Budapesti Műszaki és Gazdaságtudományi Egyetem
Villamosmérnöki és Informatikai Kar
Méréstechnika és Információs Rendszerek Tanszék

Általános célú szerkesztőfelület parciális modellekhez

Szakdolgozat

Készítette
Deim Péter Pál

Konzulens
Semeráth Oszkár

2016. december 9.

Tartalomjegyzék

HALLGATÓI NYILATKOZAT

Alulírott *Deim Péter Pál*, szigorló hallgató kijelentem, hogy ezt a szakdolgozatot meg nem engedett segítség nélkül, saját magam készítettem, csak a megadott forrásokat (szakirodalom, eszközök stb.) használtam fel. Minden olyan részt, melyet szó szerint, vagy azonos értelemben, de átfogalmazva más forrásból átvettem, egyértelműen, a forrás megadásával megjelöltem.

Hozzájárulok, hogy a jelen munkám alapadatait (szerző(k), cím, angol és magyar nyelvű tartalmi kivonat, készítés éve, konzulens(ek) neve) a BME VIK nyilvánosan hozzáférhető elektronikus formában, a munka teljes szövegét pedig az egyetem belső hálózatán keresztül (vagy autentikált felhasználók számára) közzétegye. Kijelentem, hogy a benyújtott munka és annak elektronikus verziója megegyezik. Dékáni engedéllyel titkosított diplomatervek esetén a dolgozat szövege csak 3 év eltelte után válik hozzáférhetővé.

Budapest, 2016. december 9.

Deim Péter Pál
hallgató

Kivonat

Informatika világában manapság egyre elterjedtebb rendszerek és szoftverek fejlesztésénél, hogy modellező eszközöket használunk. Ezen eszközök hatékonyabbá teszik a tervezési munkálatokat, azonban többségük nem biztosít lehetőséget arra, hogy hiányos, vagy hibás modelleket készítsünk.

A szoftver általában már a tervezés alatt is megkövetelik azt, hogy a modellünk jólformált és teljes legyen. Ez sokszor olyan döntések meghozatalát kényszeríti ki, amivel nem szeretnénk a tervezés azon szakaszában foglalkozni. Ilyen esetben a legtöbb, amit tehetünk, hogy megjegyzéseket írunk fel.

Jó lenne egy olyan eszköz, ami a modellezés absztrakciós szintjén képes kezelni az ilyen döntéseket. Egyre többen foglalkoznak ezzel a kérdéssel, és sokféle megközelítés született már a problémára. Így került előtérbe a parciális modell és a MAVO (May-Arbitrary-Variable-Open word) absztrakció is.

Dolgozatomban egy olyan általános célú modellező környezetet mutatok be, aminek segítségével lehetséges részleges modelleket létrehozni és szerkeszteni. Az eszköz a MAVO absztrakciót támogatja. Ez úgy jelenik meg, hogy annotációkkal lehet felruházni a bizonytalan elemeket a modellben. Illetve egy esetben, az 'Open world' esetében magát a modellt lehet megjelölni. Az annotációk nem csupán jelzés értékűek, hanem funkcionalitás is társul hozzájuk. A modellező eszköz biztosít az annotációk feloldására automatizált megoldást, ezt finomításnak nevezik. Ezáltal csökkenthető a modell részlegessége, ami végül egy olyan modellhez vezethet, amiben egyáltalán nincsen annotáció, tehát teljes.

Végeredményben, tehát a szerkesztővel minőségibb tervezés lehet megvalósítani, jobb dokumentálhatóságot biztosít és a prototípus gyártást is meggyorsítja.

Abstract

In the world of IT it is increasingly popular these days to use modelling tools in the development of systems and softwares. These tools make the design more efficient but most of them are incapable of creating incomplete models.

Sofwares usually determine already at the design phase that the model is appropriately constructed and complete. This often leads to decisions that should not be necessary to deal with at this early stage of the designing process. The best we can do in such a case is to leave comments.

A device that could handle this type of decisions on an abstract level of modelling would come handy. More and more professionals are busy with this problem and there are already several possible solutions from varous perspectives around. This is how the parcial model and the MAVO (May-Arbitrary-Variable-Open word) abstractions have come to the fore.

In my thesis I will demonstrate a general modelling environment with the help of which it is possible to create and construct partial models. The tool supports the MAVO abstraction. This works based on the possibility to assign annotations to uncertain elements in the model. And in the case of the 'Open world' the model itself can be assigned. The annotations are not only signs but they are also functional. The modelling tool provides an automated solution to unlock the annotations, which is called refinement. Thus the partiality of the model can be gradually reduced as far as it is complete and does not contain any annotations.

In the end, it is possible to realize a quality design with the editor, which results in more precise documentation and speeds up the production.

1. fejezet

Bevezetés

1.1. Témamegjelölés

Modellvezérelt tervezésnek nagy szerepe van az informatikában egy szoftver vagy rendszer létrehozásánál. Modellek segítségével sokkal strukturáltabban és megbízhatóbban lehet tervezni. A modellek helyességét automatikusan ellenőrizni és a modellből kódot generálni is lehet.

1.2. Problémafelvetés

Mai modellező eszközökkel általában nem lehetséges, hogy részleges, vagy hibás modelleket kezeljünk, elmentsünk. A modell készítése közben lehetnek döntések, amikkel nem is feltétlen szükséges foglalkozni az adott tervezési szakaszban, mégis ahhoz, hogy a modell érvényes legyen rákényszerül az ember. Ezeket az elemeket célszerű lenne valamilyen módon megjelölni és a szerkesztést folytatni. Előfordulhat hogy a modellben egy elem megléte, vagy a multiplicitása kétséges. Esetleg olyan, hogy nem lehet eldönteni egy elemről hogy az melyik másikkal áll kapcsolatban.

1.3. Célkitűzés

Dolgozatom célja, olyan általános célú modell elkészítése, aminek segítségével lehet részleges, vagy hiányos modelleket is ábrázolni. Például, előfordulhat olyan, hogy nem tudjuk eldönteni egy attribútumról, hogy melyik elemhez tartozik. Ilyet általában a modellező eszközök nem támogatnak. Célszerű lenne egy olyan általános módszert kitalálni, amivel lehetséges az ilyen és ehhez hasonló esetek megjelölése, kezelése.

1.4. Kontribúció

Kutatásom során megismerkedtem a részleges modellezés leglényegesebb aspektusaival. Létrehoztam egy olyan általános célú modellező nyelvet, ami alapján lehetséges részleges modellek készítése. Ezután elkészítettem egy olyan vizuális szerkesztőfelületet, aminek a segítségével parciális modell példányokat lehet készíteni és szerkeszteni.

1.5. Hozzáadott érték

A Szerkesztő nem csupán egy megjelenítő eszközt biztosít a részleges modellnek, hanem más funkciókkal is ellátja azt. A modell finomítására is lehetőséget ad, ami a részleges mo-

dellek szerkesztésének egy fontos funkciója. Ezáltal gyorsabb a prototipizálás és a tervezési döntések dokumentálása.

1.6. Dolgozat felépítése

Előismeretek részben (lásd fejezet 2) először egy konkrét példát mutatok egy modellre. Későbbiekben ezen a példán keresztül mutatom be a részleges modellezés lényegét. A parciális modellezésnek először a szintaktikáját tárgyalom, majd a hozzá tartozó szemantikát ismertetem. Ezekben a részekben a részlegesség négy fajtáját emelem ki: May, Var, Abs, OW. Áttekintés fejezetben (lásd fejezet 3) a konkrét feladat tervét mutatom be egy ábra segítségével. Ismertetem a modell felépítését illetve a szerkesztőfelület lehetőségeit. Ezt követően a megvalósítási folyamat előzményeként kitérek a technológiákra, amikre szükség volt a megvalósításhoz (lásd fejezet 4). A szerkesztőfelület dokumentálása és működésének bemutatását követően végül a továbbfejlesztési lehetőségekről is lesz szó (lásd fejezet 5).

2. fejezet

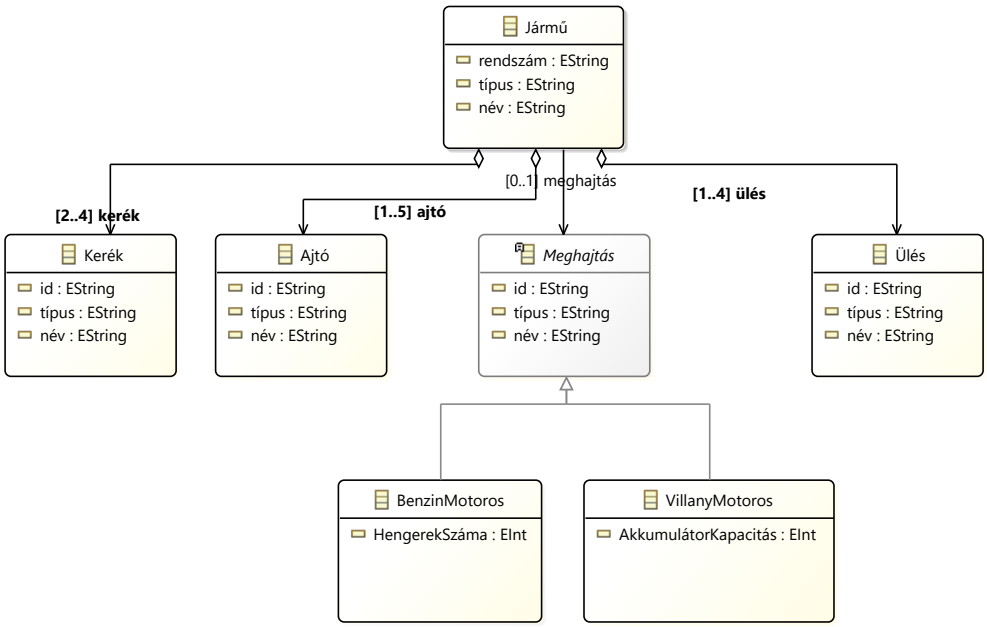
Előismeretek

2.1. Modellezési esettanulmány

A részleges modellezést legegyszerűbben egy gyakorlati példán lehet bemutatni. Vegyük példának az UML [15] osztálydiagramját. Az UML (Unified Modeling Language) egy szabványos modellezési nyelv, ami a tervezést, fejlesztést és egyéb folyamatokat segíti. Ezt a nyelvet informatikában és egyéb üzleti területeken egyaránt alkalmazzák. Magában foglal több diagram típust is.

Az osztálydiagram egy strukturális diagram, segítségével egy rendszerben előforduló objektumokat, azok tulajdonságait és kapcsolatait lehet modellezni. Az osztályok téglalapokkal vannak jelezve, a hozzájuk tartozó attribútumok a téglalapon belül helyezkednek el és az elemek összekötésével a kapcsolatokat lehet jelezni.

Az alábbi példában (lásd Ábra 2.1) egy járműnek az osztálydiagramja látható. A járműnek vannak attribútumai és tartalmaz ajtót, kereket és üléseket, továbbá van neki egy meghajtása, ami egy absztrakt osztály. Ezek ugyan azon attribútumokkal van ellátva: azonosító, típus, név. A meghajtásnak két leszármazottja van a benzinmotoros és a villanymotoros meghajtás. A benzinmotor tulajdonsága a hengerek száma a villanymotor tulajdonsága pedig az akkumulátor kapacitása.



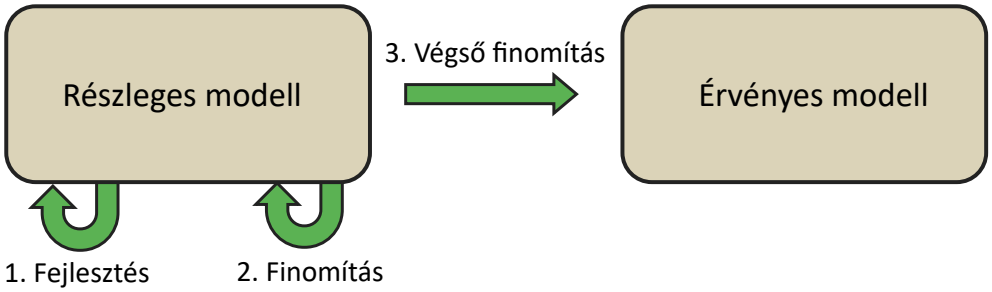
2.1. ábra. Jármű egyszerűsített osztálydiagramja

2.2. Részleges modell

Részleges modell segítségével lehetőség van tervezői döntések dokumentálására. Általában a modellező eszközökkel csak érvényes modelleket lehet készíteni. Így a modell készítője belekényszerül olyan döntések meghozatalába, amiket csak később kéne meghozni, emiatt elveszhetnek tervezői döntések. Részleges modellben lehetőség van ezeket a kétséges helyzeteket már a modellezés szintjén kezelni. Így a modell nem csupán a struktúráról tartalmaz információt, hanem a részlegességről is, tehát hogy teljesen specifikált a modell vagy sem.

Az előbbi példát tekintve, lehetséges az hogy a járműnek egyáltalán nem szeretnénk ajtót, mert egy motort akarunk modellezni. Ekkor feleslegessé válik teljesen az ajtó jelenléte a modellben. Erről információt az UML szabályai szerint nem tudunk tárolni. Vagy feljegyezzük a lehetséges változtatást, vagy pedig létrehozunk egy másik diagramot, amibe van a járművön ajtó és egy olyat, amiben nincs. Ezek egyike sem tűnik jó megoldásnak. Egy ilyen egyszerű diagram esetén még akár átlátható de egy nagyobb, akár több száz elemből álló modell esetén, ha máshol is van ilyen kétség a végleges modellel kapcsolatban, akkor már kezelhetetlenné válik. Például 5 ilyen döntés esetében 2^5 darab diagramot kéne párhuzamosan fejleszteni.

Lehetőség van a modell finomítására is (lásd Ábra 2.2). Finomítás során a parciális modellből kikerülnek bizonytalan elemek. Ez azt jelenti, hogy a modellben jelzett részlegesség mértéke csökkenthető és ennek eredményeképpen véges számú finomítás után a modellből teljesen eltűnnek a részlegességek.



2.2. ábra. Finomítás menete

2.2.1. Szintaktika

Részleges modellek esetén annotációkkal lehet megjelölni a modellt, illetve annak elemeit. Ez úgy valósítható meg, hogy vesszük egy modell gráfreprezentációját, és az abban lévő báziselemekre annotációkat helyezünk el. Modellezés során négy fajta részlegességet jelölhetünk meg. Annotációk a modell minden elemére kerülhetnek, lehet attribútumra, objektumokra, élekre vagy akár magára a modellre is.

- May: modellelemre vonatkozik
- Set: modellelemre vonatkozik
- Var: modellelemre vonatkozik
- OW: modellre vonatkozik

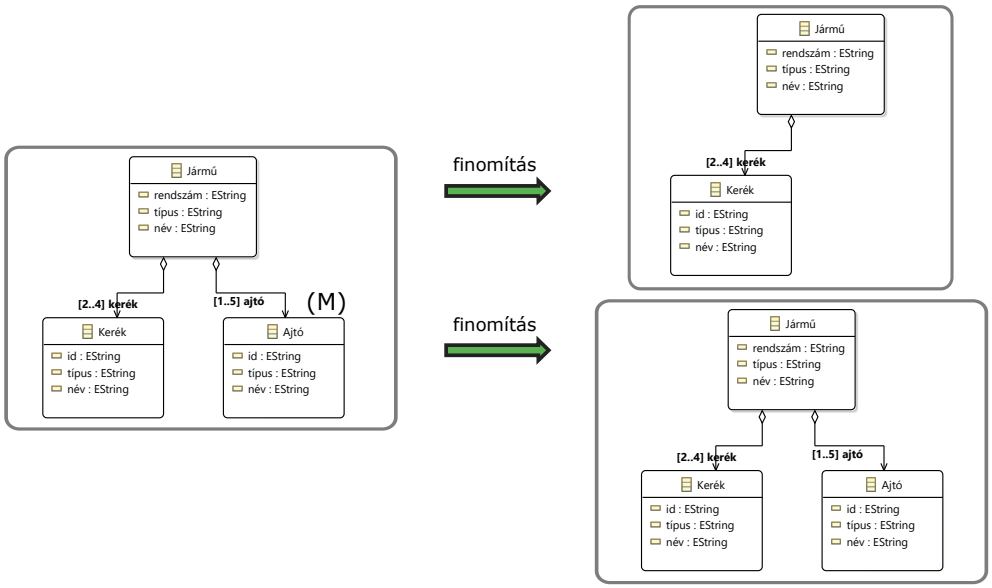
2.2.2. Szemantika

Alábbi fejezetben bemutatom, hogy hogyan jelenhetnek meg parciális modellben a különböző részlegességek.

May

Annotációkkal láthatjuk el a modellt az alapján, hogy egy modellelem biztosan benne lesz a végleges modellben, vagy pedig még bizonytalan a léte. 'M' May exist (lehet, hogy benne lesz a végleges modellben, de lehet, hogy nem), 'E' Must exist (biztosan benne lesz a végleges modellben). A modell finomításával lépésről lépésre egyre kevesebb 'M' lesz. 'M' az vagy átvált 'E'–re, vagy teljesen kikerül a modellből. Akkor tekinthető a modell véglegesnek, ha már nem szerepel benne 'M'-el megjelölt elem. Továbbiakban az 'E' jelöléssel nem foglalkozom, mert amelyik elemen nincs annotáció, az benne van a modellben.

Tegyük fel, hogy nem tudjuk milyen járművet akarunk még modellezni a kezdeti fázisban. Ezért a kiindulási objektummodellben elláttuk May annotációkkal a járműnek az ajtó elemét. Így finomítás során ez az elem később eltűnhet de akár meg is maradhat. Ha nem akarjuk modellezni az ajtók számát a jármű lehet akár egy motor. Ahol két ajtó jelenik meg a modellben ott a jármű lehet egy autó. Erre példa az alábbi diagramrészlet (lásd Ábra 2.3).

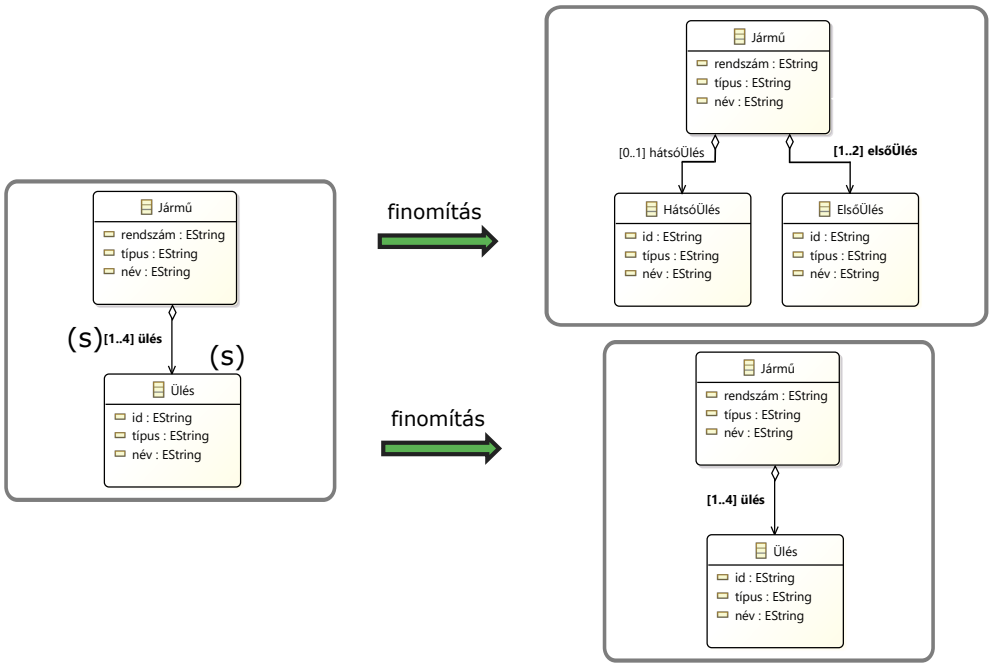


2.3. ábra. May részlegesség feloldása

Abs

Ennek is két különböző annotálási módja lehet: 'P', Particular (egyedi elem) és 'S', azaz Set (egy vagy több elemet jelölhet). Particular az olyan elem, ami már biztosan benne lesz a végleges modellben, azonban a Set olyan elemet jelöl, ami legalább egyszer elő fog fordulni a végleges modellben, de lehetséges hogy többször is. Finomítások során az 'S'-el megjelölt tagokat szétbontjuk több részre vagy meghagyhatjuk egyedi elemként, de a végleges modellben már csak egyedi elemek szerepelhetnek. Egyedi elem esetében az annotációt mellőzhetjük.

Diagram kezdeti fázisában még nincs eldöntve, hogy az ülés az egyedi elem vagy sem, ezért meg van jelölve egy 'S' annotációval. Lehetséges, hogy egy együléses versenyautót szeretnénk építeni, de lehet, hogy egy családi autót, aminek van hátsó ülése is. Finomítás során az ülést megtartjuk eredeti formájában. A másik lehetőség viszont az, hogy szétbontjuk első illetve hátsó ülésre. Ebben az esetben ugyan azok a tulajdonságok lesznek meg mind a két ülésben (pl.: szín, anyaghasználat), de mégis modell szempontjából külön kezelendőek. Erre példa az alábbi diagramrészlet (lásd Ábra 2.4).



2.4. ábra. Abs részlegesség feloldása

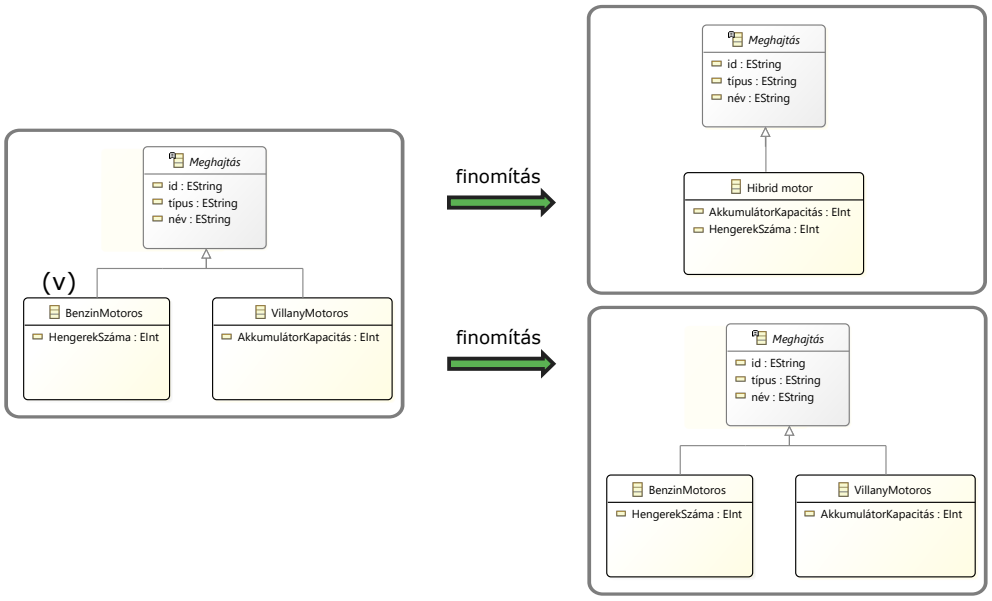
Var

Kétféleképpen lehet annotálni az elemeket: 'C' Constant (konstans elem) és 'V' Variable (változó elem). Bizonyos értelemben az Abs fordítottjának lehet tekinteni, mert nem elemeket többszöröz, hanem összeolvaszt. Felveszünk több elemet a modellbe, ami lehet, hogy későbbiekben összeolvasztunk egy elemmé, tehát fenn áll a lehetősége annak, hogy két elem megegyezik a modellezés első szakaszában. Amikor elkezdünk egy modellt, akkor nem biztos, hogy meg tudjuk mondani az elemekről, hogy azok a későbbiekben azonosak-e vagy különbözőek. A végleges modellben már csak konstans elemek lehetnek. A konstans elemek annotációja ebben a helyzetben is elhagyható, mert ami nincs megjelölve az tekinthető konstansnak.

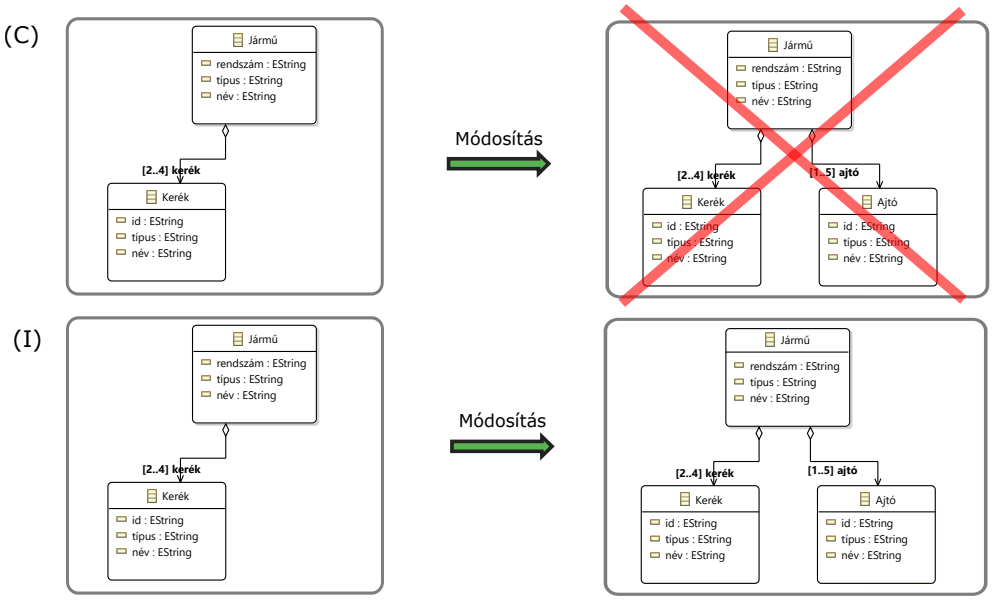
Itt a példában látható (lásd Ábra 2.5), hogy a kezdeti állapotban még két külön meghajtási mód lehetséges: villanymotoros és Benzinmotoros. Ezek meg lettek jelölve 'V' annotációval. A villanymotornak a fontos jellemzője az akkumulátorkapacitás, míg a benzinmotornak a hengerek száma. Finomítás után ez megmaradhat ilyen különálló formában, de akár összeolvaszthatjuk ezt a két motort és a végeredményben egy hibrid motor lesz, ami mind a kettő motor tulajdonságát magában hordozza.

OW

Modellezés folyamán lehet megjelölni azt, hogy a modell már végleges-e vagy sem, tehát adhatunk-e új elemeket a modellhez vagy nem. Ez a részlegesség nem a modell elemeire vonatkozik, hanem a teljes modellről árul el információt. Két annotáció lehet: 'I' Incomplete (befejezetlen) és a 'C' Complete (befejezett). Ha egy modell a 'C' annotációval van megjelölve, akkor az már nem, ha 'I'-vel, akkor még módosítható (lásd Ábra 2.6).



2.5. ábra. Var részlegesség feloldása



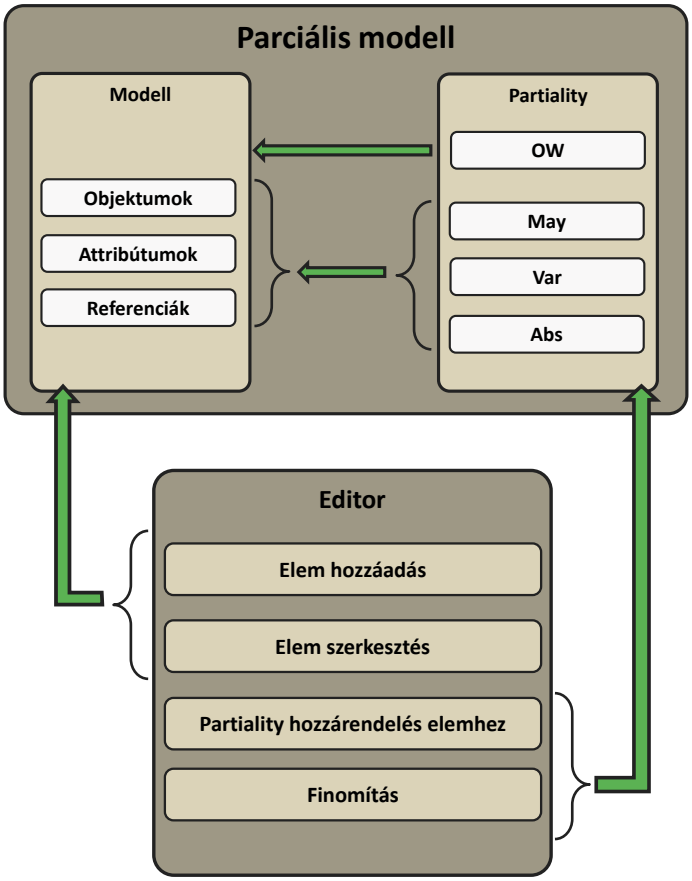
2.6. ábra. OW részlegesség hatása

3. fejezet

Áttekintés

3.1. Parciális modell szerkesztő funkciói

Ahhoz, hogy a szerkesztés generikusan (azaz tetszőleges tervezőeszközhöz) működjön egy általános modell szükséges. Így ez a modell tartalmazni fog objektumokat, attribútumokat és az ezek közti kapcsolatot kifejező referenciákat. Ezen felül minden elemhez lehetséges rendelni *May*, *Var* vagy *Abs* részlegességet. Magához a modellhez pedig *OW* részlegességet lehet rendelni (lásd Ábra 3.1).



3.1. ábra. Részleges modell és a rajta végezhető műveletek

3.2. Fejlesztési lehetőségek

Egy osztálymodellt olyan programmal szerkesztjük, ami fa struktúrában jeleníti meg a modellt. Tehát például az attribútum csak egy másik osztály gyereke lehet, akkor nem létezhet attribútum objektum nélkül. Attribútumot nem lehet több objektumhoz is hozzákapcsolni. Kizárólag érvényes modelleket lehet készíteni benne. Azonban a modell tervezése folyamán nem feltétlenül szükséges, hogy ilyen megkötések legyenek, mert ez csak a végső modellnél fontos. A részleges modell szerkesztésénél nincsenek ilyen feltételek (lásd Ábra 3.1).

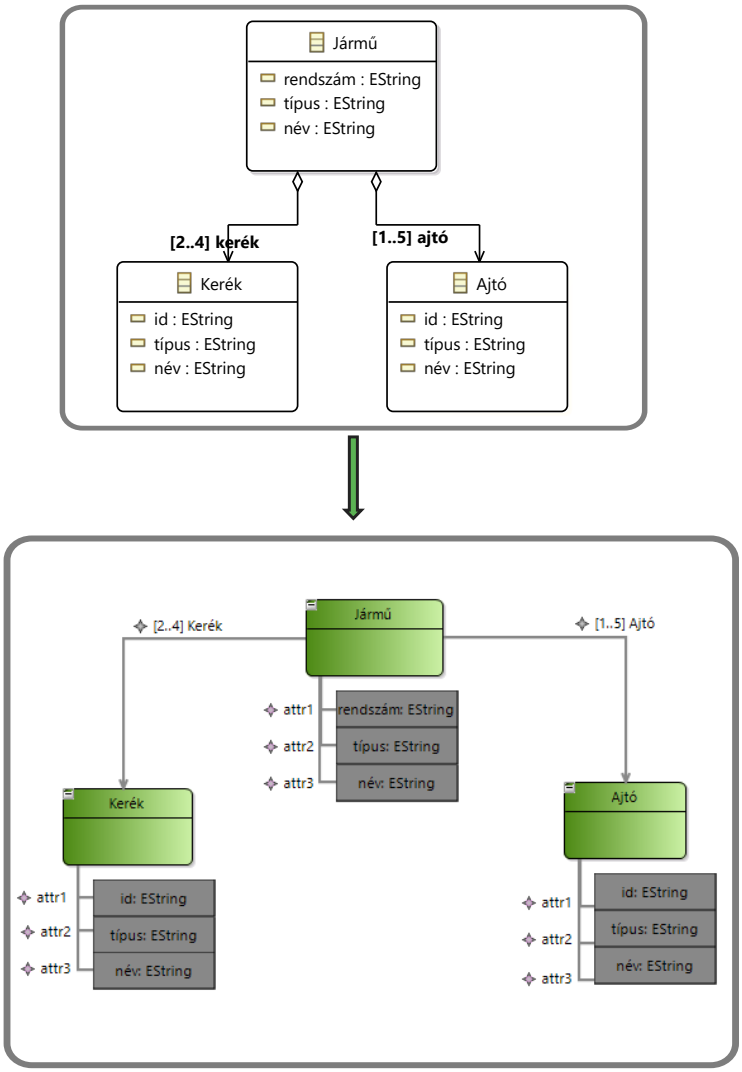
- Az editor képes modellt létrehozni és manipulálni. Lehetőséget ad új objektumok, attribútumok, referenciák létrehozására.
- Elemekhez és a modellhez a már említett részlegességeket hozzá lehet rendelni.
- Lehetséges attribútumot létrehozni objektum nélkül, vagy több objektumhoz is hozzá lehet kapcsolni anélkül, hogy ez konfliktust okozna a szerkesztőnek.
- Parciális modell lehetőséget biztosíthat arra is, hogy példányosítsunk absztrakt osztályokat. Ugyanis egy absztrakt osztály becsomagolható a parciális modell egy objektumába, ezáltal viselkedése megváltozhat.
- Egy objektumdiagram esetében a multiplicitás korlátos lehet, amennyiben az osztálydiagramjában erre vonatkozó kikötések szerepelnek. Ha van minimum vagy maximum megszorítás, akkor ezek nem hághatóak át. Tegyük fel, hogy egy 'A' objektum legfeljebb egy 'B' típusú objektumot tartalmazhat. 'B1' és 'B2' is 'B' típusú. Kezdetben nem tudjuk eldönteni még, hogy melyik objektum lesz a végleges modellben, akkor nincs lehetőségünk mindkettőt létrehozni, hogy majd később dönthessük el a sorsukat, hanem az egyiket mindenképp el kell hagyni. Részleges modell ennél megengedőbb és ha szükséges 'B1' és 'B2' objektum is kapcsolódhat 'A' objektumhoz. Egy parciális modellnek nem kell összefüggőnek lennie és számára az inverz élek betartása sem kötelező.

3.3. Esettanulmány

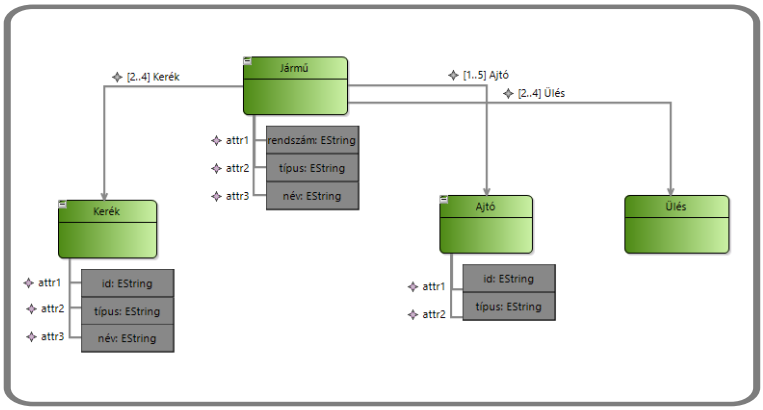
A következő példában, ami egy járműnek a modellje bemutatom, hogy egy parciális modellszerkesztő milyen lehetőségeket biztosít. Tételezzük fel, hogy egy már meglévő modellt 'M' szeretnénk parciális modellként kezelni. Ehhez az első lépés, hogy importáljuk a modellt. Ez azt jelenti, hogy minden egyes elemét (Objektum, attribútum, referencia) megfeleltetjük egy részleges modellbeli elemmel (lásd Ábra 3.2). Az így kapott eredmény egy parciális modell, amit nevezzünk 'PM'-nek.

Következő lépésben módosíthatjuk a modellt. Például adhatunk hozzá új elemet. Járműnek legyen egy ülése, amit összekötünk az autóval. Elemet törölni is lehet. Egyik objektum attribútumát eltávolíthatjuk, legyen ez az ajtónak a 'név' attribútuma. Módosítások után így néz ki a modell Ábra 3.3.

Eddigi módosításokhoz még nem kellett a részleges modell lehetőségeit felhasználni, ezt az eredeti modellben is megtehettük volna, de így látható, hogy a modell készítés lehetőségei nem szűkültek. Most viszont egy attribútumot hozzáadunk a modellhez anélkül, hogy az bármelyik objektumhoz is kapcsolódna. Ez az attribútum legyen a 'szín' nevű, aminek típusa 'EString'. Ezután ezt az attribútumot mivel még nem tudjuk hova is tartozzon dönthetünk úgy hogy több objektumhoz is hozzákapcsoljuk, például kerékhez és az ajtóhoz is. Jelen pillanatban akkor van egy attribútumunk, ami két különböző objektumban is benne van. Ez azonban a végleges modellben biztosan nem maradhat így, mert nem



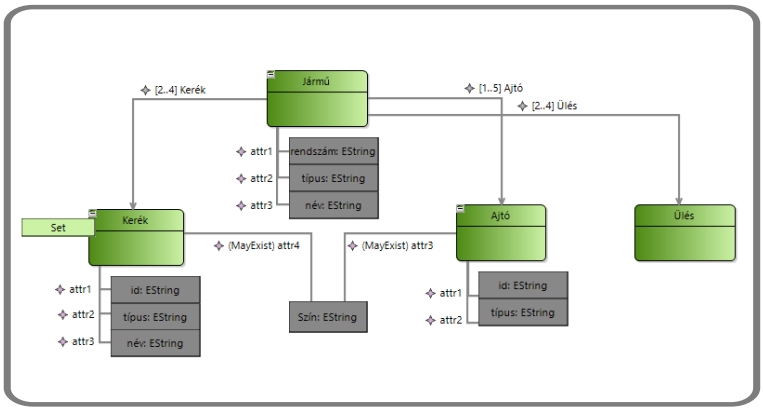
3.2. ábra. Osztálydiagram importálása parciális modellbe



3.3. ábra. Módosított modell

lehetne visszaexportálni 'M' típusú modellé, ugyanis az nem enged meg ilyen esetet. Hogy ezt ne keljen fejben tartanunk meg tudjuk jelölni. Tehetünk az attribútumra egy 'May' annotációt mind a két objektum részéről. A szemléltető ábrán ez úgy jelenik meg, hogy az attribútumot és objektumot összekötő vonalat megjelöljük (MayExist) . Ezután egy

meglévő 'Kerék' objektumot is megjelölünk 'Set' annotációval, ami azt jelenti, hogy több ilyen kerék elem lehet a modellben, de az is lehet hogy marad egyedinek. Így a modellre rátekintve leolvasható, hogy van egy attribútum aminek nem tudjuk a pontos hovatartozását és egy másik, aminek a multiplicitása kérdéses. Megjelölések után a modell így fog kinézni Ábra 3.4. .



3.4. ábra. Részlegesség hozzárendelése a modell elemeihez

A modellt lehetséges finomítani és szükséges is, ha vissza szeretnénk nyerni egy 'M'-el megegyező típusú modellt. Finomítás következtében eltüntethetjük az annotációkat és azokkal együtt a bizonytalanságokat is. Első lépésben finomítsuk az attribútumot. Fixáljuk le, hogy melyik objektumhoz fog tartozni. Így a 'szín' csak az ajtó attribútuma lesz. A modell ekkor még tartalmaz részlegességet, ezért a finomítást folytatjuk. A 'Set' részlegességet is feloldjuk, aminek következtében egy új elem jön létre, a pótkerék. Ez az új objektum megörököl minden tulajdonságot ami a kerékben is megvan.

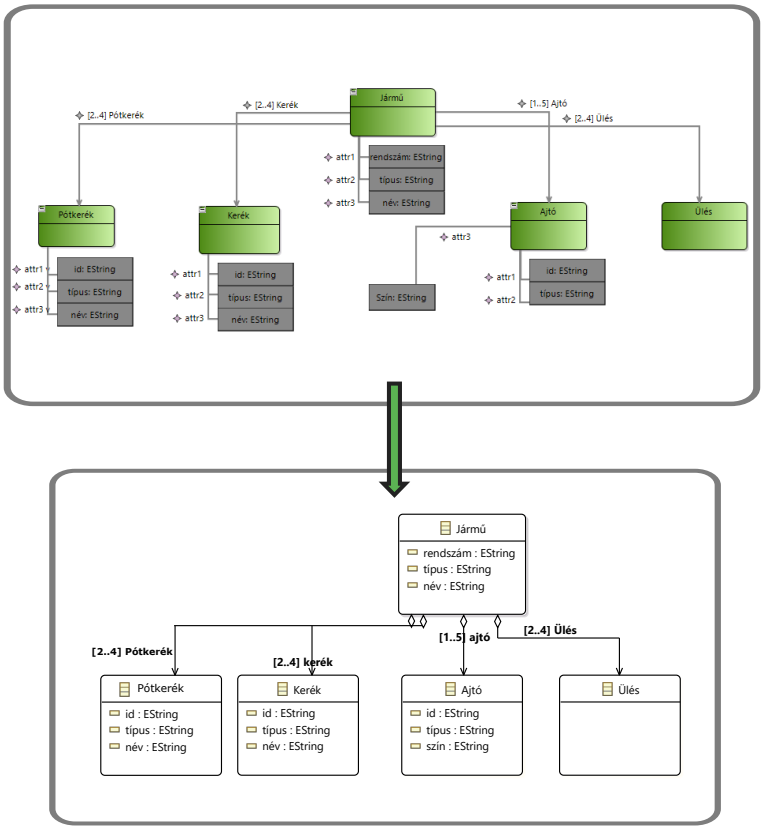
Végül, mivel már a modellünk nem tartalmaz részlegességet, és módosítani se kívánjuk, akár vissza is alakíthatjuk az eredeti 'M' modell típusával megegyező típusú modellé (lásd Ábra 3.5).

3.4. Kapcsolódó munkák

Kutatásom során megnéztem egy másik modellezőeszközt, ami a 'May' részlegesség feloldására nyújt megoldást és ezzel kapcsolatos döntéstámogatást szolgáltat. A többi részlegesség kezelését nem teszi lehetővé ([8]).

Egy másik tanulmányban megismerkedtem egy generikus parciális modell keretrendszerével, ami lehetővé teszi a MAVO absztrakciók kezelését ([11]).

Egy harmadik tanulmányban arra találtam megoldást, hogy miként lehet kibővíteni egy meglévő modellt, hogy képesek legyen részlegesség kezelésére. Ebben a cikkben is MAVO absztrakció alkalmazása volt középpontban ([10]).



3.5. ábra. Finomított modell visszaalakítása (exportálás)

4. fejezet

Megvalósítás

4.1. Felhasznált technológiák

4.1.1. Eclipse

A feladatot Eclipse-be [4] integrált tervezőeszközként valósítottam meg. Ez egy szabadon bővíthető nyílt forráskódú szoftverkeretrendszer. Sokféle modellező eszköz integrálható Eclipse-be. Azért ezt választottam mert a modellezés általam használt részeihez is biztosít megfelelő keretrendszereket.

4.1.2. Metamodell

A metamodell összefoglalja egy modellezési nyelv legfőbb fogalmait, relációit és alapvető struktúráját. Egy olyan alapséma, amire illeszkedik az összes hozzá tartozó alacsonyabb absztrakciós szinttel rendelkező modell. Tegyük fel hogy M* modell M1 modellnek a metamodellje. Akkor M* metamodell meghatározza az M1 modellben lévő elemeket, attribútumokat és ezeknek lehetséges kapcsolatait. Például az alábbi modelleknek (lásd Ábra 4.1) a már korábban említett jármű modell tekinthető metamodelljének. Bal oldalon egy motor példánymodellje, jobb oldalon pedig egy biciklinek a modellje látható. Az általam elkészített metamodell a parciális modellek számára attól általános, hogy nem csupán az osztálydiagram metamodellje lehet, hanem ugyan úgy lehet a jármű metamodellje is.

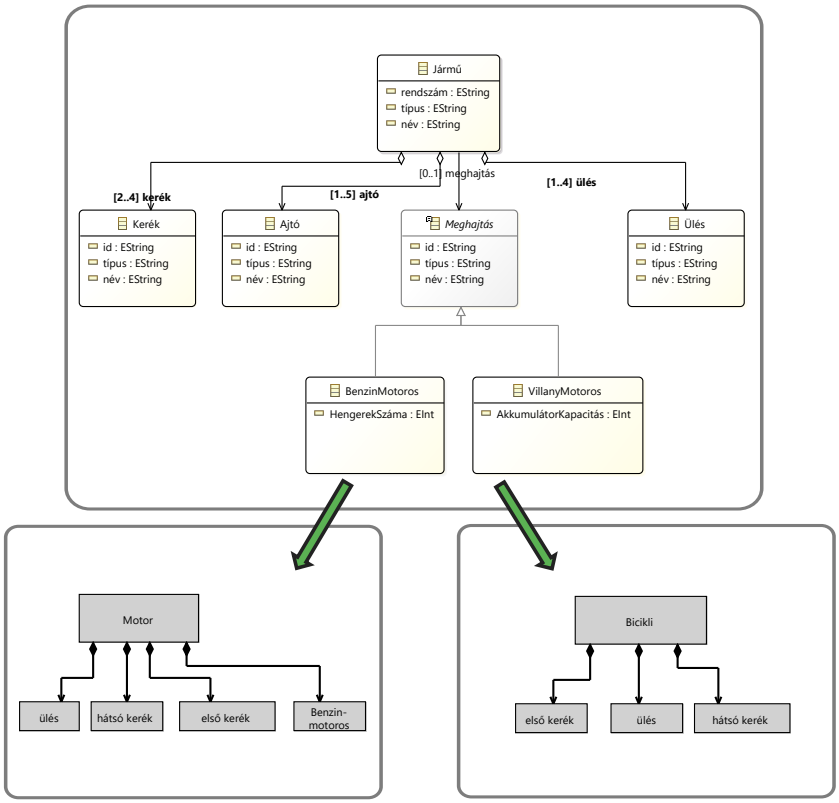
4.1.3. EMF

EMF az Eclipse Modelling Framework [5] rövidítése. Ez egy olyan keretrendszer, mely modellek készítését teszi lehetővé. Az Eclispe-be beépülő EMF modellező eszközök segítségével könnyedén lehet grafikusan metamodelleket alkotni, melyből később konkrét példánymodellt készíthetünk. Továbbá lehetőséget biztosít metamodellből java kódot generálni, ami leképezi a modell elemeit osztályokba és a hozzá tartozó kapcsolatokat és attribútumokat is.

4.1.4. Sirius

Ez egy szintén Eclipse-be épülő keretrendszer [13]. Ez lehetővé teszi hogy EMF modellekhez vizuális megjelenítő és szerkesztő felületet készítsünk. Lehetőség van új objektumok létrehozására, törlésére és szerkesztésére. Az egyes elemek módosítását megszorításokhoz lehet kötni. A szerkesztő elkészítéséhez egy ”viewpoint specification” projektet kell létrehozni, majd ezen belül egy ”odesign” kiterjesztésű fájlt. A fájl definiálja a szerkesztő működését, amiben a következőket műveletek végezhetők:

- Be lehet állítani milyen modellt kezeljen.



4.1. ábra. Jármű metamodellhez tartozó példánymodellek

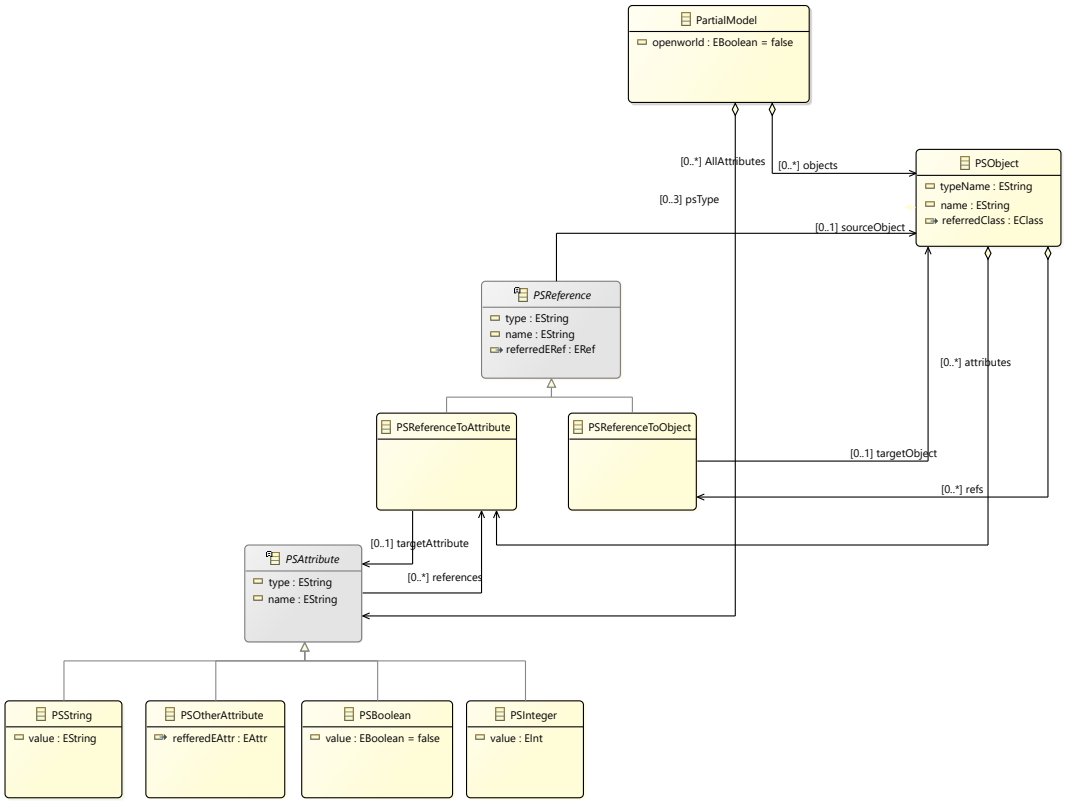
- Reprezentáció típusa is megadható. Például: diagram, táblázat, fastruktúra.
- Modellelemek stílusa, megjelenése. Például: szín, forma.
- Mi történjen egy elem módosításánál, törlésénél, vagy létrehozásánál.
- Lehet külső java kódot futtatni benne.
- Az egyes funkciók működését Aql kifejezésekkel lehet leírni. Aql az Annotation Query Language [3] rövidítése. Lambda [7] kifejezésekhez hasonlóan lehet alkalmazni.

4.2. Általános modell

Egy általános modell elemekből áll, amik tulajdonságokkal rendelkeznek, az elemek között pedig kapcsolatok vannak. Tehát olyan meta modellt kell készíteni, ami mindezeket magába foglalja. A metamodell elkészítéséhez EMF-et használtam (lásd Ábra 4.2).

4.2.1. Objektum (PSObject)

A modellben a PSObject-ek reprezentálják az általános modell elemeit. Rendelkezik névvel (name), ami egy 'EString' és referált osztállyal (ReferredEClass) aminek típusa 'EClass'. a referált osztály mutatja meg, hogy az objektum milyen típusú. PSObjectek tartalmaznak tetszőleges mennyiségű referenciát másik objektumokra (PSReferenceToObject) és attribútumokra (PSReferenceToAttribute).



4.2. ábra. Általános modell, ami legtöbb modell metamodelljének tekinthető

4.2.2. Referencia (PSReference)

Az elemek közötti kapcsolatokat reprezentálják. Fontos, hogy ezek is külön elemként jelenjenek meg a modellben, hisz ehhez is szeretnénk majd részlegességet társítani annotációk formájában. A PSReference egy absztrakt objektum két leszármazottja van a PSReferenceToAttribute és PSReferenceToObject. Erre azért volt szükség mert logikailag kifejezi azt, hogy az attribútum valamelyik objektum része és csak azzal együtt érvényes, míg egy objektum önmagában is értelmezhető. Rendelkezik névvel (name) és egy 'ERefference' típusú attribútummal (ReferredERef). 'ReferredERef' segítségével adható meg a referencia típusa. Van neki forrásobjektuma (sourceObject) és célobjektuma (targetObject) vagy célattribútuma (targetAttribute).

4.2.3. Attribútum (PSAttribute)

Az objektumok tulajdonságai külön egy PSAttribute nevű elemben vannak hozzárendelve az objektumokhoz, így lehetőség van az attribútumokhoz is részlegességet rendelni. Lehetséges továbbá az is hogy egy attribútum több objektumhoz is tartozzon, ugyanis az attribútumokat nem közvetlenül a PSObjeckt-ek tárolják hanem csak hivatkoznak rájuk. Azonban az attribútumok tárolják a rájuk mutató PSReference-ek referenciáit. PSAttribute absztrakt ezért több fajtája lehet.

- PSString: 'String' típusú attribútumot reprezentál. Értéke 'String' típusú lehet csak.
- PSBoolean: 'Boolean' típusú attribútumot reprezentál. Értéke kizárólag 'Boolean' lehet.
- PSInteger: 'Integer' típusú attribútumot reprezentál. Értéke csak 'Int' típusú lehet.

- PSOtherAttribute : Tartalmaz egy 'EAttribute'-ra mutató referenciát. (refferedEAttribute). Tetszőleges egyéb attribútumot referálhat.

4.3. Kiegészítés részleges modellé

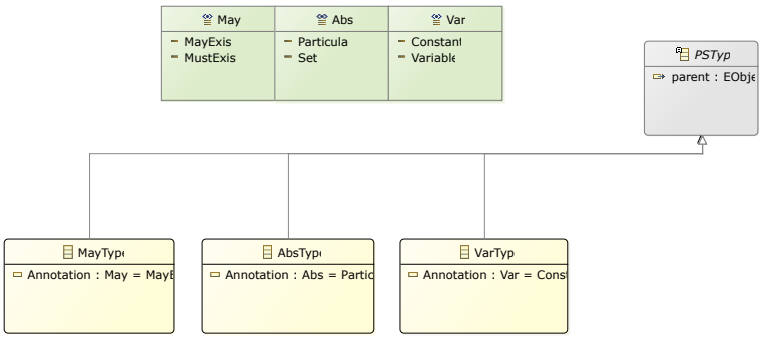
4.3.1. OW részlegesség

Az *OW* részlegességet a modell gyökerében egy boolean változóval tudjuk szemléltetni.

4.3.2. Részlegesség típusa (PSType)

Minden egyes modellbeli elemhez tudnunk kell társítani annotációkat. Ezt a PSType-al tehetjük meg (lásd Ábra 4.3). Ilyen PSType elemet tartalmazhat az attribútum a referencia és az objektum is. Egy időben több fajta részlegesség is lehet egy elemen. Részlegességek szerint három leszármazottja van a PSType-nak:

- MayType: *May* részlegességet jelöl, azon az elemen amelyik tartalmazza.
- AbsType: *Abs* részlegességet jelöl, azon az elemen amelyik tartalmazza.
- VarType: *Var* részlegességet jelöl, azon az elemen amelyik tartalmazza.



4.3. ábra. Általános modell kiegészítve részleges modellekre jellemző tulajdonságokkal

4.4. Szerkesztőfelülete

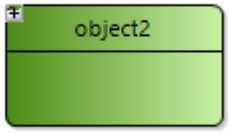
4.4.1. Modell elemeinek stílusa és alapfunkciók

Objektum (PSObject)

Világoszöld téglalapként jelenik meg (lásd Ábra 4.4). A téglalap fejrészében megjelenik az objektum neve.

Funkciók:

- Bal felső sarkában pedig egy kis '+' jel segítségével megjeleníthetjük a hozzá tartozó attribútumokat. Amennyiben ezek láthatóak akkor egy '-' jel jelenik meg amivel elrejthetjük.
- PSObject létrehozása az eszköztárból kiválasztva lehetséges, majd a szerkesztőfelületre kattintva megjelenik egy négyzet egy alapértelmezett névvel. A név formátuma:



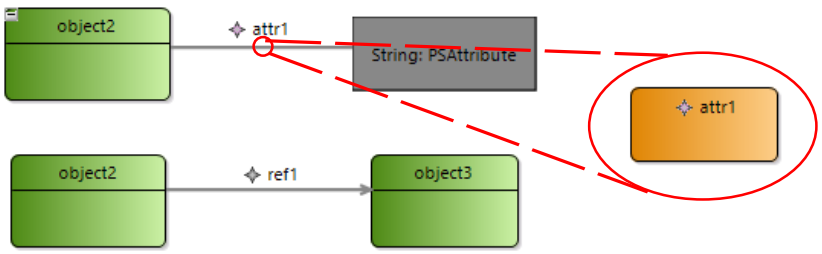
4.4. ábra. PSObject

'Object{szám}'. A szám helyén egy automatikusan generált szám kerül, a már meglévő PSObject-ek száma növelve eggyel.

- Az objektum nevére rákattintva az közvetlenül szerkeszthetővé válik. A többi tulajdonság a Sirius által alapból biztosított 'Properties' fülön érhető el.
- Törlés esetén megsemmisül az objektumból kivezető összes referencia. Sirius tartalmazás esetén automatikusan kitörli az objektumban lévő egyéb elemeket. Az objektumra mutató referenciák viszont nem törlődnek maguktól, mert a forrás PSObject-ben vannak. Mivel a PSObject nem tárolja a rá mutató éleket így először az összes PSObject referenciáján végigiterálva kitörli azokat, amik rá mutatnak, majd törli önmagát is.

Referencia (PSReference)

A felületen szürke vonalként jelenik meg. Összeköthet PSObject-et attribútummal vagy egy másik PSObject-el. Utóbbi esetben a vonal végén nyíl is van. A referencián szövegesen megjelenik neve és zárójelben a hozzá tartozó részlegességek is. Amennyiben rákattintunk a referencia vonalára kétszer, akkor egy új ablak jelenik meg. Ebben az ablakban egy narancssárga négyzeten a referencia neve van. Ez azért fontos mert itt tudunk hozzáadni részlegességeket a referenciához a későbbiekben (lásd Ábra 4.5).



4.5. ábra. PSReference (baloldalt a részleges modellen megjelenő formában vonalként, jobb oldalon a rákattintás utáni nézet látható)

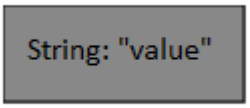
Funkciók:

- Kétszer rákattintva egy referenciára megjelenik egy új szerkesztőfelület. Itt csak egy narancssárga téglalap van a referencia nevével. Erre a nézetre azért van szükség, mert részlegességek így könnyebben kezelhetők. A részlegességek magyarázatánál bővebben lesz erről szó.

- Létrehozása a szerkesztőfelület oldalán lévő sávból kiválasztva lehetséges (reference). Aztán egy PSObject-re kattintva a kezdőpontja, majd második kattintásra a cél objektuma vagy attribútuma jelölhető ki a referenciának. A kiválasztó felületen nincs külön referencia objektumra és attribútumra, ez a felhasználó számára rejtve marad, azonban a háttérben mégis külön kezelődik. Referencia létrehozásának a logikájában van egy elágazás (switch), ami vizsgálja, hogy a célobjektum, tehát amire másodjára kattintunk az milyen típusú. A referencia típusát aql kifejezéssel lehet eldönteni. A referencia nevét hasonlóan a PSObject-hez aql segítségével generálja: 'attr{szám}' vagy 'ref{szám}'. Itt a 'szám' a forrás objektumból kiinduló attribútumokra vagy objektumokra mutató referenciák számát veszi figyelembe. Attribútumra mutató referencia esetében kicsit különbözik a működés. Amennyiben a referált attribútumra már mutat másik referencia, akkor az új referenciához automatikusan hozzá generálunk egy May részlegességet. Ennek az az értelme, hogy gyakorlatban nem fordulhat elő olyan, hogy egy attribútum több objektumhoz is tartozik.
- Éleket át lehet huzalozni, azaz másik célelemet lehet választani nekik. PSObject-re mutató élt nem lehet megváltoztatni PSAttribute-ra mutató élre és fordítva se. Új cél kiválasztása esetén a régi felülíródik. Attribútum esetében a régi attribútumból törlődik az él referenciája és az új attribútumba íródik be. Ha attribútumról olyan attribútumra húzzuk át az élet, amibe már vezet él akkor ahhoz automatikusan hozzá generálódik egy 'May' részlegesség, mint ahogy az új él létrehozásnál is történt.
- Törlés esetén az objektum és a hozzá tartozó referenciák törlődnek.

Attribútum (PSAttribute)

Szürke téglalapként jelenik meg a szerkesztőfelületen (lásd Ábra 4.6). Az attribútum típusa és értéke a téglalap közepén látható kettősponttal elválasztva.



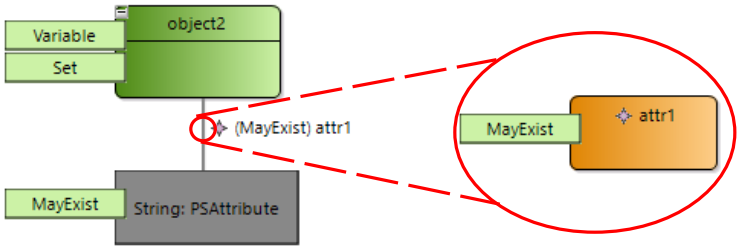
4.6. ábra. PSAttribute

Funkciók:

- Létrehozása az eszköztárról kiválasztva lehetséges, ezután a szerkesztő panelen kattintva létrejön egy attribútum. Ez nem tartozik még egyetlen objektumhoz sem.
- A típusának és értékének szerkesztése lehetséges a Sirius által biztosított 'Properties' fülön.
- Törlésénél végig iterál az összes rá mutató referencián és kitörli azokat, majd törli önmagát is.

Részlegesség típusa (PSType)

A szerkesztőfelületen világoszöld téglalapként jelenik meg az elem mellett, amihez hozzá lett rendelve (lásd Ábra 4.7). Referencia esetén zárójelben látszik az él neve előtt, de ha kétszer rákattintunk a referenciára, hogy megnyissuk a szerkesztőfelületét, ott az élet



4.7. ábra. PSType (baloldalt a részleges modell szerkesztőfelülete, jobb oldalon a referencia szerkesztőfelülete)

reprezentáló narancssárga téglalap mellett fog megjelenni (lásd Ábra 4.4).

Általános funkciók:

- Létrehozása a szerkesztőfelület oldalán lévő sávból kiválasztva lehetséges (May, Abs, Var), utána arra az elemre kattintva amelyikhez a részlegességet hozzá akarom rendelni létrejön a PSType. Él esetében ugyan ez a referencia saját szerkesztőfelületén lehetséges.
- Törlés esetén eltávolítja magát az objektumból.

4.4.2. Finomítás funkció

Duplán kattintva a PSType-ra történik meg a feloldása a részlegességnek. A PSType-nak három fajtája van, amik különböző módokon viselkednek attól függően, hogy milyen típusú elemre vannak rátéve, ezért ezeket külön tárgyalom.

PSObject-May részlegesség feloldása

Feloldásnál az objektum kitörli a belőle kivezető éleket, majd az összes más objektumból rá mutató élet. Kitörli az attribútumra mutató éleket is és amennyiben az adott attribútumra nem vezet más objektumból él akkor az is törlődik (lásd Ábra 4.8). Ezután maga az objektum is törlődik. A feloldást egyszerűen az annotáció törlésével is meg lehet oldani.

PSAttribute-May részlegesség feloldása

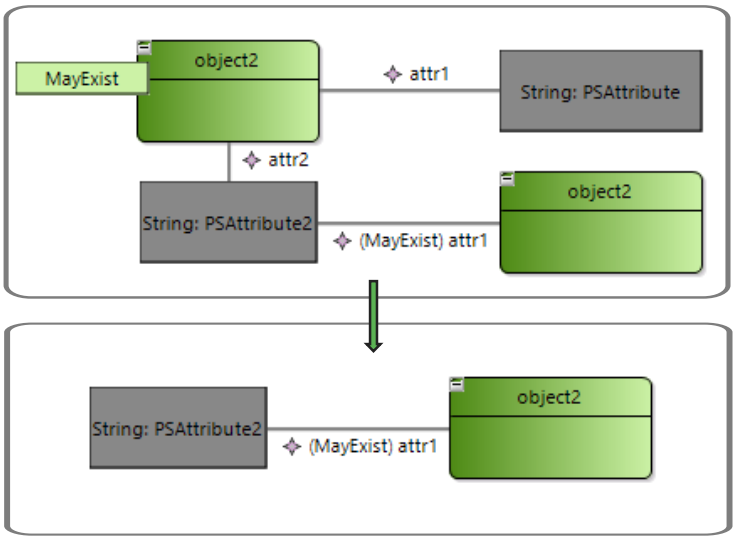
Élek nem törlődnek automatikusan mert a PSObject tartalmazza őket. Végig kell iterálni az összes attribútumra mutató élen és kitörölni azokat. Ezután kitörlődik az attribútum is (lásd Ábra 4.9).

PSReferenceToObject-May részlegesség feloldása

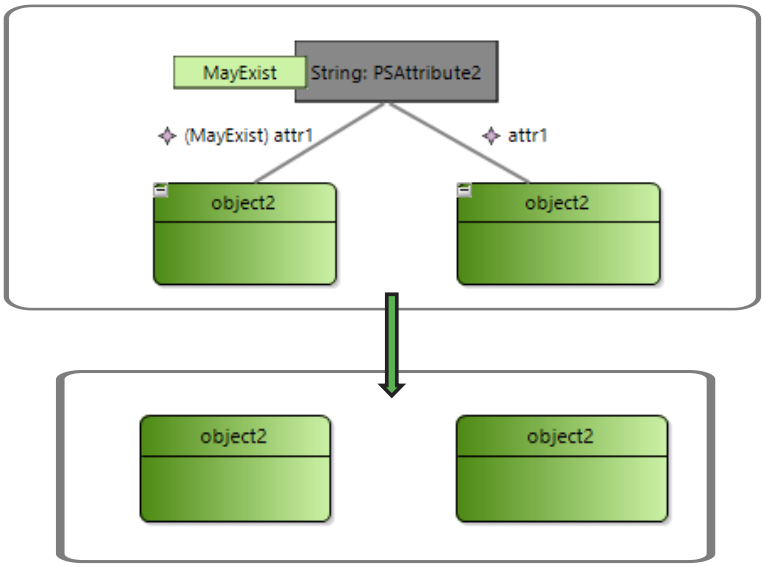
Kitörli a referenciát a forrásobjektumból.

PSReferenceToAttribute-May részlegesség feloldása

Az él törlődik, de eltérően az PSObject-ekre mutató élektől, amennyiben a célattribútumba nem vezet másik él, akkor az attribútum is megszűnik. A finomítás lényege, hogy általa csökken a részlegessége a modellnek. Azért nem lenne értelme az üresen maradt attribútumot meghagyni, ami nem kötődik semmilyen objektumhoz, mert nem csökkent volna a modell részlegessége (lásd Ábra 4.10).



4.8. ábra. PSObject finomítása (May)



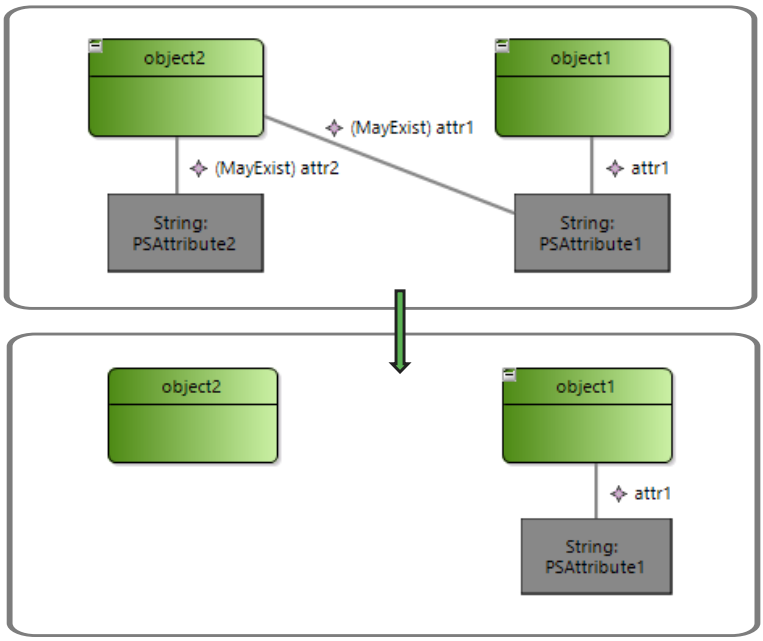
4.9. ábra. PSAttribute finomítása (May)

PSObject-Set részlegesség feloldása

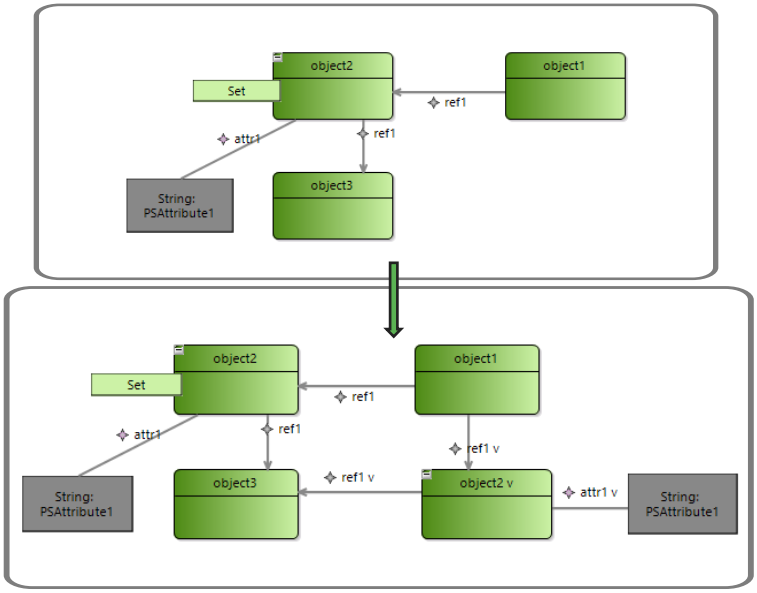
Létrejön egy új objektum, aminek ugyanazok a tulajdonságai, mint az eredeti objektumnak, amin a 'Set' annotáció volt. Az új PSObject referenciái ugyan azokra az objektumokra mutatnak, mint az eredeti PSObject esetében. Új attribútumok jönnek létre az eredeti objektumhoz tartozó attribútumok mintájára és ezek lesznek összekötve az új PSObject-el. Az új élek nevei ugyan azok mint az eredeti esetben, de a végére kerül egy kis 'v' betű (lásd Ábra 4.11).

PSAttribute-Set részlegesség feloldása

Létrejön egy új attribútum és ugyan ahhoz az PSObject-hez fog tartozni mint az eredeti, tehát az objektumból él fog vezetni az attribútumba. A típusát és értékét is megörökli az eredeti attribútumnak. Új elemhez tartozó referenciák nevei után egy kis 'v' betű kerül (lásd Ábra 4.12).



4.10. ábra. PSReferenceToAttribute finomítása (May)



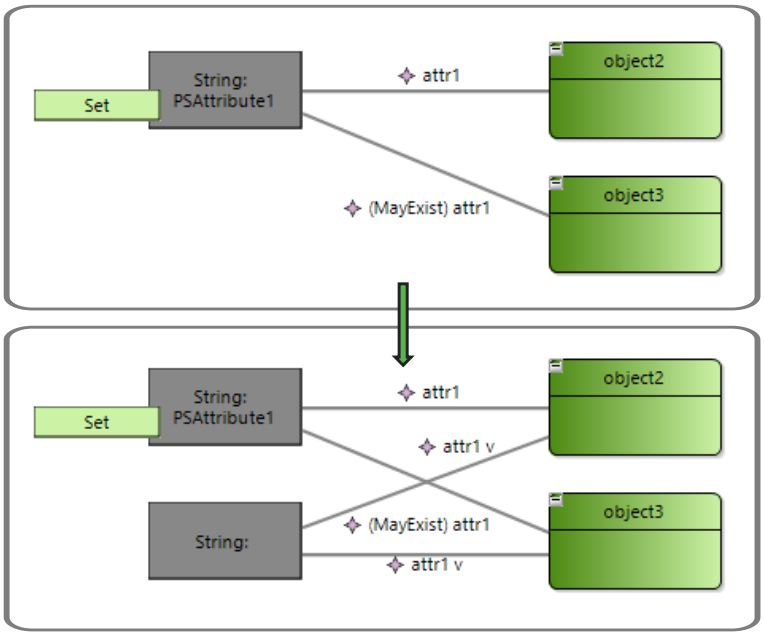
4.11. ábra. PSObject finomítása (Set)

PSReferenceToObject-Set részlegesség feloldása

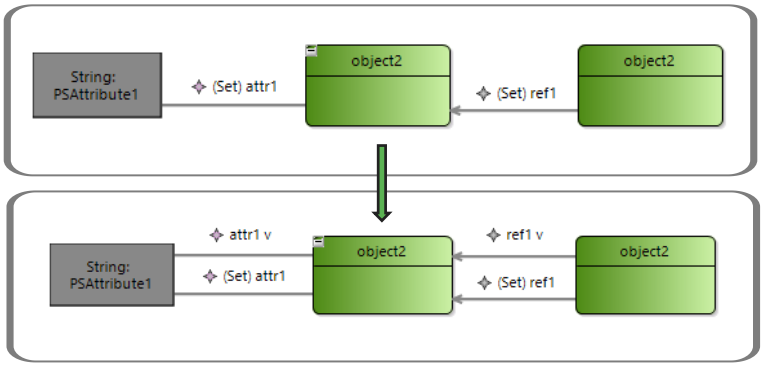
Az eredeti referencia forrásobjektumában létrehoz egy újat, ami ugyan arra az objektumra mutat (lásd Ábra 4.13).

PSReferenceToAttribute-Set részlegesség feloldása

Az eredeti referencia forrásobjektumában létrehoz egy újat, ami ugyan arra az attribútumra mutat, ezután a célattribútum referenciái közé is hozzáadjuk az új élet (lásd Ábra 4.13).



4.12. ábra. PSAttribute finomítása (Set)



4.13. ábra. PSReference finomítása (Set)

Var részlegességről általában

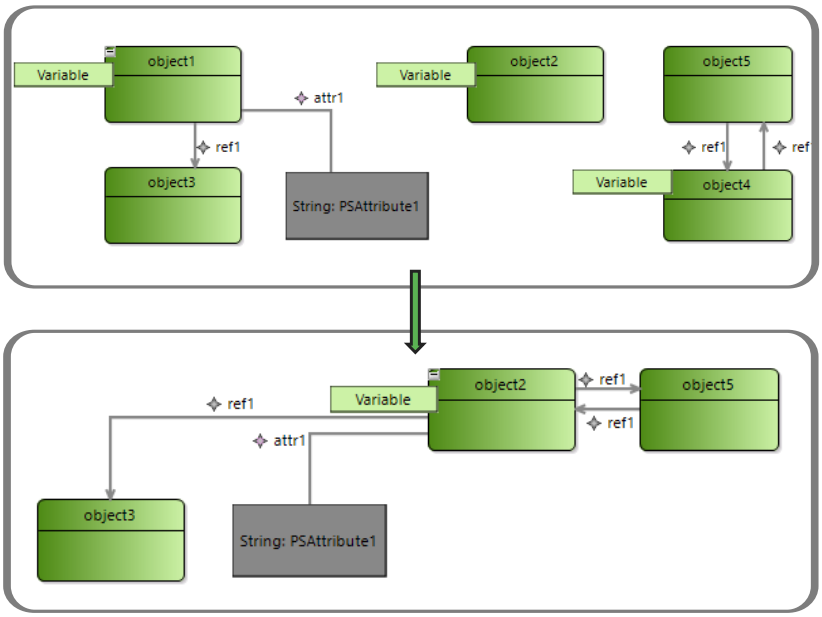
Eredeti objektumnak, elemnek nevezem a továbbiakban azt a PSObject-et amelyiknél a 'Var' részlegességére duplán kattintottunk. A 'Var' részlegességnek van egy id-ja, ami segítségével meg lehet jelölni, hogy melyik 'Var' részlegességek tartoznak egybe, erre a későbbiekben 'Var id'-val hivatkozom.

PSObject-Var részlegesség feloldása

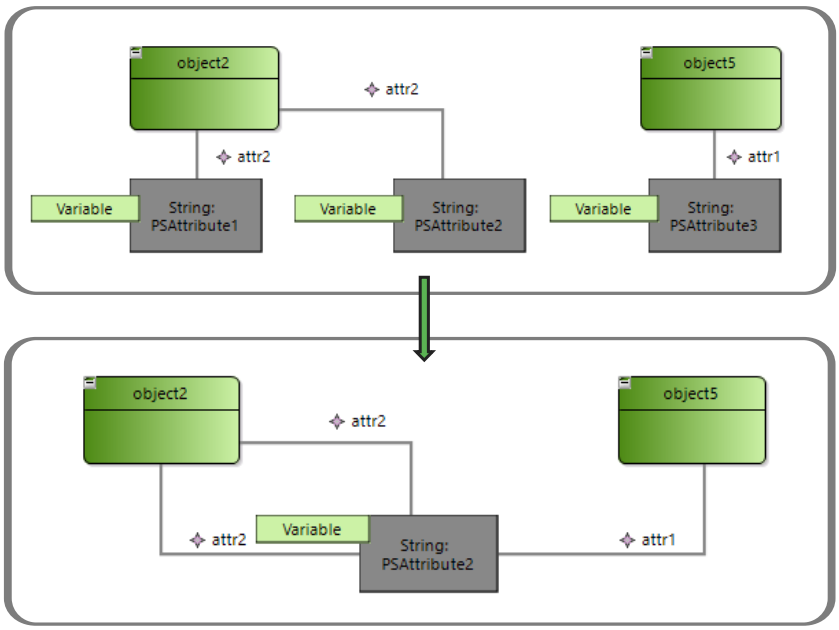
Az összes olyan objektumnak, amelyiknek megegyezik a 'Var id'-ja az törlődik, kivéve az eredeti de előtte az összes hozzá tartozó referencia mintájára létrejönnek az eredeti objektumban élek. Azok az objektumok élei, amik eddig egy törölt objektumba mutattak, azok most az eredeti objektumra fognak mutatni (lásd Ábra 4.14).

PSAttribute-Var részlegesség feloldása

Az eredetin kívül az összes olyan attribútum kitörlődik, aminek a 'Var id'-ja megegyezik az eredeti elem 'Var id'-jával. A törölt elemekbe futó élek az eredeti elemre fognak ezentúl mutatni (lásd Ábra 4.15).



4.14. ábra. PSObject finomítása (Var)



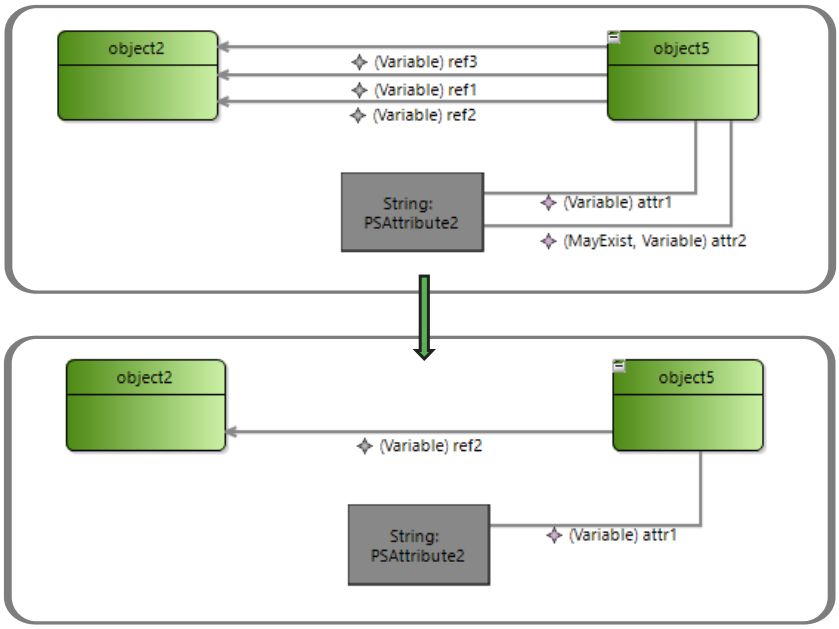
4.15. ábra. PSAttribute finomítása (Var)

PSReferenceToObject-Var részlegesség feloldása

Csak az eredeti referencia forrásobjektumában keres vele megegyező 'Var id'-val rendelkező referenciát. Ezeket a referenciákat törli (lásd Ábra 4.16).

PSReferenceToAttribute-Var részlegesség feloldása

Hasonlóan a PSReferenceToObject esetéhez csak az eredeti referencia forrásobjektumában keres vele megegyező 'Var id'-val rendelkező referenciát. Ezeket a referenciákat törli (lásd Ábra 4.16).



4.16. ábra. OSReference finomítása (Var)

5. fejezet

Összefoglalás és továbbfejlesztési lehetőségek

Dolgozatom során elkészítettem egy általános célú modellezési nyelvet, melynek segítségével parciális modelleket lehet készíteni. Ehhez tartozik egy szerkesztőfelület, ami támogatja ezen modellek módosítását, fejlesztését. Lehet benne objektumokat, attribútumokat és referenciákat létrehozni. Objektumok tartalmazhatnak attribútumokat és referenciák segítségével kapcsolódhatnak más objektumokhoz. Attribútumok függetlenül is létezhetnek az objektumtól. A felsorolt három elem mindegyikéhez lehetséges részlegességeket rendelni: 'May', 'Var', 'Abs'. Magát a modellt pedig meg lehet jelölni 'OW' részlegességgel. A modellt finomítani lehet, mely során csökkenthető annak részlegessége. Ezen részlegességek feloldása a részlegességek szemantikájától függően történik.

A részleges modellek vizsgálatát rengeteg más szemszögből is meg lehet közelíteni. Lehetséges a szerkesztőt továbbfejleszteni, új funkciókkal és validációkkal bővíteni. Jelenleg a modellező eszközben tudunk készíteni parciális modelleket, de már meglévő modellek szerkesztése részleges modellként nem lehetséges. Erre lehetne készíteni egy programot, ami szerkeszteni kívánt modell minden elemét megfelelteti egy parciális modellbeli elemmel. Így például a dolgozatban szereplő jármű (lásd Ábra 2.1) minden objektumát "becsomagolhatjuk" agy részleges modellhez tartozó objektumba. Ezután ez az elkészített szerkesztőben módosítható. Amennyiben nem kívánunk változtatni a modell egy hasonló programmal visszaalakítható lehet. Ezen a szálon tovább haladva lehetne részlegességet tartalmazó modellből is generálni az importálthoz hasonló modellt. A generáló program többféle modellt készíthet a parciális modellből, figyelembe véve, hogy milyen részlegességeket tartalmaz.

Irodalomjegyzék

[1] Acceleo. https://wiki.eclipse.org/Acceleo/Acceleo_Operations_Reference.

[2] András Szabolcs Nagy: Sirius tutorial. https://github.com/FTSRG/lecture-notes/wiki/2016_sirius.

[3] Aql reference. http://www.ibm.com/support/knowledgecenter/SSPT3X_3.0.0/com.ibm.swg.im.infosphere.biginsights.aqlref.doc/doc/aql-overview.html.

[4] Eclipse. https://eclipse.org/.

[5] Emf (eclipse modelling framework). http://www.eclipse.org/emf.

[6] Emf (eclipse modelling framework) tutorial. http://eclipsesource.com/blogs/tutorials/emf-tutorial.

[7] Lambda expression. https://www.techopedia.com/definition/3826/lambda-expression.

[8] Alessio Di Sandro Rick Salay Michalis Famelis, Naama Ben-David – Marsha Chechik: *MU-MMINT: an IDE for Model Uncertainty*. University of Toronto, 2015. IEEE/ACM 37th IEEE International Conference on Software Engineering.

[9] Ocl. https://wiki.eclipse.org/Acceleo/OCL_Operations_Reference.

[10] Michalis Famelis Salay, Rick – Marsha Chechik: Language independent refinement using partial modeling. 2012., 224–239. p. Fundamental Approaches to Software Engineering.

[11] Rick Salay – Marsha Chechik: A generalized formal framework for partial modeling. 2015., 133–148. p.

[12] Sirius advanced tutorial. https://wiki.eclipse.org/Sirius/Tutorials/AdvancedTutorial.

[13] Sirius dokumentáció. https://www.eclipse.org/sirius/doc/.

[14] Sirius starter tutorial. https://wiki.eclipse.org/Sirius/Tutorials/StarterTutorial.

[15] UML introduction. http://www.uml.org/what-is-uml.htm.