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

Abstract

1. fejezet

Bevezetés

1.1. Témamegjelölés

Modell-vezérelt tervezésnek nagy szerepe van az informatikában egy szoftver vagy rendszer létrehozásánál. Modellek segítségével sokkal jobban, strukturáltabban és megbízhatóbban lehet tervezni. A modellek helyességét automatikusan ellenőrizni és a modellből kódot generálni is lehet. Ehhez a rengeteg eszköz áll rendelkezésünkre, viszont ezek nem mindenre nyújtanak megoldást.

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 modellt, ami alapján lehetséges részleges modellek készítése. Ezután elkészítettem egy olyan vizuális szerkesztőfelületet, aminek a segítségével részleges modell példányokat lehet készíteni.

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 modellek szerkesztésének egy fontos funkciója. Ennek jelentését és jelentőségét a dolgozat folyamán részletesen kifejtem.

1.6. Dolgozat felépítése

Előismeretek részben 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 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.A szerkesztő minden eleméhez le van jegyezve annak megjelenése és működése. Végül a továbbfejlesztési lehetőségekről lesz szó.

2. fejezet

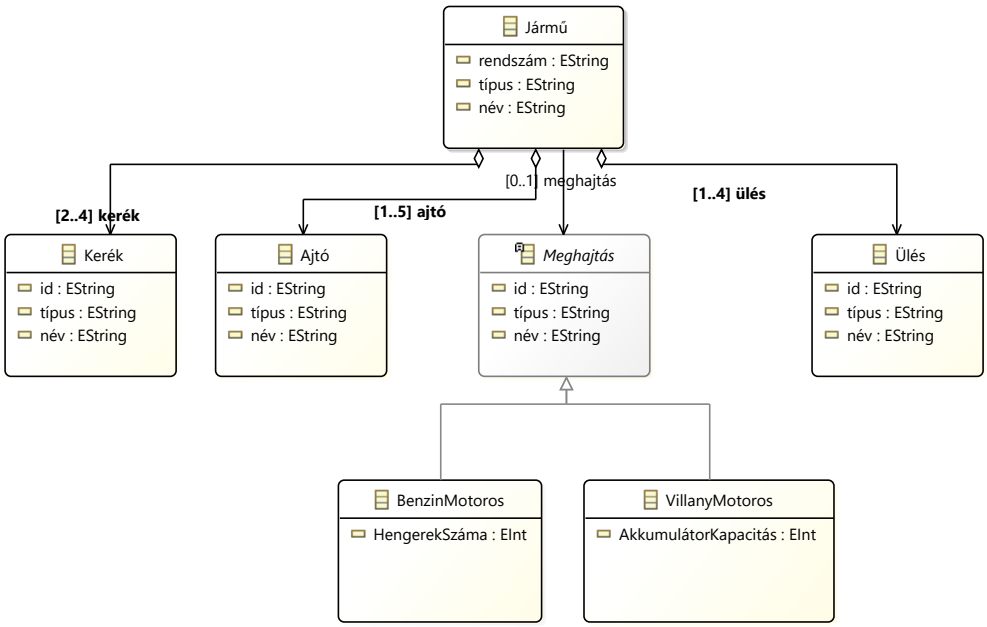
Előismeretek

2.1. Felvezetés

A részleges modellezést legegyszerűbben egy gyakorlati példán lehet bemutatni. Vegyük példának az UML [13] osztálydiagramját. Az UML (Unified Modeling Language) egy szabványos, objektumorientált 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 objektumok 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: id, 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.

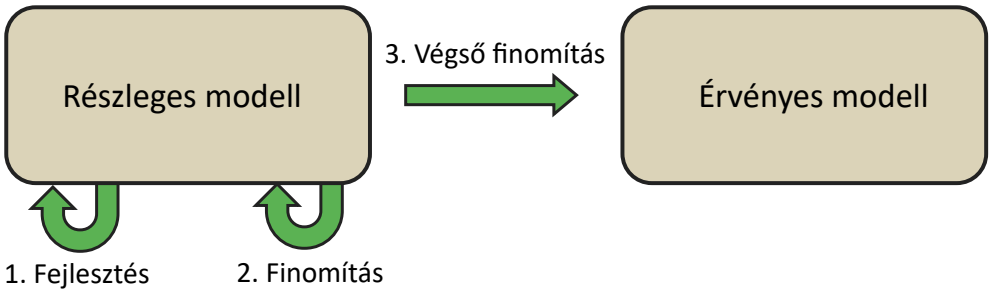


2.1. ábra. Jármű egyszerűsített osztálydiagramja

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 strukturális információt tartalmaz, hanem a részlegességéről is, tehát hogy teljesen specifikált a modell vagy sem. Az előbbi példát tekintve, lehetséges az hogy a járműnek egyáltalán nem szeretnénk ajtót, mert például egy motort szeretnénk 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 picike 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 25 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. Modellezés során 4 fajta részlegességet jelölhetünk meg [9]. Annotációk a modell minden elemére kerülhetnek, lehet attribútumra, objektumokra vagy akár élekre is.

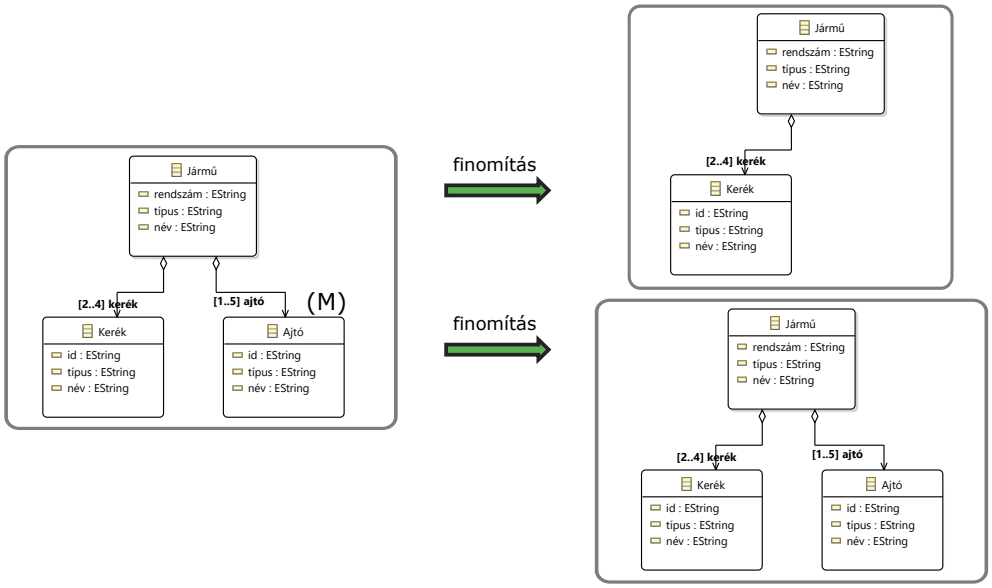
2.2.2. Szemantika

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.

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.

Az a jármű aminek nincs ajtaja lehet akár egy motor, a két ajtós változat pedig egy autó. Erre példa az alábbi diagramrészlet (lásd Ábra 2.3).



2.3. ábra. May részlegesség feloldása

Abs

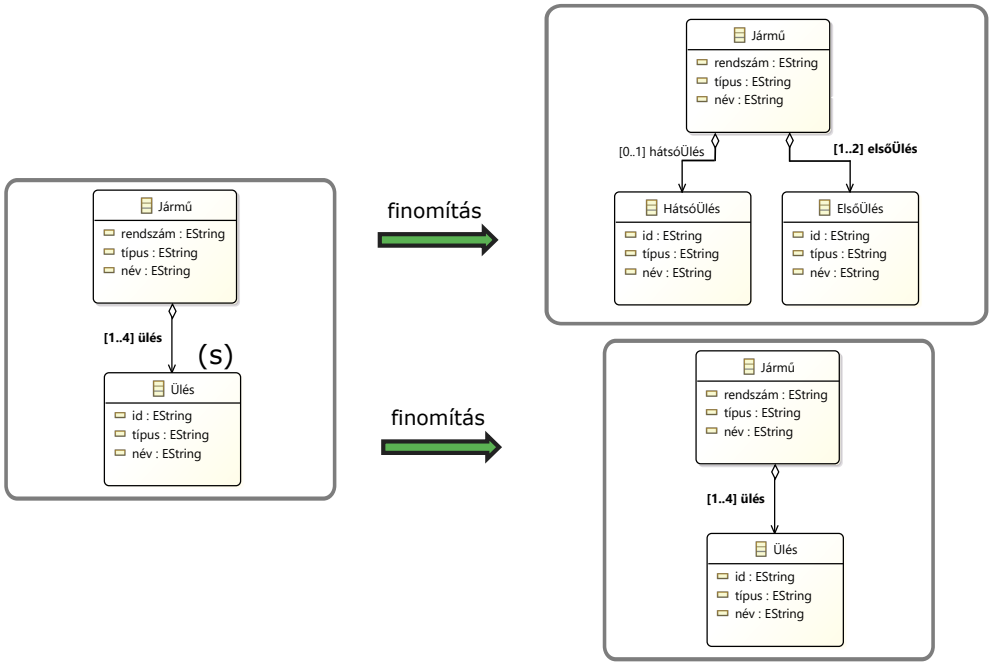
Ennek is két különböző annotálási módja lehet: 'P', Particular (egyedi elem) és 'S', azaz Set (egy vagy több elemet jelölhet). Particular az olyan elem, ami már biztosan benne lesz a végleges modellben, azonban a Set olyan elemet jelöl, ami lehet, hogy a végleges modellben több elemként fog megjelenni.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

Diagram kezdeti fázisában még nincs eldöntve, hogy az ülés az egyedi elem vagy sem, ezért meg van jelölve egy 'S' annotációval. Finomítás során lehetséges, hogy 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 de mégis modell szempontjából külön kezelendőek. Erre példa az alábbi diagramrészlet (lásd Ábra 2.4).

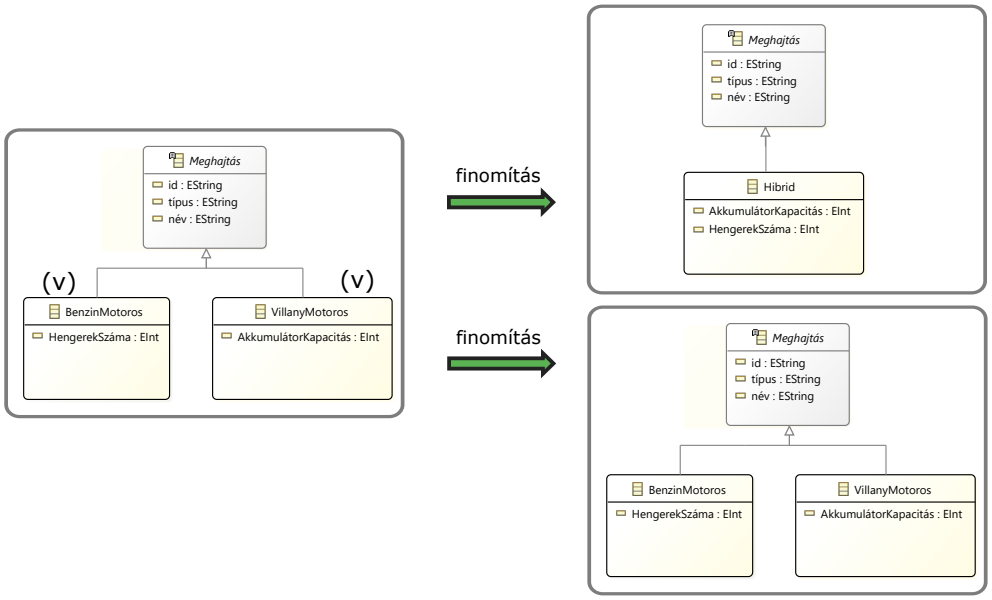
Var

Kétféleképpen lehet annotálni az elemeket: 'C' Constant (konstans elem) és 'V' Variable (változó elem). Bizonyos értelemben az Abs fordítottjának lehet tekinteni. Felveszünk több elemet, ami lehet, hogy későbbiekben összeolvasztunk egy elemmé, tehát fenn áll a lehetősége annak, hogy két elem megegyezik. Amikor elkezdünk egy modellt, akkor nem biztos, hogy meg tudjuk mondani elemekről, hogy azok a későbbiekben azonosak-e vagy különbözőek. A végleges modellben már csak konstans elemek lehetnek.

Itt a példában (lásd Ábra 2.5) látható, hogy a kezdeti állapotban még két külön kereke van a járműnek egyik kerekén kék dísztárcsa van a másik kereke pedig széles felnivel rendelkezik. Ezek meg lettek jelölve 'V' annotációval. Finomítás után ez megmaradhat ilyen különálló formában, de akár összeolvaszthatjuk ezt a két kereket és a végeredményben egy kerék marad, ami mind a kettő kerék tulajdonságát magában hordozza.



2.4. ábra. Abs részlegesség feloldása



2.5. ábra. Var részlegesség feloldása

OW

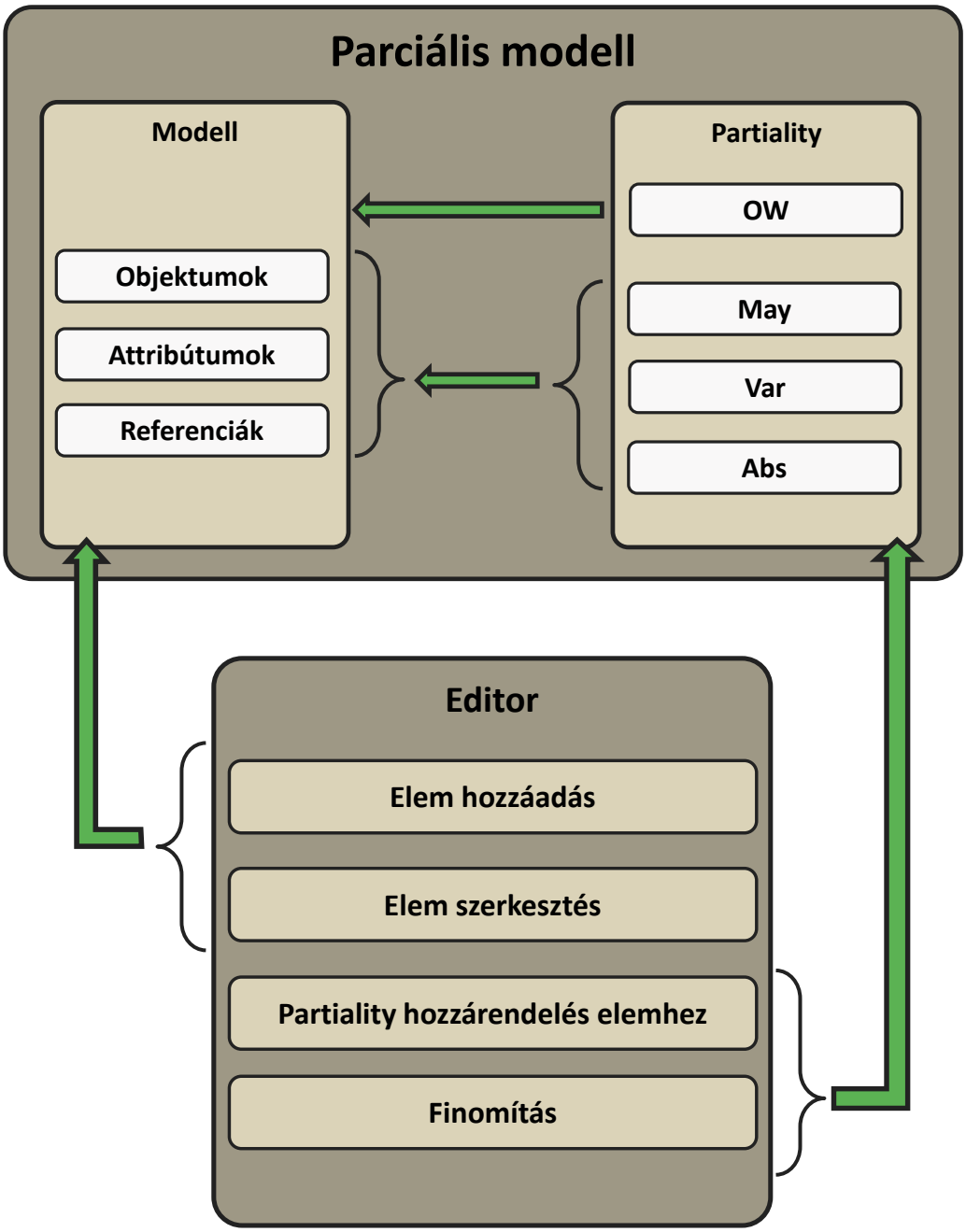
Modellezés folyamán lehet megjelölni azt, hogy a modell már végleges-e vagy sem, tehát adhatunk-e új elemeket a modellhez vagy nem. Ez a részlegesség nem a modell elemeire vonatkozik, hanem a teljes modellről árul el információt.

3. fejezet

Áttekintés

Kutatásom megnéztem egy másik modellezőeszközt, ami a 'May' részlegesség feloldására nyújt megoldást és ezzel kapcsolatos döntéstámogatást szolgáltat[7]. A cél egy olyan modellező eszköz elkészítése, ami segítségével lehetséges részleges modelleket készíteni. Ehhez olyan vizuális szerkesztőfelület társul, ami megkönnyíti ezt a folyamatot. Ahhoz, hogy ez generikusan működjön egy általános modell szükséges. Így ez a modell tartalmazni fog objektumokat, attribútumokat és az ezek közti kapcsolatot kifejező referenciákat. Ezen felül a részlegesség kifejezésére minden ilyen elemhez lehetséges rendelni *May*, *Var* vagy *Abs* részlegességet. Magához a modellhez pedig *OW* részlegességet lehet rendelni.

A kapcsolódó editor képes részleges modellt létrehozni és manipulálni. Lehetőséget ad új objektumok, attribútumok, referenciák létrehozására. Ezen elemekhez a már fent említett részlegességek rendelhetők. A szerkesztő a részlegességek feloldására, tehát finomításra is biztosít eszközöket.(áttekintő ábra lásd Ábra 3.1)



3.1. ábra. Részleges modell és a rajta végezhető műveletek

4. fejezet

Megvalósítás

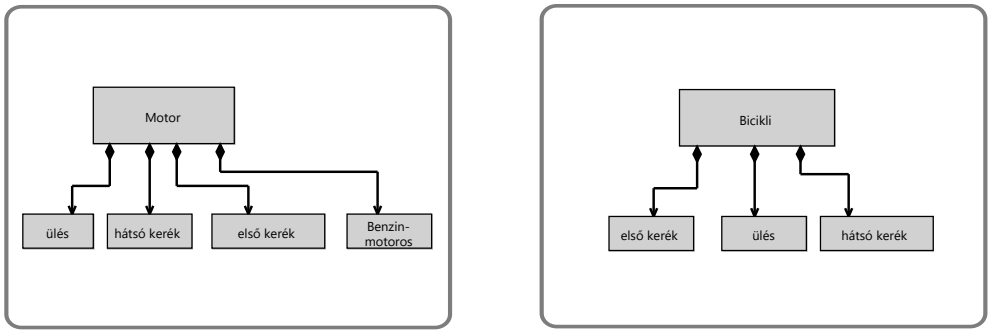
4.1. Szükséges eszközök

4.1.1. Eclipse

A feladatot Eclipse-be 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 (lásd Ábra 2.1) tekinthető metamodelljének. Bal oldalon egy motor példánymodellje, jobb oldalon pedig egy biciklinek a modellje látható.



4.1. ábra. Jármű metamodellhez tartozó példánymodellek

4.1.3. EMF

EMF az Eclipse Modelling Framework [4] rövidítése. Ez egy olyan keretrendszer, mely az eclips-be beépülve teszi lehetővé modellek készítését. Segítségével könnyedén grafikus eszközökkel lehet 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 [11]. Ez lehetővé teszi hogy EMF modellekhez vizuális megjelenítő és szerkesztő felületet készítsünk. Rendkívül sokrétű lehetőséget nyújt. Lehetőség van új objektumok létrehozására szerkesztésére és biztosít validációs lehetőségeket is. Az egyes elemek módosítását megszorításokhoz lehet kötni. Egy úgynevezett "viewpoint specification" projektet kell létrehozni, majd ezen belül egy "odesign" kiterjesztésű fájlt. Ebben a fájlban be kell állítani milyen modellel szeretnénk dolgozni és azután lehet a modellhez tartozó editor működését és megjelenését definiálni. Ki lehet választani milyen típusú reprezentációt szeretnék a modellemnek. Például: diagram, táblázat, fastruktúra. Meg lehet adni hogy a modellben lévő elemek vizuálisan hogyan jelenjenek meg. Ez a stílus akár dinamikusan változhat az elem tulajdonságától függően. Ebben a fájlban lehet pontosan megadni, hogy mi történjen egy elem létrehozásánál, törlésénél vagy módosításánál. Ezen események bekövetkezése kiválthatnak további eseményeket. Például kitörlök egy elemet és ennek következtében módosítok egy másikat.

4.1.5. Java

Objektumorientált programozási nyelv. EMF keretrendszer biztosít lehetőséget arra, hogy az elkészített modellből Java kód generálható. Az EMF modellből kigenerált kódot futtatva egy új Eclipse alkalmazás indítható, melyben már lehetőség van a metamodellben definiált modell példányosítására. Siriusban képes külső Java kód futtatására. Ezzel például bővíthető a szerkesztőfelület funkcionalitása.

4.1.6. Aql

Az Annotation Query Language [3] rövidítése. Lambda [6] kifejezésekhez hasonlóan lehet alkalmazni. Az editor elkészítésébe van nagy szerepe. Az egyes funkciókat előfeltételeit lehet leírni ezen a nyelven.

4.1.7. Általános modell

Jellemzés

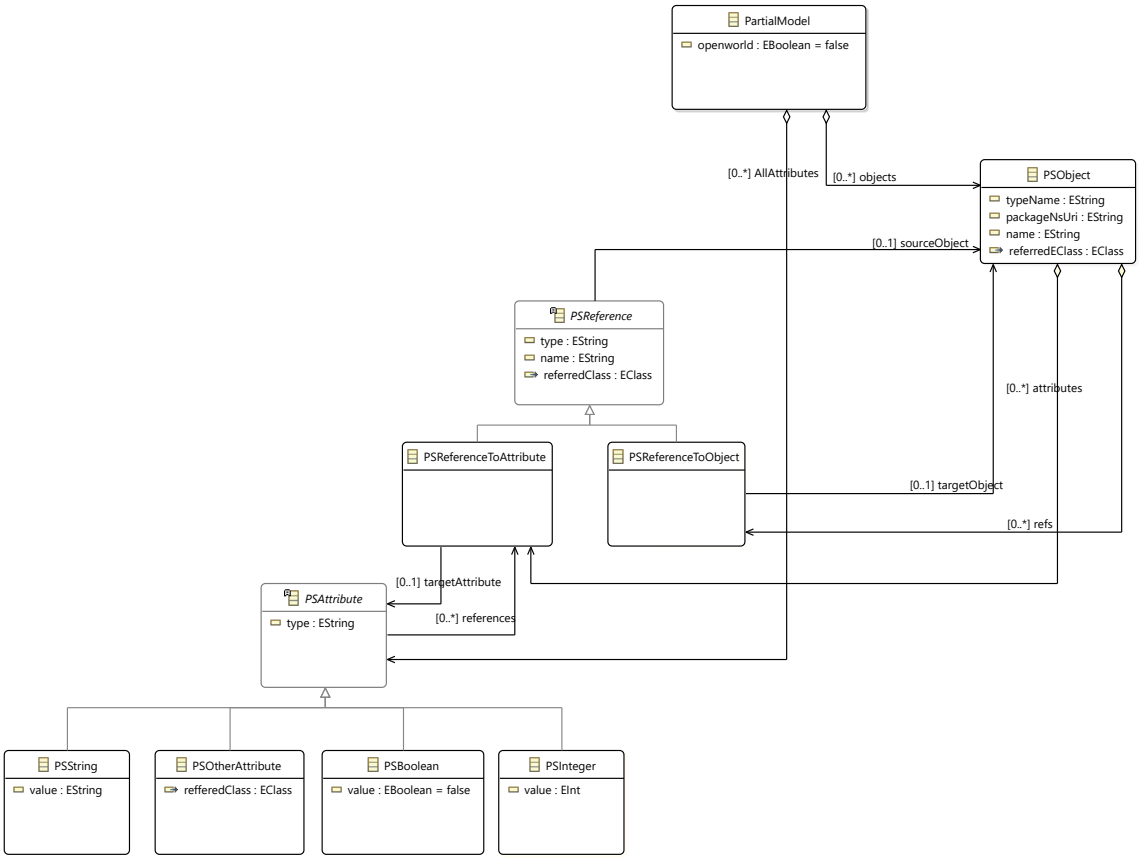
Egy általános modell elemekből áll, amik tulajdonságokkal rendelkeznek, az elemek között pedig kapcsolatok vannak. Tehát olyan meta modellt kell készíteni, ami mindezeket magába foglalja. A metamodell elkészítéséhez EMF-et használtam.

PSObject

A fenti modellben (lásd Ábra 4.2) a PSObject-ek reprezentálják az általános modell elemeit. Ez bármilyen típusú lehet. Rendelkezik névvel (name), referált osztállyal (ReferredEClass) és a referált osztály konkrét példányának az azonosítójával (uri). így a PSObject-en keresztül tudunk részlegességet rendelni a hivatkozott objektumhoz, ami erre egyébként nem lenne képes. PSObjectek tartalmaznak tetszőleges mennyiségű referenciát másik objektumokra (PSReferenceToObject) és attribútumokra (PSReferenceToAttribute).

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 későbbiekben



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

a szerkesztő elkészítésénél más funkcionalitások vonatkoznak rájuk. Rendelkezik névvel (name), referált osztállyal (ReferredEClass) és típussal (type). Van neki forrásobjektuma (sourceObject) és célobjektuma (targetObject) vagy célattribútuma (targetAttribute).

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, ez a szerkesztő felület szempontjából lesz fontos. PSAttribute is absztrakt ezért több fajtája lehet:

- PSString
- PSBoolean
- PSInteger
- PSOtherAttribute

4.1.8. Kiegészítés részleges modellé

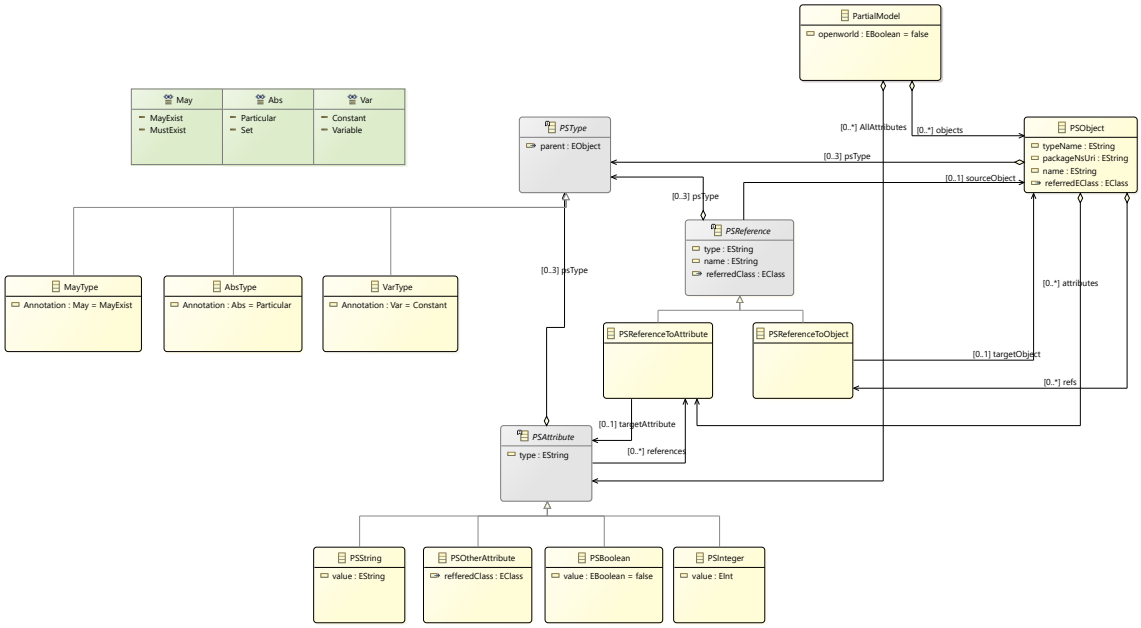
OW részlegesség

Az *OW* részlegességet a modell gyökerében egy boolean változóval tudjuk szemléltetni. Fenti metamodell alapján lehetőségünk van részleges modellek készítésére.

PSType

Minden egyes modellbeli elemhez tudnunk kell társítani annotációkat. Ezt a PSType-al tehetjük meg, amiből leszármaznak a már említett részlegesség fajták (*May*, *Var*, *Abs*):

- MayType
- AbsType
- VarType



4.3. ábra. Általános modell kiegészítve részleges modellekre jellemző tulajdonságokkal

4.2. Szerkesztő

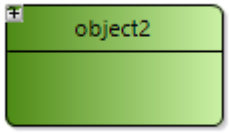
4.2.1. Példánymodell felülete

PSObject

Világoszöld téglalapként jelenik meg. 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.



4.4. ábra. PSObject

- PSObject létrehozása a szerkesztőfelület oldalán lévő sávból kiválasztva lehetséges. Az Object nevet kiválasztva, majd a szerkesztőfelületre kattintva megjelenik a négyzet egy alapértelmezett névvel. A név formátuma: 'Object{szám}'. A szám helyén egy automatikusan generált szám kerül, a már meglévő PSObject-ek száma növelve eggyel. Ezt aql kifejezéssel lehet megtenni:

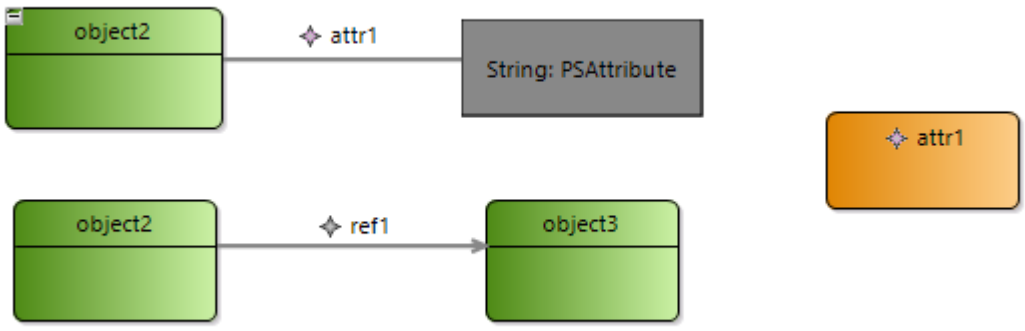
$$aql :' object' + container.objects- > size() \tag{4.1}$$

- 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.

PSReference

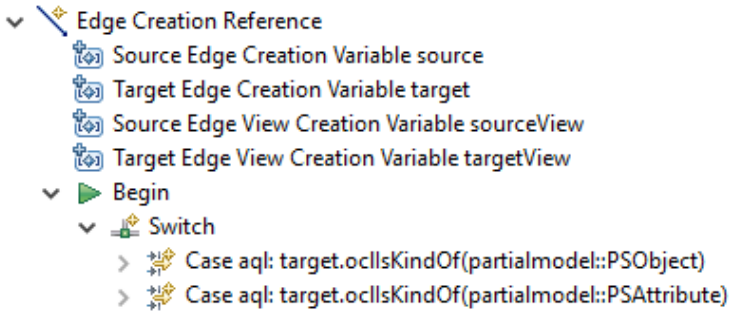
A felületen szürke vonalként jelenik meg. Összeköthet PSObject-et attribútummal vagy egy másik PSObject-el. Utóbbi esetben a vonal végén nyíl is van. A referencián szövegesen megjelenik neve és zárójelben a hozzá tartozó részlegességek is. Amennyiben rákattintunk a referenciára kétszer megjelenik egy másik szerkesztőfelület, ahol a referencia egy narancssárga téglalapként jelenik meg.



4.5. ábra. PSReference (baloldalt a részleges modellen megjelenő formában, 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 (lásd Ábra 4.6). 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 nem szükséges plusz logikát beiktatni. Sirius beépítetten kezeli.



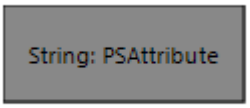
4.6. ábra. Referencia létrehozásának logikája

PSAttribute

Szürke téglalapként jelenik meg a szerkesztőfelületen. Az attribútum típusa és értéke a téglalap közepén látható kettősponttal elválasztva.

Funkciók:

- Létrehozása a szerkesztőfelület oldalán lévő sávból kiválasztva lehetséges (UndefinedAttribute). Ezután a szerkesztő panelen kattintva létrejön egy attribútum. Ez nem tartozik még egyetlen objektumhoz sem.

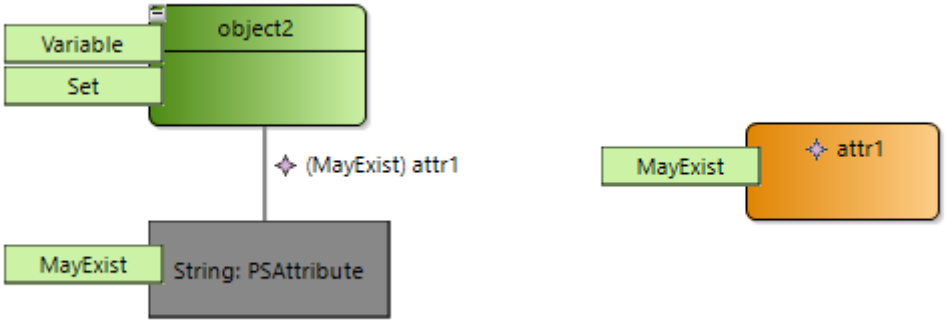


4.7. ábra. PSAttribute

- 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.

PSType

A szerkesztőfelületen világoszöld téglalapként jelenik meg a mellett az elem mellett amihez hozzá lett rendelve. Referencia esetén zárójelben látszik az él neve előtt, de ha kétszer rákattintunk a referenciára, hogy megnyissuk a szerkesztőfelületét, ott az élet reprezentáló narancssárga téglalap mellett fog megjelenni.



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

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 az részlegesség. Él esetében ugyan ez a referencia saját szerkesztőfelületén lehetséges.
- Törlés esetén nem szükséges plusz logikát beiktatni. Sirius beépítetten kezeli.

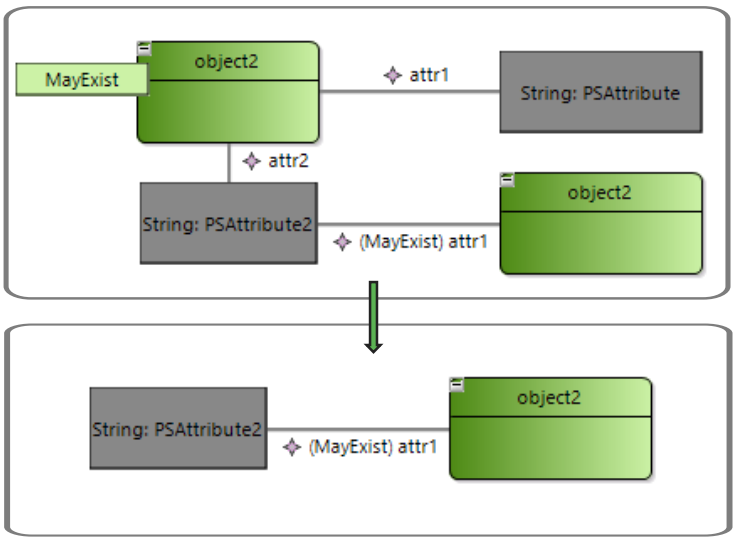
4.2.2. Finomítás

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 feloldása

Feloldásnál az objektum és az összes hozzátartozó attribútum törlődik. Ezt úgy oldottam meg, hogy végigiterál az összes élen ami attribútumra mutat és törli azokat. Amennyiben az attribútumra mutat más objektumból is referencia akkor az nem törlődik (lásd Ábra 4.9). Aql feltétel, ami azt vizsgálja, hogy nem vezet-e más objektumból él az attribútumba:

$$aql : i.targetAttribute.references- > forAll(x|x.sourceObject = element.eContainer()) \tag{4.2}$$



4.9. ábra. PSObject finomítása (May)

PSAttribute May 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 4.10).

PSReferenceToObject May feloldása

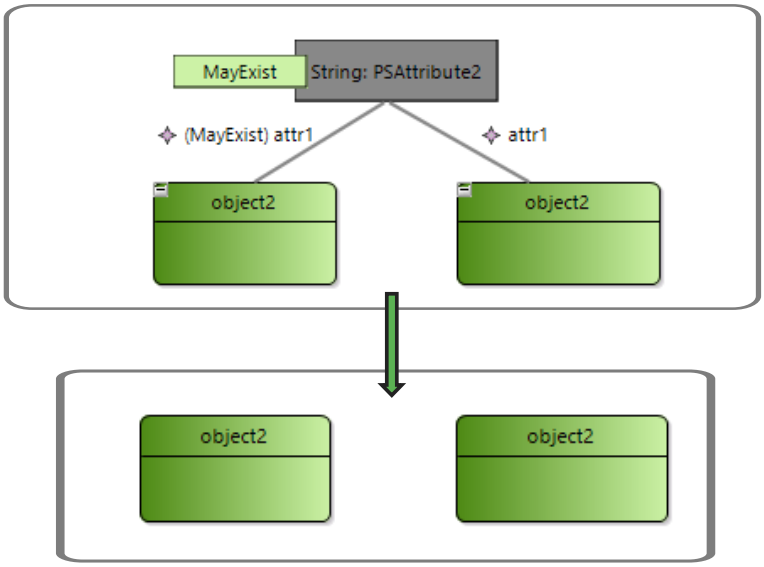
Kitörli a referenciát a forrásobjektumból.

PSReferenceToAttribute May feloldása

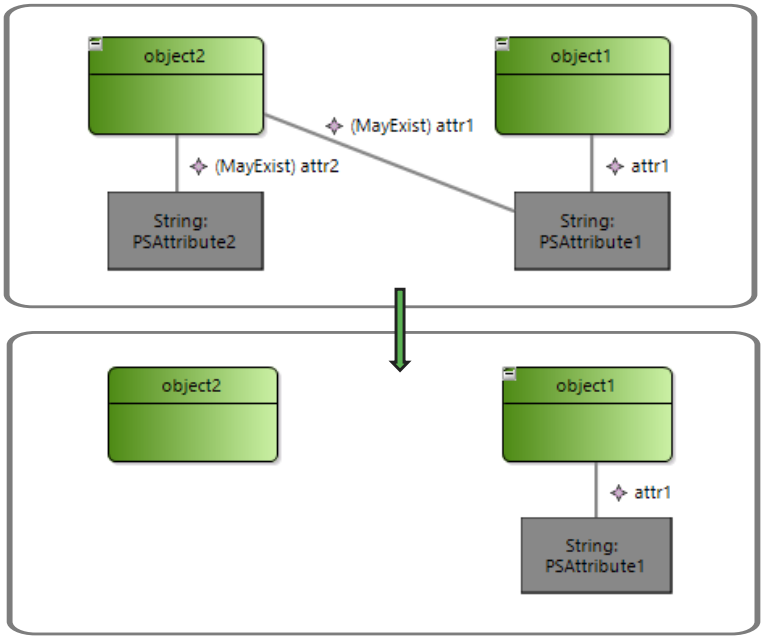
Az él törlődik, de eltérően az PSObject-ekre mutató referenciától, amennyiben a célattribútumba nem vezet másik él, akkor az attribútum is megszűnik. A finomítás lényege, hogy általa csökken a részlegessége a modellnek. Ezért nem lenne értelme az üresen maradt attribútumot meghagyni, ami nem kötődik semmilyen objektumhoz, mert nem csökkent volna a modell részlegessége (lásd Ábra 4.11).

PSObject Set 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



4.10. ábra. PSAttribute finomítása (May)

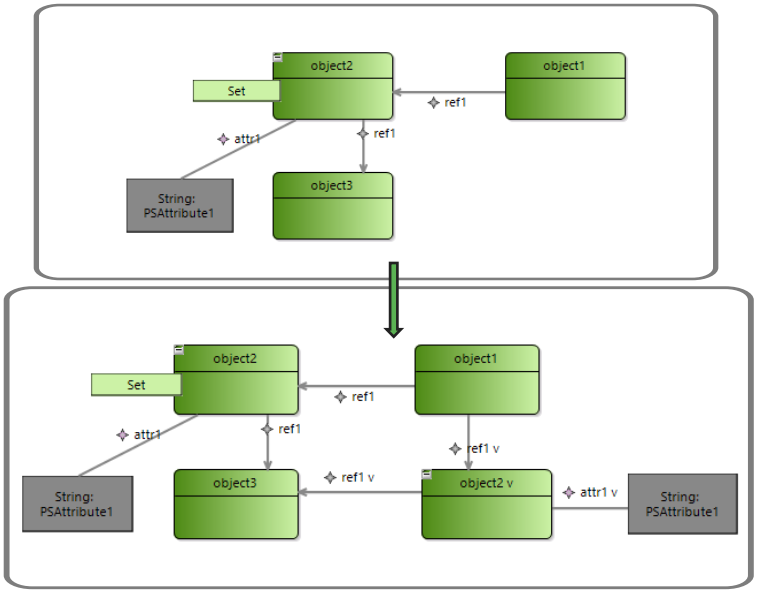


4.11. ábra. PSReferenceToAttribute finomítása (May)

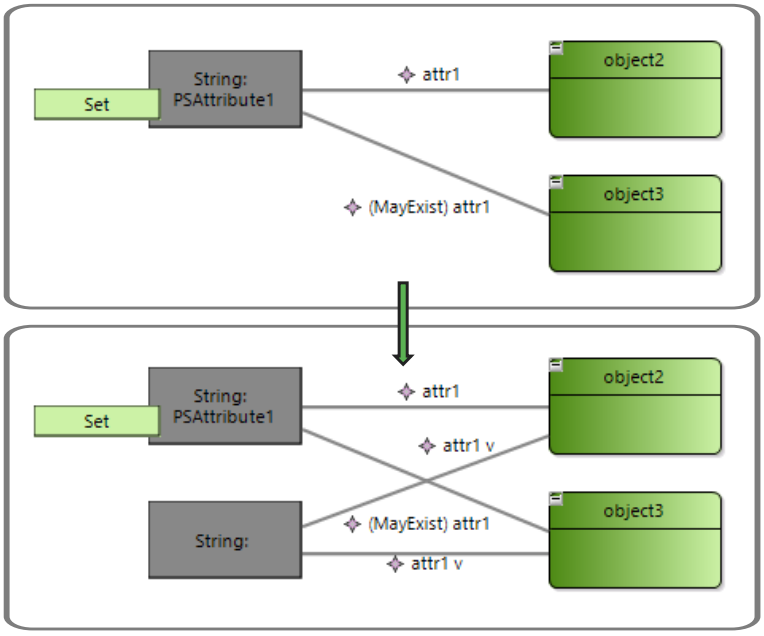
mutatnak, mint az eredeti PSObject-nél. Új attribútumok jönnek létre az eredeti objektumhoz tartozó attribútumok mintájára és ezek lesznek az új elem attribútumai. Az új élek nevei ugyan azok mint az eredetié de a végére kerül egy kis 'v' betű (lásd Ábra 4.12).

PSAttribute Set 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. Az új elemhez tartozó referenciák nevei után egy kis 'v' betű kerül (lásd Ábra 4.13).



4.12. ábra. PSObject finomítása (Set)



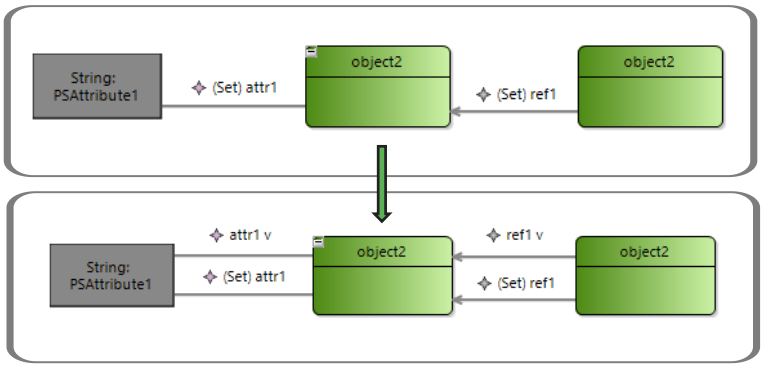
4.13. ábra. PSAttribute finomítása (Set)

PSReferenceToObject Set feloldása

Az eredeti referencia forrásobjektumában létrehoz egy újat, ami ugyan arra az objektumra mutat (lásd Ábra 4.14).

PSReferenceToAttribute Set 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 4.14).



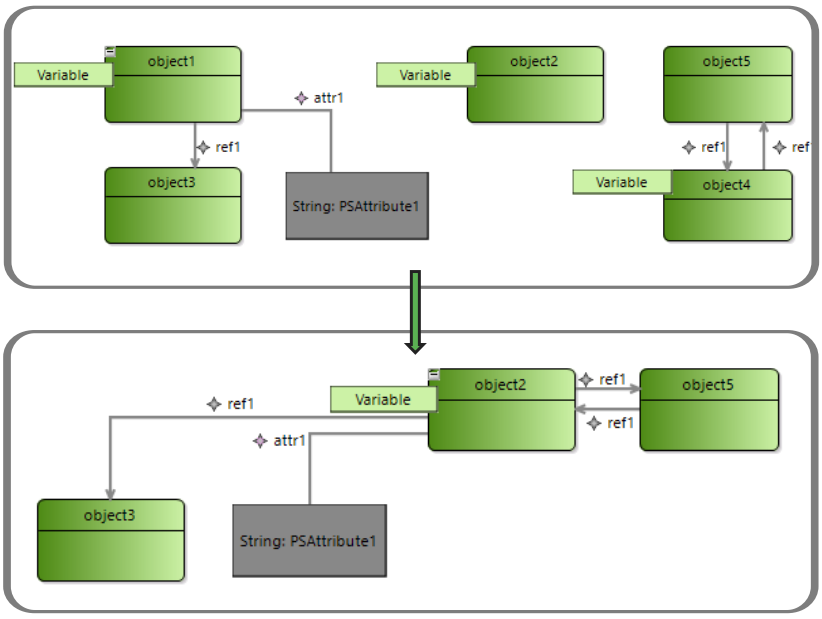
4.14. ábra. PSReference finomítása (Set)

Var részlegességről általában

Eredeti objektumnak, elemnek nevezzük azt az PSObject-et amelyiknek 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 feloldása

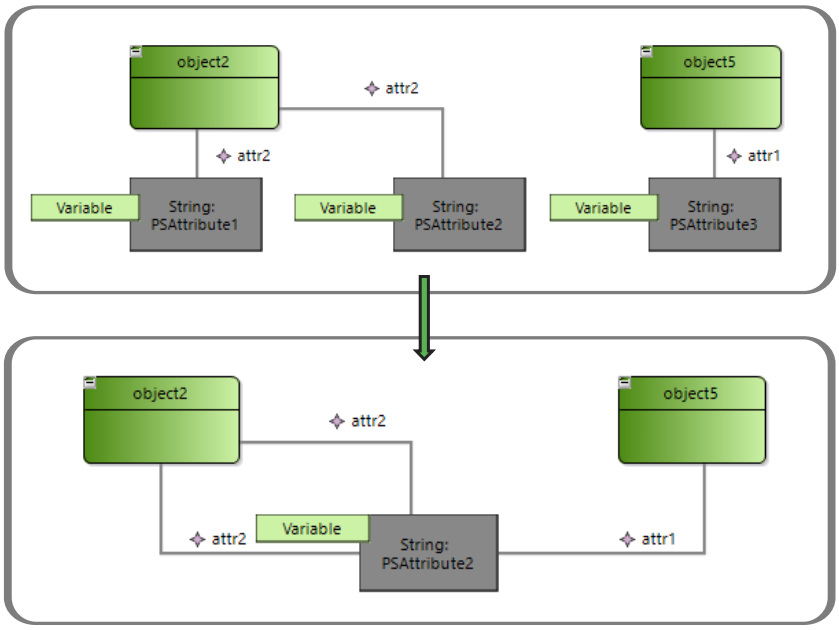
Az összes olyan objektumnak, amelyiknek megegyezik a 'Var id'-ja az törlődik, kivéve az eredeti de előtte az összes hozzá tartozó referencia mintájára létrejönnek az eredeti objektumban élek. Azok az objektumok élei amikből eddig egy törölt objektumba mutattak, azok most az eredeti objektumra fognak mutatni (lásd Ábra 4.15).



4.15. ábra. PSObject finomítása (Var)

PSAttribute Var 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 4.16).



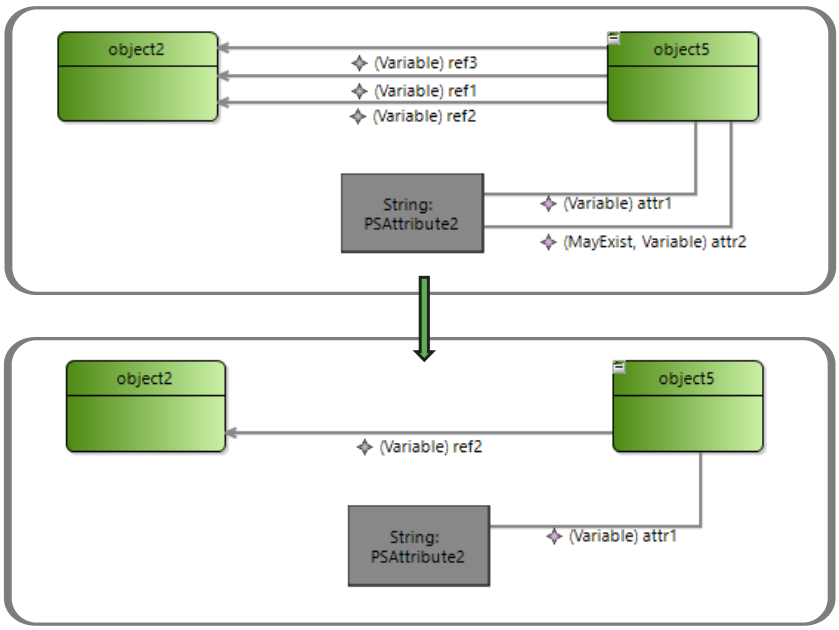
4.16. ábra. PSAttribute finomítása (Var)

PSReferenceToObject Var feloldása

Csak az eredeti referencia forrásobjektumában keres vele megegyező 'Var id'-val rendelkező referenciát. Ezeket a referenciákat törli (lásd Ábra 4.17).

PSReferenceToAttribute Var feloldása

Hasonlóan a PSReferenceToObject esetéhez csak az eredeti referencia forrásobjektumában keres vele megegyező 'Var id'-val rendelkező referenciát. Ezeket a referenciákat törli (lásd Ábra 4.17).



4.17. ábra. OSReference finomítása (Var)

5. fejezet

Összefoglalás és továbbfejlesztési lehetőségek

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

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

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

[7] 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.

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

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

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

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

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

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

Függelék