Production Systems and Information Engineering
Volume 11 (2), pp. 111-118
doi: https://doi.org/10.32968/psaie.2023.2.10

ADATBÁZIS ÉS ONTOLÓGIAI ALAPÚ MODELLEK

AGÁRDI ANITA
Miskolci Egyetem, Informatikai Intézet
agardianita@iit.uni-miskolc.hu

Absztrakt.
Jelen cikk célja az UML, adatbázis és ontológiai alapú modellek bemutatása, a modellek összevetése. Az ontológiák egy bizonyos terület tudásának leírását jelentik. Az ontológiák fő célja egy adott területhez tartozó szemantikai objektumok ábrázolása. Az ontológia során az objektumokat osztályokhoz rendeljük, és leírjuk azokat, a tulajdonságok segítségével. Az Unified Modeling Language (UML) egy egységes modellező nyelv, amely olyan grafikus jelöléseket használ, amelyek a szoftverfejlesztési folyamat számos aspektusát képesek kezelni, az adatbázis tervezéstől a kódmodulok kölcsönhatásáig. Az adatbázisokkal kapcsolatban három jelentősebb modellt említek meg: hierarchikus adatmodell, hálós adatmodell, relációs adatmodell. A cikkben a modellek bemutatása után a konverziós lépéseket, modellek hasonlóságát és különbségét és néhány minta-modellt mutatok be a szakirodalom szerint.

Keywords: UML, ontológia, adatbázis

1. Bevezetés

Az ontológiák egy bizonyos terület tudásának leírását jelentik. Az ontológiák fő célja egy adott területhez tartozó szemantikai objektumok ábrázolása. Az ontológia során az objektumokat osztályokhoz rendeljük, és leírjuk azokat, a tulajdonságok segítségével. Az OWL megkönnyíti az információtartalom szoftveres értelmezését. Ez webes információtartalom feldolgozására lett tervezve. [1]

Az ontológiai tárolás hátrányai [1]:

- jelentős tárolási költséget igényel;
- jelentős válaszidővel jár, mivel nagyszámú asszociációs műveletet kell végrehajtani

Az utóbbi időben a tudásbázis felépítéséhez használt módszerek nagymértékben fejlődtek. Az ontológia alapkifejezések szókincséből és azok pontos meghatározásából áll. Az ontológia azonban több mint szókincs. Nem csak a fogalmak osztályozása, hanem összetett kapcsolatok definiálására is használjuk. Az ontológia meghatározásával tehát létrejön egy közös szókincs azoknak a kutatóknak, akik információkat szeretnének megosztani egymással egy témakörből. A szókincs tartalmazza a terület fogalmának meghatározását és a köztük lévő kapcsolatokat. Jelenleg sok tudományterület szabványosított ontológiákat fejleszt ki. [1]

Az OWL -ben lévő osztály egyedeket tartalmazó halmazként értelmezhető. Az egyedet az osztálytól függetlenül határozzák meg. Az osztályok szuperosztályok -

alosztályok - hierarchiájában szerveződnek. A Thing nevű osztály az OWL -ben az összes egyed halmazát képviseli, tehát minden osztály az OWL: Thing alosztálya. [1]
Az OWL tulajdonságok mindig binárisak, és végeiket domain-nek és range-nek nevezik. Az OWL –ben az alábbi korlátozásokat adhatjuk meg példányok minimális (minCardinality) és maximális (maxCardinality) számát. Ezen kívül egy OWL tulajdonság funkcionális (funkcionális tulajdonság) vagy inverz funkcionális (inverz funkcionális) lehet. . Az OWL kétféle tulajdonságot különböztet meg, úgynevezett objektum tulajdonságokat és adattípus tulajdonságokat. Az objektum tulajdonságuk doman-je és range-je is osztály, míg az adattípus tulajdonságoknak a doman-jük osztály a range-jük pedig egyszerű típus. [1]
Az Unified Modeling Language (UML) egy egységes modellező nyelv, amely olyan grafikus jelöléseket használ, amelyek a szoftverfejlesztési folyamat számos aspektusát képesek kezelni, az adatbázis tervezéstől a kódmodulok kölcsönhatásáig. Az UML modellezési technikák a rendszer entitásokat osztályokban modellezik attribútumokkal és viselkedéssel, a diagramok pedig a modellező elemek ábrázolásának és megtekintésének eszközei. [1]
Az UML az alábbi diagramokat tartalmazza [2]:
Statikus szempont szerint:

- Osztálydiagram: osztályokat és azok tulajdonságait, metódusait tartalmazza
- Komponensdiagramok: komponensek leírását tartalmazza, mint fájl, modul, csomag stb.
- Struktúradiagramok: az osztályok struktúráját mutatja.
- Objektumdiagramok: az osztálydiagramok egy aktuális állapotát mutatják.
- Csomagdiagramok: az elemek csomagokba szerveződését mutatja
- Dinamikus szempont szerint:
- Aktivitásdiagramok: munkafolyamat modellezésére használjuk
- Állapotgép diagramok: lehetséges állapotokat jelentik.
- Használati eset (use case) diagramok: a rendszernek és felhasználóinak kapcsolatát adja meg
- Interakciós diagramok: rendszerelemek közötti kommunikációt adják meg

Az adatbázisokkal kapcsolatban három jelentősebb modellt említek meg: hierarchikus adatmodell, hálós adatmodell, relációs adatmodell. [3]
Hierarchikus adatmodell [3]: az adatokat– egy hierarchiában kell elrendezni. Az adatok között fennálló kapcsolat szülő-gyermek (1:N) kapcsolatnak felel meg. Leginkább a fa struktúrával szemléltethető.
Hálós adatmodell [3]: az adatok közötti kapcsolatokat gráfok írják le. A gráf csomópontok és ezeket összekötő élek együttese. Két csomópont között akkor van kapcsolat ha él köti őket össze. Egy csomópontból tetszőleges számú indulhat ki, így könnyen ábrázolhatók az N:M típusú kapcsolatok.
Relációs adatmodell [3]: a reláció gyakorlatilag egy táblázat, annak összes tartalmával együtt. A reláció sorai a logikailag összetartozó adatokat tartalmazzák. Egy-egy objektumot írnak le a sorok, az oszlopaiban pedig az adatok találhatók. Egy sor és oszlop metszését mezőnek nevezzük. A mező tartalmazza az adatot.

2. Modellek összehasonlítása

Ebben a fejezetben az adatbázis, UML és ontológiai modellek összehasonlítását mutatom be a szakirodalom alapján. Az [4] szerzői az UML és OWL összehasonlítását végezték el, az alábbi táblázatban.

UML		OWL
Osztály (class)		Osztály (class)
Egyed (individual)		Egyed (instance)
Attribútum (attribute)		Funkcionális adattípus tulajdonság (Functional Datatype Property)
Azonosító (Identifier)		Inverz funkcionális tulajdonság (Inverse-Functional Property)
Kapcsolatok (Relationships)	Asszociáció (Association)	Objektum tulajdonság domain és range specifikációval
	Öröklődés (Inheritance)	Hierarchikus kapcsolat két ontológiai fogalom között.

1. táblázat: UML OWL összehasonlítás [4]

Az [5] szerzői szerint az ontológia nyitott világon alapszik, tehát ha egy állítás nem vonható le igaznak vagy hamisnak, akkor azt feltételezzük, hogy ismeretlen. Ezzel szemben az UML modelleket a zárt világ feltételezése szerint értelmezzük. Ez azt jelenti, hogy ha egy állítás nem igaz, akkor hamis. A logika alapú ontológiai nyelvek lehetővé teszik az egyenértékű osztályok meghatározását. Ez azt jelenti, hogy két különböző nevű osztály leírhatja ugyanazokat az egyedet (objektumot). Az UML -ben két különböző osztálynév különböző osztályokat írhat csak le.

Az ontológiai alapú szoftverfejlesztésről az alábbi következtetéseket írták:

- A szoftver forráskódjának megértése nehéz a fejlesztő számára, ha a dokumentáció elavulttá válik vagy nem érhető el.
- A szoftver bevezetése és üzembe helyezése során szerzett ismeretek csak a kódban tükröződnek.
- A szoftverfejlesztők nem ismerik a szemantikus webes ontológiákat.

A tudásábrázolási modellek egységesítése:

- A heterogén forrásokból származó adatok szintaktikai és szemantikai harmonizálása nehéz feladat.

UML-alapú ontológia modellezés:

- Az RDF grafikonok vizualizációi alacsony szintűek, és nem elegendőek az ontológiai építőelemek, például osztályok és tulajdonságok fogalmi szintjének megjelenítéséhez.
- Az UML nem elégíti ki az ontológiai fogalmak ábrázolásának igényeit.
- Az ontológiai nyelvek nem ismertek, és az olvasónak sokszor átláthatatlanok a szöveges definíciók.
- A meglévő modellezési módszerek nem teszik lehetővé az ontológiai egyedek modellezését UML nyelven.
- A szoftverfejlesztők nem ismerik a szemantikus webes ontológiákat.

Ismeretek megszerzése:

- Új ontológia létrehozása nehéz feladat, amely megköveteli az ontológiai nyelv megértését, mely nem könnyű feladat.
- Az UML nyelvben nincs következtető motor, ellentétben az ontológiai nyelvekkel

Modell validálás:

- Az UML modellből hiányzik a jól meghatározott szemantika
- A szoftverfejlesztés kezdeti szakaszában létrehozott UML modellek valószínűleg tervezési hibákat (nem kielégítő koncepciókat, következetlen viselkedési diagramokat) tartalmaznak, amelyek befolyásolják a szoftver minőségét
- Az UML nyelv nem engedélyezi a következtetés (következtető motor hiánya), ellentétben az ontológiai nyelvekkel

Az [1] szerzői az alábbi összehasonlításokat készítették el. Az UML és OWL osztály nagyon hasonlít egymásra, ezek valójában egymásba konvertálhatóak, ahogyan az osztály-alosztály kapcsolatok is. Az egyed az adott osztály egy példánya mindkét modellezés során. Mindkét nyelvben meg lehet adni multiplicitást. Az UML csomag megfelelője maga az ontológia fájl. A különbségeket a 3. táblázat szemlélteti.

UML	OWL	Komment
Osztály (Class)	Osztály (Class)	
Egyed (instance)	Egyed (individual)	Nem kapcsolódó egyedek az OWL osztályban
Attribútum, bináris asszociáció (Attribute, Binary association)	Tulajdonság (Property)	A tulajdonságok OWL-ben globálisak (Property OWL can be globally)
Alosztály (Subclass)	Alosztály (Subclass)	
Enumeration	OneOff	
Multiplicitás (Multiplicity)	minCardinality maxCardinality	OWL-ben a kardinalitást range-el deklarálják
csomag (package)	ontológia	

2. táblázat: Hasonlóságok [1]

UML	OWL
navigable, non-navigable	
származtatott	
absztrakt osztályok	
Osztályok, mint egyedek	
	Thing, globális property-k, autonóm egyedek
	allValuesFrom, someValuesFrom
	SymmetricProperty, TransitiveProperty
	Osztályok, mint egyedek
	disjointWith, complementOf

3. táblázat: Különbségek [1]

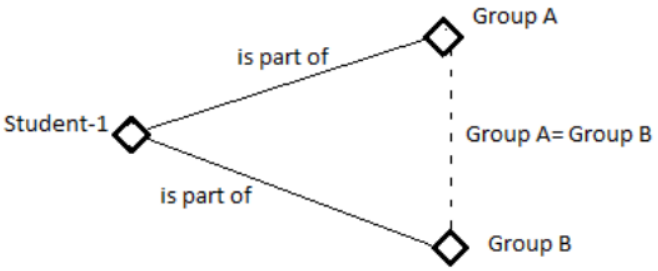
A [6] szerzői a relációs modell és ontológia különbségeket, hasonlóságokat és konverziót tárgyalták. Az alábbi táblázat foglalja össze az eredményeiket.

Relációs modell	OWL komponens
Reláció	Osztály
Idegen kulcs	Két osztály kölcsönösen inverz 1. Az egyik objektum tulajdonság (object property): - Domain: Az oszlopot tartalmazó táblázatnak megfelelő osztály, - Tartomány (range): a hivatkozott táblázatnak megfelelő osztály, 2. A második objektum tulajdonság az első fordítottja, tehát inverze.
Egyszeres és többszörös öröklődés	Öröklődés az ontológiában
Több-több kapcsolat	Két kölcsönösen inverz objektumtulajdonság, amelyek a reláció résztvevő tábláinak megfelelő osztályokra támaszkodnak
Egyszerű oszlop	Adattípus tulajdonság - Domain: Az oszlopot tartalmazó táblázatnak megfelelő osztály, - Tartomány: az oszloptípus XML sémával kifejezve,
Oszlop UNIQUE megkötéssel	Inverz funkcionális tulajdonság - Domain: Az oszlopot tartalmazó táblázatnak megfelelő osztály, - Tartomány: az oszloptípus XML sémával kifejezve,
Oszlop NOT NULL megkötéssel	Adattípus tulajdonság - Domain: Az oszlopot tartalmazó táblázatnak megfelelő osztály, - Tartomány: az oszloptípus XML sémával kifejezve, - Minimális számosság = 1
Primary key	Fordított funkcionális tulajdonság - Domain: Az oszlopot tartalmazó táblázatnak megfelelő osztály, - Tartomány: az oszloptípus XML sémával kifejezve, - Minimális számosság = 1

4. táblázat: Relációs modell és ontológiai modell [6]

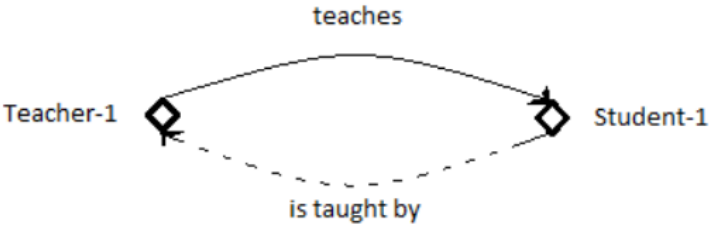
UML – ontológia, relációs modell – ontológai konverziókról számos cikket olvashatunk a szakirodalomban, melyekből néhányat a következőkben be fogok mutatni.
A [4] szerzői egy moodle ontológiát UML diagrammal is ábrázoltak. A model az alábbi elemeket tartalmazza: ‘Calendar’, ‘Group’, ‘LearningResource’, ‘LessonActivity’, ‘User’. Ezen osztályok alosztályokat is tartalmaznak, például a ‘User’ osztály ‘Student’ és ‘Teacher’ osztályokat. Az egyes osztályok tulajdonságokat is tartalmaznak, mint ‘id’, ‘title’, ‘description’ stb, és metódusokat is pl. ‘newOperation()’. Az UML - OWL összehasonlításról is beszámoltak a szerzők. A szerzők konverziót adtak az alábbiakra:

- Funkcionális objektumtulajdonság (Functional object property): Diák és Csoport osztályok közötti asszociációs kapcsolat, értelmezhető, mert egy tanuló csak egy csoport tagja lehet. Ezért ha az A csoport tagja és a B csoport tagja, ez azt jelenti, hogy A csoport = B csoport.



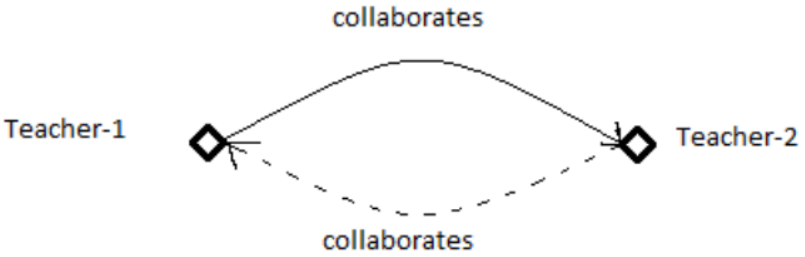
1. ábra: Funkcionális objektumtulajdonság (Functional object property) [4]

- Inverz objektumtulajdonság (Inverse object property): A diák és tanár közötti kapcsolat úgy értelmezhető, mint egyik a másiknak az inverze. Tehát „egy tanár több diákot is taníthat”, vagy „egy diákot több tanár is taníthat”.



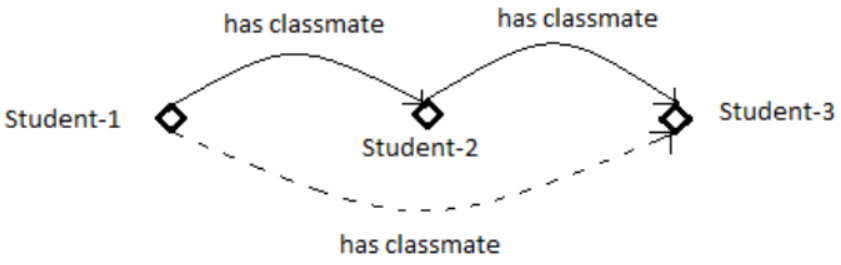
2. ábra: Inverz objektumtulajdonság (Inverse object property) [4]

- Szimmetrikus objektumtulajdonság (Symmetric object property): Például, ha az 1. tanár együttműködik a 2. tanárral, az azt jelenti, hogy a 2. tanár is együttműködik az 1. tanárral.



3. ábra: Szimmetrikus objektumtulajdonság (Symmetric object property) [4]

- Tranzitív objektumtulajdonság (Transitive object property): Például, ha az 1. diáknak osztálytársa a 2. diák és a 2. diáknak a 3. diák osztálytársa, akkor arra a következtetésre juthatunk, hogy az 1. diáknak a 3. diák is osztálytársa.



4. ábra: Tranzitív objektumtulajdonság (Transitive object property) [4]

A [7] szerzői szintén egyetemi oktatást modellezték le. UML-OWL transzformációt hajtottak verge a modellen. Az alábbi osztályokat tartalmazza a

model: ‘Staff’, ‘Course’, ‘Student’, ‘Taking’, ‘Semester’, ‘Instructor’, ‘Administrator’, ‘Professor’, ‘Graduate’. Ezen osztályoknak adattípus propertijük is van, pl. ‘staffNo’, ‘courseNo’ stb.
A [8] szerzői egy cég modelljét készítették el. UML-OWL konverziót is végrehajtottak. A Cég tartalmazza az alábbi osztályokat: ‘Secretary’, ‘Employee’, ‘Department’, ‘Location’, ‘Project’, ‘Manages’, ‘Work_On’, ‘Dependent’, ‘Engineer’, ‘Technical’. Az osztályok tulajdonságokkal is rendelkeznek, mint ‘emp_id’, ‘start_date’ stb.

3. Összefoglaló elemzések

Ezen fejezet értékeli az adatbázis, UML és ontológia kapcsolatait, a konverzió lehetőségét. Az adatbázis és az UML modellek a szoftverfejlesztők számára ismert modellezési nyelvek. Az ontológiákat a tudás ábrázolására használjuk, a következtető motor pedig meglévő információk segítségével új információk következtetését végzi el. Az egyes modellek kapcsán számos különbséget és hasonlóságot is felfedezhetünk, melyek a konverzióban is megmutatkoznak. Az alábbiakban néhány ilyen elemet fogok bemutatni.
Az osztály az OWL-ben, UML-ben és relációs adatbázisban is megegyezik (relációs adatbázisban relációnak hívjuk). Az egyed OWL-ben és UML-ben is egyed, relációs adatbázis megfelelője egy reláció sorát jelenti. Az attribútum UML-ben és OWL-ben is megegyezik, relációs adatbázisban a reláció oszlopának típusát jelenti. Mindhárom modellezés tartalmaz kommenteket. Az UML-beli csomag megfelelője OWL-ben a teljes ontológia, míg relációs adatbázisban a teljes adatbázis.
Hiányosságok közül az alábbiakat emelném ki. Az OWL számos objektum tulajdonságot tartalmaz, melyekkel következtetéseket tud levonni. Ilyen tulajdonságok a funkcionális objektumtulajdonság, inverz objektumtulajdonság, szimmetrikus objektumtulajdonság és a tranzitív objektumtulajdonság. Ezen tulajdonságokat az UML és az adatbázis alapú modellezés nem tartalmazza.
A szerkesztő programok használata is egy fontos szempont lehet. Az ontológiák létrehozását ontológiai szerkesztővel (Protége) érdemes létrehozni. Egy OWL fájl olvashatósága szövegszerkesztővel emberi feldolgozása nehézkes, azonban ontológiai szerkesztővel könnyedén létrehozható. Az UML modell diagramok szintén szerkesztővel hozhatóak létre. Az adatbázis létrehozása, elemekkel feltöltése, módosítása parancsokkal vagy webes felületen történik. Az OWL-hez is tartozik lekérdező nyelv, ez a SPARQL, mely az SQL-hez hasonlít.

4. Összefoglalás

Jelen cikk az UML, ontológia és adatbázis alapú modellezést mutatja be és veti össze a szakirodalom alapján. Az ontológiák egy adott szakterülethez tartozó szemantikai objektumokat ábrázolnak. Az UML egységes, grafikus modellező nyelv, melyet a szoftverfejlesztésben gyakran használnak a szoftverek modellezésére. Az adatbázis modellek közül a relációs adatmodell az egyik leggyakrabban használt adatmodell. A cikkben a modellek bemutatásán kívül az egyes modellek konverzióit is bemutattam és néhány mintamodell konverziót a szakirodalom alapján.

Köszönetnyilvánítás. „Az Innovációs és Technológiai Minisztérium ÚNKP-21-3 kódszámú Új Nemzeti Kiválóság Programjának a Nemzeti Kutatási, Fejlesztési és Innovációs Alapból finanszírozott szakmai támogatásával készült.”

Hivatkozások

[1] Pătraşcu, A. (2015). Comparative analysis between OWL modelling and UML modelling. Petroleum-Gas University of Ploiesti Bulletin, Technical Series, 67(2).

[2] Sike Sándor, Varga László: Szoftvertechnológia és UML, http://people.inf.elte.hu/pagsaai/SEGEDLETEK/Szofttech/progtechUMLkonyv.pdf (Letöltés: 2021. 11. 02.)

[3] http://www.geo.u-szeged.hu/~laci/ab-Geoinfo-tananyag/ch02s03.html (Letöltés: 2021. 11. 02.)

[4] Bouihi, B., & Bahaj, M. (2019). An UML to OWL based approach for extracting Moodle’s Ontology for Social Network Analysis. Procedia computer science, 148, 313-322. https://doi.org/10.1016/j.procs.2019.01.039

[5] Mkhinini, M. M., Labbani-Narsis, O., & Nicolle, C. (2020). Combining UML and ontology: An exploratory survey. Computer Science Review, 35, 100223. https://doi.org/10.1016/j.cosrev.2019.100223

[6] Louhdi, M. R. C., Behja, H., & El Alaoui, S. O. (2013, November). Transformation rules for building owl ontologies from relational databases. In Second International Conference on Advanced Information Technologies and Applications (pp. 271-283). https://doi.org/10.5121/csit.2013.3822

[7] Xu, Z., Ni, Y., He, W., Lin, L., & Yan, Q. (2012). Automatic extraction of OWL ontologies from UML class diagrams: a semantics-preserving approach. World Wide Web, 15(5), 517-545. https://doi.org/10.1007/s11280-011-0147-z

[8] Vo, M. H. L., & Hoang, Q. (2020). Transformation of UML class diagram into OWL Ontology. Journal of Information and Telecommunication, 4(1), 1-16. https://doi.org/10.1080/24751839.2019.1686681