ban elkészített fontosabb osztályok rövid ismertetése után nézzük meg fentebb említett Semantic MediaWiki technológiát. A Semantic MediaWiki-t a napjainkban sokak által használt és jól ismert MediaiWiki szoftver kiterjesztéseként fejlesztették ki. A MediaWiki segítségével a felhasználók rendszerezhették ismereteiket, az ismeretek mennyisége fokozatosan növekszik, ezért szükségessé vált ezeknek az ismereteknek a „szemantikus annotálása”. A Semantic MediaWiki egy komplex programcsomag, mely az egyszerű SQL lekérdezésektől az ontológiák feldolgozásáig nagyon sok új lehetőséget nyújt.

A Rubicon egy webontológia kezelő-rendszer, ami a „Great Battles of Second World War” webontológiát dolgozza fel, így válik lehetővé az egyes hadjáratok bemutatása és a keresési funkciók. Ehhez azonban egy komplex eszközrendszer támogatására volt szükségem. A webontológia egy OWL dokumentum, és a PHP-ban elkészített osztályoknak szüksége volt az OWL dokumentum feldolgozására, hogy fel tudják használni a belőlük nyerhető információkat, amelyekből a *Semantic MediaWiki* következtető gépének segítségével megvalósítható a keresési funkció.

A kezelő-rendszer ugyanis több rétegen alapul, az egyes rétegeknek kommunikálniuk kell egymás között a megfelelő protokollokon keresztül. Tekintsük meg az alábbi ábrát, amely a rétegek közötti kommunikációt jeleníti meg:



18. A kezelő-rendszer rétegeinek kommunikációja

Jól látható, hogy három fő réteget lehet megkülönböztetni. Az *alsó input-output réteg* magában foglalja a honlap megjelenítéséért felelős osztályokat. A grafikus felhasználói felületen adhatunk meg különböző inputokat, mint például egy keresési feltételt. A keresési feltétel egyfajta igényként jelentkezik, amit a *köztes feldolgozó réteg* értelmez és validál. Validálás után átadja az igényt a *Semantic MediaWiki*-t tartalmazó réteg felé.

Ezen a szinten a keresési igényből lekérdezést vagy lekérdezéseket hoz létre (ezek speciális SMW lekérdezések, amik ugyanolyan elven működnek, mint a MySQL lekérdezések). A lekérdezések eredményeképpen létrejön egy ideiglenes adatbázis. A következtető gép az ideiglenes adatbázis alapján kiszűri és visszaadja a keresési feltételnek megfelelő sorokat sikeres keresés esetén, egyébként nulla darab sort szolgáltat a keresés eredményeként.

Ezután a *köztes feldolgozó réteghez* kerül a keresés eredménye. Ha sikeres volt, akkor szerializálni kell az eredményt. Ez azt jelenti, hogy sztringgé kell alakítani, amit aztán az alsó input-output réteghez eljuttatva deszerializálni kell, vagyis a paraméterként kapott sztringet szét kell darabolni, majd egy tömbbe feltölteni. A tömb tartalma ezek után megjelenik a képernyőn egy felugró ablakban. A sikeres lekérdezéseket a rendszer eltárolja (cache), így ismétlődő keresések esetén azonnal vizualizálható az eredmény. Természetesen egy mennyiség után törlődnek a legrégebbi keresések eredményei.

CampaignScript technológia alkalmazása

A hadjáratok bemutatását szerettem volna interaktívabbá tenni, nem csupán néhány kép váltogatásával szemléltetni az eseményeket, az egyes hadmozdulatokat. Ekkor jutott eszembe az ötlet, hogy definiálni kellene egy fájlban a fontosabb utasításokat, melyek irányíthatják az egységeket, különböző parancsokat kaphatnának, ezeket az utasításokat egy saját szkript-nyelvben foglaltam össze.

A szkriptben megadott utasításokkal definiálhatjuk a hadműveletben szerepeltetni kívánt hadseregcsoportokat vagy hadseregeket. Az így definiált egységeket pedig mozgathatjuk a térképen egy megadott célpont felé (például egy elfoglalandó város) vagy parancsot adhatunk egy hadseregnek, hogy rohamozzon meg egy várost, tüzérségi támogatást kérhetünk egy adott célpont ellen, egyesíthetünk két sereget, stb.

A különböző hadjáratok szkriptjei külön fájlokban tárolódnak. Jelenleg a Barbarossa hadjárathoz létezik ilyen állomány, a *CampaignScript.scr*. A campaignScript típusú fájlokat .scr kiterjesztéssel láttam el. Ezeket a szkript fájlokat egy PHP-ban elkészített osztály dolgozza fel és értelmezi. A jelenlegi utasításkészlet nincs még kihasználva teljes mértékben, de lehetőséget ad a hadjáratok még részletesebb szimulációjához.

A campaignScript fájlok két részre bonthatóak, fejlécre és törzsre. A fejlécben történik a hadjáratban résztvevő egységek inicalizálása. A törzs részben pedig a hadművelet menetének a leírása történik, a leírás konkrét utasításokból áll, az egyes utasítások paraméterezhetőek. A campaignScript-ben az alábbi szimbólumokat különböztetjük meg:

- **{,}**: egy blokkot határoznak meg.
- **;**: utasítás végét jelző karakter.
- **//**: a szkriptfájlban elhelyezett megjegyzést jelöli, a jel után szereplő karaktersorozatot a sor végéig figyelmen kívül hagyja a szkriptet értelmező kód.

A szkriptnyelv a következő utasításokat tartalmazza:

- Alaputasítások:
    - army: egy egység létrehozására szolgál
    - command: egy komplett hadparancs megadására nyílik lehetőség.
    - event: a hadművelet egy fontosabb eseményét lehet definiálni vele.

- execute: végrehajt egy másik eseményt vagy parancsot.

- Hadparancsok:
  - artillery_strike: tüzérségi támogatást lehet kérni vele egy megadott célpontra, amit paraméterként kell megadni.
  - assault: egy város vagy egy másik egység elleni rohamra ad utasítást.
  - capture: egy egység elfogására vagy egy város elfoglalására ad utasítást.
  - air_strike: légitámadásra ad parancsot.
  - move: egy egység egy konkrét pont felé történő mozgatására adhatunk parancsot.
  - target: új célpontot jelölhetünk ki egységünk számára.
  - wait: paraméterként hozzárendelt ideig várakoztat egy megadott egységet.

- Kiemelt parancsok („<” és „>” szimbólumok között lehet csak használni őket):
  - allies/axis: a szövetséges és tengelyhatalmi oldalon harcoló egységek definiálására szolgál.
  - campaing: a szkript fájl kezdetét és végét jelöli.
  - head: a szkript fájl fejlécének megadására szolgál.

- Speciális parancsok:
  - digin: a kijelölt egység beássa magát az adott pozícióján.
  - embark/disembark: a levegőben vagy hajón szállított egységek ki- és berakodását teszi lehetővé.
  - reunit/split: egyesíthetünk és feloszthatunk egy egységet.
  - surrender: az adott egység megadja magát.

Tekintsünk meg egy szkript fájlrészletet, ami ez előbb felsorolt utasítások egy részét felhasználja:

```
<campaign>
  <head>
      <allies>
      army 19. hadsereg(Konev);
```

```
            army 50. hadsereg(Petrov);
             …
            </allies>

            <axis>
            army 4. hadsereg(Kluge);
            army 6. hadsereg(Rundstedt);
            …
            </axis>
    </head>
{
    event szmolenszk_bevetele;
   command 1;
   {
            target 4. hadsereg(Kluge) szmolenszk;
            move 4. hadsereg(Kluge) 25,50;
            wait 4. hadsereg(Kluge) 25;
            capture 4. hadsereg(Kluge) szmolenszk;
    }
    …
    command 5;
    {
            artillery_strike szmolenszk;
            air_strike szmolenszk;
            digin 6. hadsereg(Rundstedt);
    }
    …
}
</campaign>
```

A példában felsorolt parancsok segítségével először négy hadsereget definiál. Majd utasítást ad a 4. hadseregnek Szmolenszk városának elfoglalására. A következő utasításokban pedig légi- és tüzérségi csapásra ad parancsot, majd tartalékba helyezi a 6. hadsereget.

Összefoglalás

A diplomamunkám elkészítése olyan volt számomra, mintha egy kutatási, fejlesztési projektet bonyolítottam volna le. Aprólékos és részletes kutatómunka előzte meg a konkrét fejlesztési időszakot. A „Great Battles of Second World War" ontológia elkészítése során mélyre kellett ásnom a történelmi emlékek és források között, sok dokumentumfilmet és tévés előadást végignéztem. Olyan volt, mint egy időutazás. Természetesen folyamatosan ügyelnem kellett arra, hogy mentes legyen az ideológiáktól és a politikától. Úgy éreztem, hogy ez egy komoly téma, amit alaposan körbe kell járni. Még olyan emberekkel is sikerült beszélgetnem, akik megjárták a Második Világháborút, sokat segítettek az ő személyes visszaemlékezéseik.

Már korábban megjegyeztem, hogy hadtörténeti szempontból dolgoztam ki a témát, ezért is kapta az ontológia a „Második Világháború Nagy Csatái" címet. Egy Debrecenben végzett történész ismerősöm felajánlotta, hogy ismerteti az ontológiát a budapesti Hadtörténeti Múzeumnak, de nem tudtak anyagi támogatást biztosítani.

A Rubicon elkészítése később, az ontológia elkészülése után jött szóba. Mivel már ismert volt előttem a szemantikus világháló ötlete, az ontológiák fogalma, ezért felmerült az ötlet, hogy az ontológiát bárki számára felhasználhatóvá, elérhetővé kellene tenni. Péterrel sokat gondolkodtunk és kerestük a megoldást, hogy milyen formában lenne jó mindezt véghez-vinni. Hosszas keresgélés és próbálgatás után a választás a *semantic mediawiki* technológiára esett. A Rubicon webes alkalmazás így egy elég komplex szoftver lett. Igyekeztem egy interaktív, hasznos portált fejleszteni. Más honlapokon már láttam animált képsorokat, videókat különböző hadműveletekről, ezeket a „bemutatókat" szerettem volna továbbfejleszteni, így találtam ki a campaignScript technológiát, hogy látványosabbá tegyem az egyes hadjáratokat.

Fontosnak tartom zárszóként kiemelni a szemantikus web jelentőségét, mert forradalmasítani fogja a jelenlegi világháló szemléletét, működését. Jelenleg még kevesen foglalkoznak a webfejlesztők közül ezzel a témával, de egyre nagyobb teret hódít. Az internet jelenleg is temérdek mennyiségű információt szolgáltat, ami óhatatlanul igényeli az egyszerűbb és pontosabb elérhetőséget, az automatikus értelmezést. Ehhez azonban olyan fogalmi rendszerekre van szükség, melyek automatikus osztályozást tesznek lehetővé, ilyenek például az ontológiák.

A képekhez, cikkekhez szolgáló források

- ”19. Barbarossa hadműveleti terv, 1941. 06. 22.” ábra:
  - http://carlisle-www.army.mil/usawc/Parameters/99spring/barbaro2.gif

- A **Rubicon** galériájában szereplő képek:
  - http://www.bibl.u-szeged.hu/bibl/mil/ww2/map/index.html
  - http://www.dean.usma.edu/history/web03/atlases/ww2%20europe/WWIIEuropeIndex.html
  - http://www.netlabor.hu/roncskutatas/modules/newbb/viewtopic.php
  - http://www.janssen-militaria.com/ebenemael.html
  - http://galerie.valka.cz/displayimage.php
  - http://www.combatgallery.com/displayimage-topn-0-393.html
  - http://hu.wikipedia.org/wiki/F%C3%A1jl:World_War_II_Casualties-hu.svg
  - http://www.archives.gov/research/ww2/photos/

- A második világháborús cikkek, leírások, az ontológia és a kutatómunkám forrásai:
  - C.Jorgensen - C.Mann: Harckocsihadviselés
  - Földi Pál: A „Barbarossa” hadművelet
  - Földi Pál: Kurszk
  - Földi Pál: A sztálingrádi csata igaz története
  - Hadi Krónika: Képek a II. világháború történetéből c. sorozat 1-42, 64-69
  - Joachim Peiper: Banditák nyolc óránál
  - Jónás Edit, Rónaszegi Éva: A Második Világháború Krónikája
  - Román István: A vörös kolostor
  - http://en.wikipedia.org/wiki/Battle_of_Crete
  - http://hu.wikipedia.org/wiki/Második_világháború
  - http://hu.wikipedia.org/wiki/2._magyar_hadsereg
  - http://www.mult-kor.hu/
  - http://www.bbc.co.uk/history/worldwars/wwtwo/
  - http://www.worldwar-2.net/
  - http://www.besthistorysites.net/WWII_Special.shtml

Irodalomjegyzék

[1] Szeredi Péter, Lukácsy Gergely, Benkő Tamás: A szemantikus világháló elmélete és gyakorlata, Typotex, Budapest, 2005

[2] „Az OWL Web Ontológia Nyelv – Alkalmazási esetek és követelmények”: http://www.w3c.hu/forditasok/OWL/REC-webont-req-20040210.html#usecase-ubiquitous

[3] OWL 1.0, W3C ajánlás (2004. február 10.):

- http://www.w3.org/TR/2004/REC-owl-features-20040210/
- http://www.w3.org/TR/owl-features/

[4] OWL 1.1, munkaterv (2006. december 19.):

- http://www.w3.org/Submission/2006/SUBM-owl11-overview-20061219/

[5] OWL 2.0, munkaterv (2008. október 10.):

- http://www.w3.org/TR/2008/WD-owl2-syntax-20081008/

[6] Protege:

- http://www.co-ode.org/downloads/protege-x/
- http://protege.stanford.edu/

[7] PHP online dokumentáció:

- http://www.php.net/

[8] Semantic MediaWiki (SMW):

- http://semantic-mediawiki.org/wiki/Semantic_MediaWiki

Köszönetnyilvánítás

Végül, de nem utolsó sorban szeretném megköszönni témavezetőmnek, Jeszenszky Péter egyetemi adjunktusnak, hogy szakmai tanácsaival, javaslataival segítségemre volt a diplomamunkám elkészítésében.