Multi-Level Modeling with Melanee

Sara Sali

University of Antwerp

sara.sali@student.uantwerpen.be

Abstract

Multi-level modeling has attracted growing interest over the last few years. There is a big number of practical tools that support this approach such as MetaEdit+, GMF, EMFText, XText, MetaDepth, Spoofax, AToMPM, Microsoft Visual Studio DSL and they are divided in two major groups: graphical and textual language based. This report explains some of the basic features and the architecture of the deep modeling tool Melanee (or sometimes referred to as Melanie, MultiLevel Modeling and Ontology Engineering Environment) and deep modeling itself through some basic examples and concludes with an evaluation of the tool and comparison to AToMPM.

Keywords: melanee, melanie, deep modeling, multi-level modeling, meta modeling, modeling, deep instantiation, deep characterization, online retail example, AToMPM

1. Introduction

Domain Specific modeling tool can be seen as divided in two major groups. One group supporting graphical domain specific modeling (DSL) such as: MetaEdit+, Poseidon for DSLs, Microsoft Visual Studio for DSLs , Atom3/AToMPM; and the other group supporting textual domain specific modeling such as: XText, JetBrains MPS, Spoofax, MetaDepth, EMFText.

Melanie is a tool which supports deep of both graphical and textual domain-specific modeling. Although there is no clear study case where this duality could be useful, in order to solve this problem a software engineer would have to create a mapping between the graphical and textual modeling

Email address: `sara.sali@student.uantwerpen.be` (Sara Sali) frameworks. If they were based on the same infrastructure then you could consider it as a lucky case (an example would be Xtext and GMF both based on the Eclipse Modeling Framework (EMF) and the Eclipse Platform). Creating the mapping would be easier compared to a more complex case where the tools used are based on different platforms. This would increase the software engineering process complexity and probably the cost of it.

What is also special about Melanee is that it is a Multi Level Modeling Tool, which means that you can create more than the traditional Abstract/Concrete Syntax levels avoiding classification issues and making a clear division between ontological and linguistic dimensions. This paper begins with a section on MLM (Multi Level Modeling) where the concept is explained, how it evolved from the issues found in the UML framework and it concludes with a comparison with it's traditional ancestor Meta Model. Section 3 gives a description of the architecture of the Tool. In section 4 the case study that will serve to test the tool is explained. It is constantly modified throughout the paper to perform different tasks or show important issues. Naturally, in section 5 it is shown the implementation on the tool, a solution to a problem while installing and a description of a failed trial to test the Textual DSL Visualizer. Section 6 throws some ideas for future work and section 7 gives some conclusions and compares Melanee with an other modeling tool, AToMPM.

2. Multi Level Modeling

2.1. Concept

Multi-level modeling also called deep meta-modeling extends the standard approach to meta-modeling by enabling modeling at an arbitrary number of meta-levels, not necessarily two. It is sometimes also referred to as ontological multi-level modeling.

With the meta modeling concepts (as defined by the UML documentation "A model is an instance of a metamodel and a metamodel is an instance of a meta-metamodel") something as the following picture can be achieved.

Wild is a representation of a documentary of Wild Life from 2001, in the next examples the 2001 name will be used. It is an instance of Video and video is an instance of class. So in this example we have a User Data level (Wild), a UML Model (Video) and the UML Meta Model (Class) which is an example of more than two level of meta models.

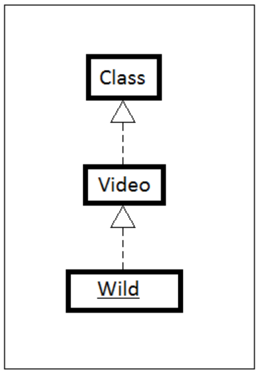


Figure 1: Three Level MM

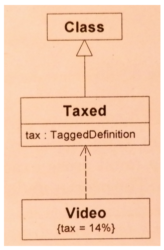


Figure 2: Stereotype

The concept of strict metamodeling provides a foundation upon which a multi level modeling hierarchy is based.

Strict Metamodeling: In a n-level modeling architecture M0,M1,...,Mn-1, every element of an Mm level model must be an instance of exactly one element in the Mm+1 level model, for all m from 0 to n-1, and any relationship other that the instance of relationship between two elements X and Y implies that level(X)=level(Y).

2.2. Traditional Modeling Issues (as seen in the UML framework)

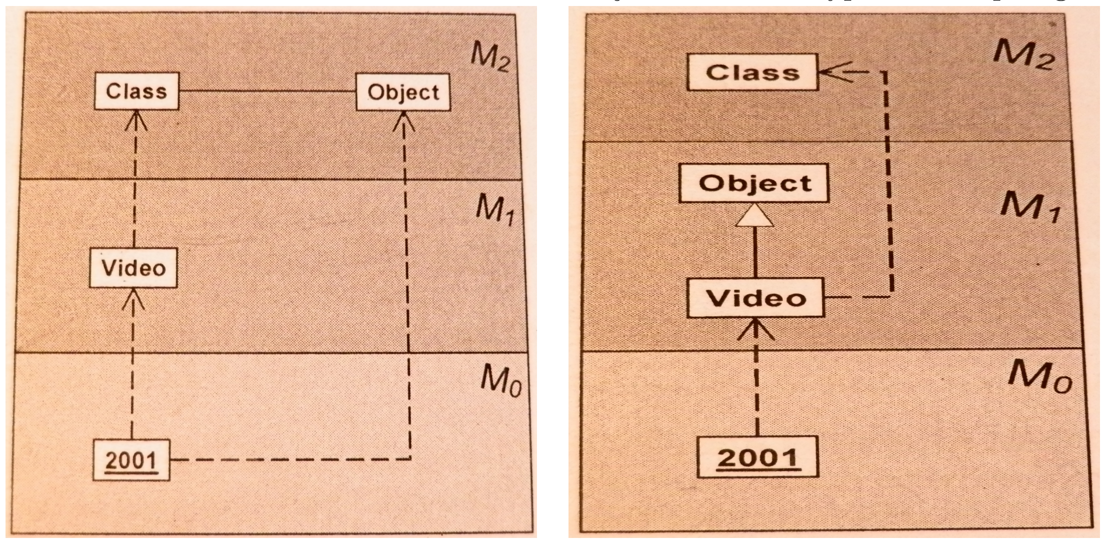
Stereotype mechanism would change the above model as Figure 2:

which would require that in order for the user to modify the tax value he/she should be able to access the higher level. This is a problem since it would require for tools not to assume a hard-coded metalevel but the user should be able to make changes to the metalevel.

In certain situations some *problems arise while modeling with the UML framework.*

Multiple classification As shown in the picture below the 2001 instance is classified both as an instance of domain type Video and an instance of metatype Object.

Multi Classification issue on the left and Object as a Prototypical concept right

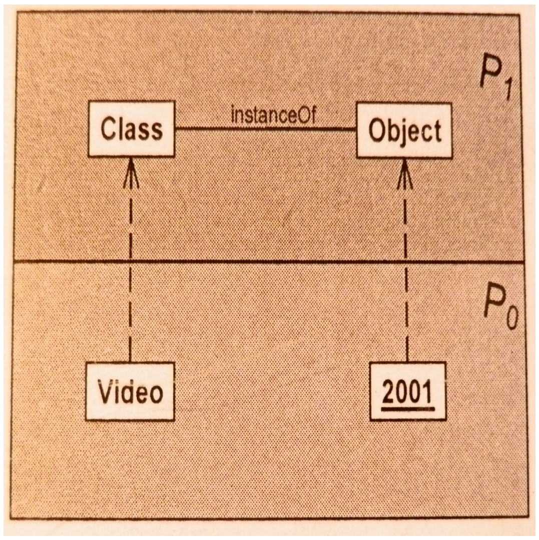


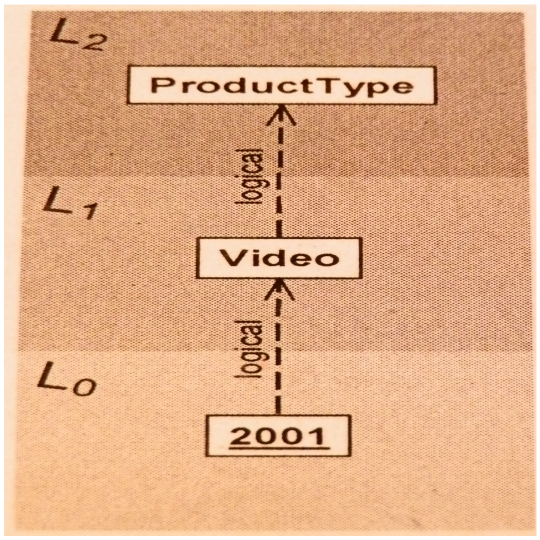
2.3. Multi Level Modeling concepts/Solutions

When trying to fix the issues of the previous subsection first by prototypical concepts then with metmodels as language definition other disadvantages are seen. And Atkinson and Kuhne (**?**) introduce three techniques or concepts that extend the standard two-level meta-modelling approach to improve it:

1. *OCA (Orthogonal Classification Architecture)*
   As explained above one of the issues with the standard meta modeling approach is the multi classification of objects. This issue comes from not properly recognizing and accommodating two different forms of classifications, the physical and logical classification.
   The physical classification dimension is the dominant viewpoint of tool builders and the logical classification dimension is the dominant viewpoint of modelers (i.e., users of UML). The key to integrate these classifications is to view them as orthogonal dimensions. By doing this the

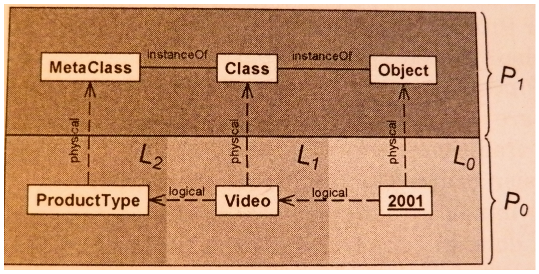
Physical dimension on the left and Logical Dimension on the right

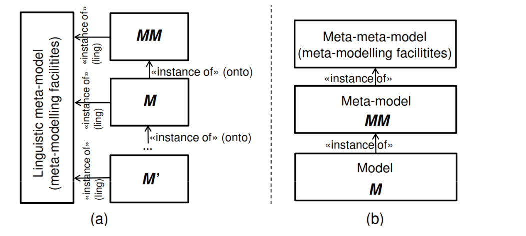




doctrine of strictness is restored and applied to each of the dimensions individually in a natural way (There is instance-of relationship between the elements in level n-1 and level n such that the element in the lower level is an instance of no more than one element in the higher level).

Two dimensional classification framework





(a) Orthogonal classification architecture. (b) Standard meta-modelling architecture.

The two levels can also be called *ontological and linguistic* dimensions. The ontological classication structure or logical dimension of an element expresses instantiation within a domain (ProductType - Video - 2001 - my2001Video). The linguistic typing or physical dimension of an element refers to the meta-modelling facility used for the construction of the element (field or clabject).

2. *Clabject (CLAss-oBJECT)*
   As seen in the example worked so far the Video's physical classifier is Class. However, from the point of view of ProductType, Video is an instance. For example, it may have a tax slot holding the percentage value to be used for the taxation of videos. From this point of view, Video should have Object as a classifier (Video has an attribute tax that is firstly declared at ProductType and Video as an instance of ProductType will have that as well, the instance facet, but Video might also have a type facet, an attribute that belongs only to it and the lower level instances of it will use, like for example the price attribute). For these situations the notion of claobject is introduced where an element in the middle levels can be both a class and an object.
3. *Deep Instantiation*
   In traditional modeling a type may specify properties of its direct instances (ProductType specifies Video, Video - 2001) but has no bearing on the instances of its instances (ProductType - 2001). This is why it is called *shallow instantiation* to separate it from the *deep instantiation* which has the above property to specify attributes for example of the instances of its instances.
   Modelers may need to add more levels to their model. In the example worked so far 2001 is a description of the movie. Not the real movie itself. Therefore a new lower level representing the real world may be added with elements that hold unique values such as serial numbers. Because the number of model levels may increase, so would the number of elements in general and in order to maintain the model with all these levels and elements, the notion of potency is introduced. The key idea is to specify a potency value to model elements and their values to indicate how many times they can be instantiated. Every level where an instance of that model is instantiated the potency value of that element will be reduced by one until it reaches 0. After that no more instances of this type can be instantiated.
   For example if the potency level of the ProductType element in L2 is 2 that means that a chain of its instances can be instantiated until L0. But if this initial value were to be 1 for example creating the 2001 instance would not be possible. This notion can be extended to creating or modifying attributes as in the Melanee tool. You assign potency to the attributes as in the picture above.

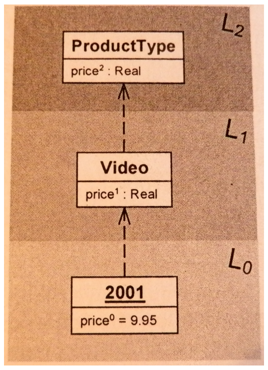


Figure 3: Deep Instantiation example in the above picture

2.4. Conclusions and Comparisons between MM and MLM

It becomes clear from the previous subsections that with increasing the number of levels in the traditional Meta Model modeling (MM) to 2+ (more than Abstract and Concrete Syntax) the models show it is not very well defined, so I would conclude that an advantage of Multi Level Modeling (MLM) is the simplification of the models by creating multiple levels.

Also because most Meta Model Environments are rigid (you can not change the meta model during runtime) dynamic addition of types/classes (for example CD to the example worked so far) is not possible, it requires a recompilation of the metamodel level (Abstract Syntax). In MLM thanks to the clabject concept these types can be added in the middle levels obeying the constraints of the above levels and creating new types for the lower levels.

Many levels can look like difficult to maintain but MLM has the notion of potency which helps with maintaining elements and attributes in different levels. While in the MM approach modeling a type can only control the features of its instances at the immediate metalevel below, the potency as a way for deep characterization, enables the declaration of properties for elements several meta-levels below.

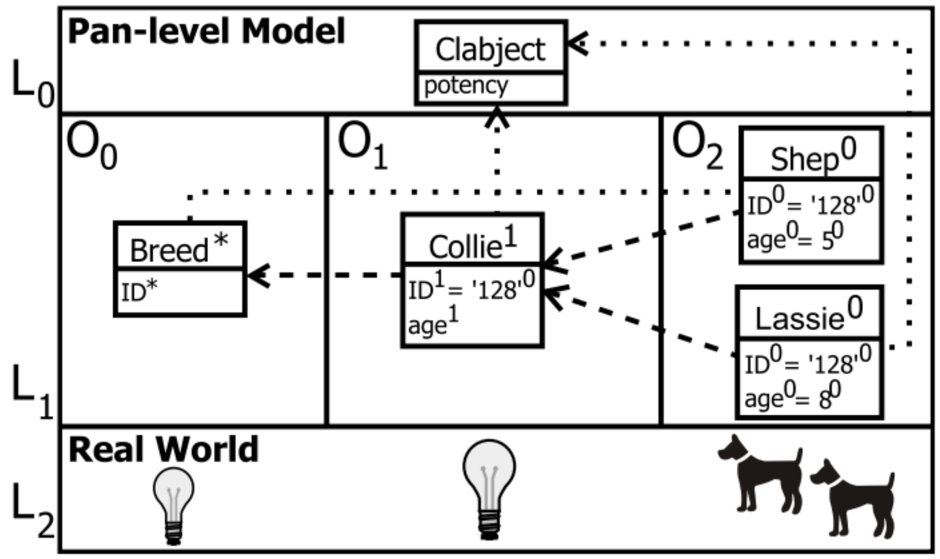
MM does not distinguish between logical and physical dimensions. In MM, Inheritence and Data Types (adding attributes) are available only at the MM Level, unless we model a way of how to use them in the concrete syntax level, unlike the LML where you can use the above in every level.

In conclusion, multi-level modelling can help in building systems in a simpler way as it will be shown in the next sections where it is described how the case study was chosen and how it was developed.

3. Melanie

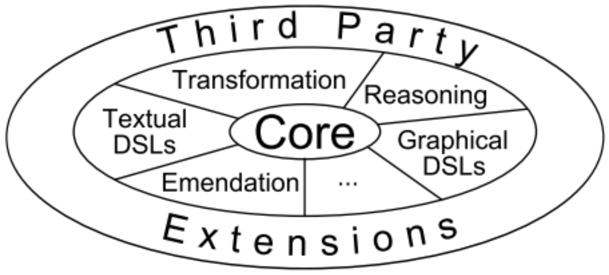
3.1. Melanie Components and Architecture

Multi Level Modeling in Melanie is made possible thanks to the orthogonal classification architecture, which separates the linguistic (horizontally dashed arrows) and ontological classifications (vertically dotted arrows) into two dimensions.



The Pan Level Model (PLM) in the picture above, defines the abstract language of any modeling language created in Melanie as an Ecore metamodel (EMF file). Modelers work on the L1 level, where they can create graphical concrete syntax for the PLM using the LML (Level agnostic Modeling Language).

The following picture shows Melanie and it's components:

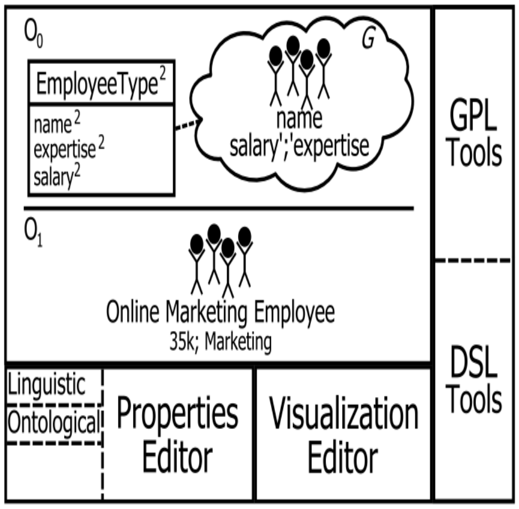


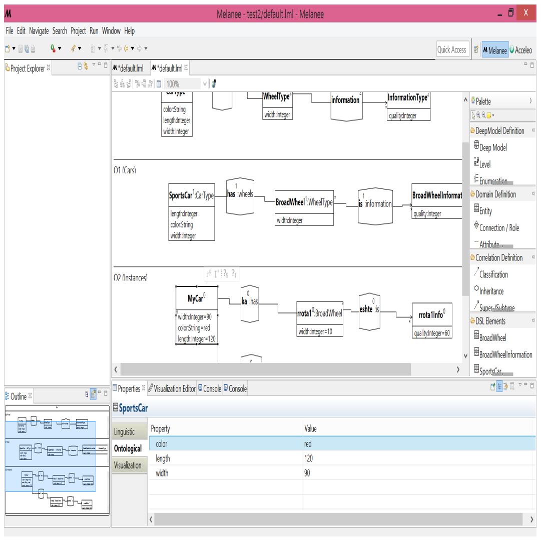
The core is in the center and it's responsible for creation and storage in EMF and rendering in LML. The components in the above layer are responsible for Domain Specific Languages support, reasoning and transformation. Melanie's architecture provides symbiotic domain specific/general purpose languages using the concept of Visualizers (how a model element is visually represented). There are two kinds of Visualizers LML and DSL. The DSL Visualizers can be textual or graphical.

1. **Reasoning and checking** in Melanie is composed by ontological validation which checks correctness at different levels of granularity and ontology population services which checks if there is any inheritance connection between two elements in the model.
2. **Graphical DSL** to add Domain Specific Language Graphical representation. If this is not done then the LML (General Purpose Language) is used. The relationship between the GPL and DSL is called symbiotic since they help reinforce each others strengths and may be used at the same time.
3. **Textual DSL** allows the default textual rendering of multi-level models to be augmented with a custom notation. The textual and graphical renderings of models are fully interchangeable and useable simultaneously.
4. **The emendation module** continually monitors the ontology and as soon as a change occurs that effects more than the currently edited model element it suggests additional changes to keep the ontology consistent.
5. **Multi-level aware Transformation Support.** Melanie incorporates an ATL adapter that supports multi-level aware transformations.

Below is shown a mock up of the tool editor and a picture of the real tool.

Melanie





SportsCar

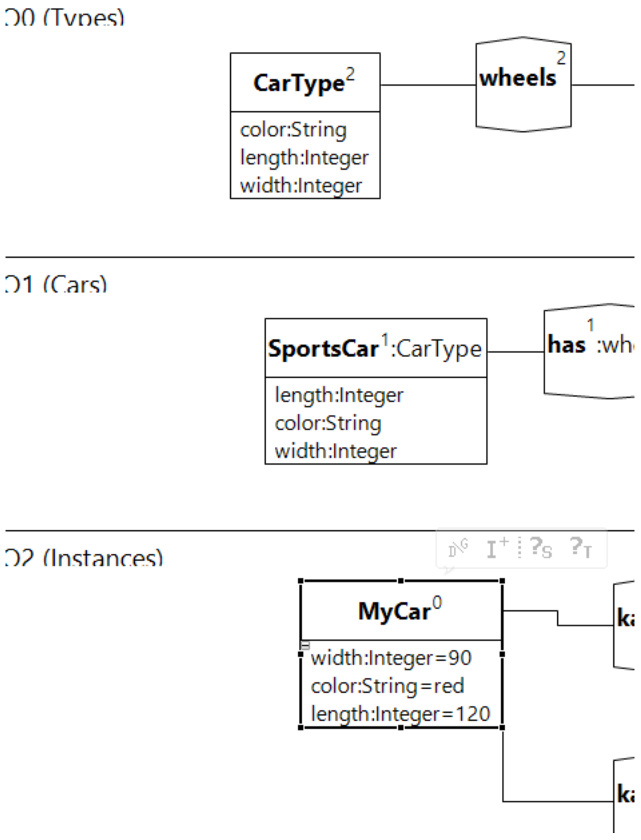
Property	Value
color	red
length	120
width	90

3.2. Melanie Features

In the paper "Rearchitecting the UML Infrastructure" of Thomas Kuhne and Colin Atkins (**?**) the new UML Infrastructure would have the features mentioned below. Next to each feature is written a short description of the implementation in Melanie.

- **Two or more Dimensional Modeling Framework.** (There are two dimensions: The physical, the dominant classification dimension from the view point of tool builders and the logical one from the viewpoint of the modelers.)

- **Unified Modeling Elements** (CLAss oBJECT (2)). The term clabject is introduced to reflect the type (class) / instance (object) duality of an ontological model-element. In the picture below the element SportsCar in the middle ontological level is a claobject since it's an instance of the CarType element and a class for MyCar.



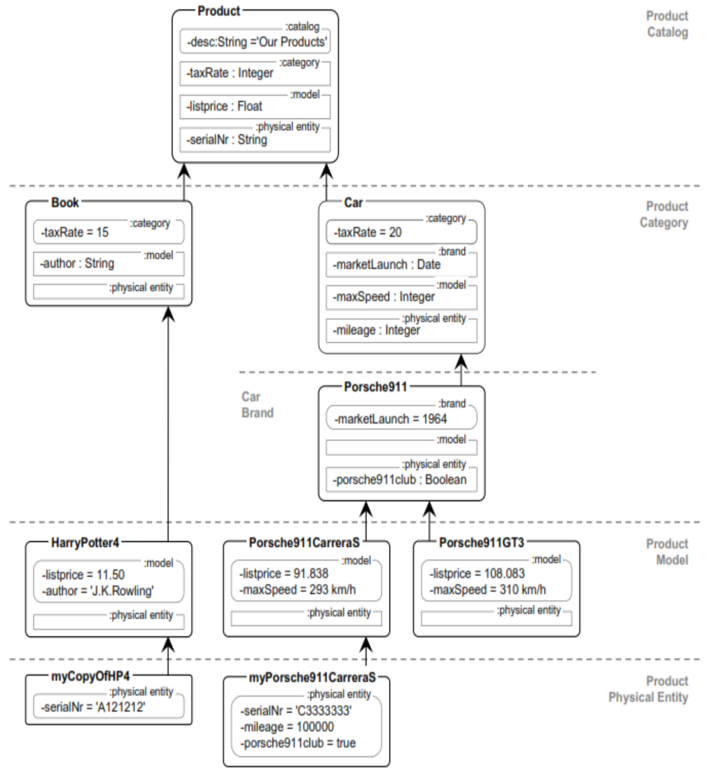
- **Deep Instantiation** (Potency) allows information to be carried over more than one instantiation level. In Melanie this is done by assigning a potency value to each element at the lowest ontological level (O0 in the example above). When the value of potency of CarType is set to two this means that instances of it can be created until the O2 level. This would not be the case if it was set to 1. Setting it to one means that at the ontological level O1 the SportsCar is an abstract class and as such we do not need to create instances of SportsCar in the O2 Level. Changing the potency at level O0 will automatically change the value in other levels.

  Moreover, Melanie implements the potency even to the attributes and their values with durability (how many instantiation steps an attribute can be handed over to instances) and mutability (how many instantiation steps the value can be changed).

4. Case Study

The pattern chosen for the case study allows the explicit modeling of types and their instances, where types are not static but can be added dynamically, so do their features. The example is the classical problem of modeling product types like books or CDs or Videos which need to be added to the system on-demand, as well as their instances. Think about it as a tentative to create an Amazon/E-Bay Database. Below is a simple MLM solution of this example from the (6) with M-relations and M-objects which Melanee does not support but would make the model more visible.

Multi level Modeling



By potency value x of attribute is meant that both mutability and durability of the attribute have the same value x.

1. **Product Catalog** Level 5, everything is an instance of Product

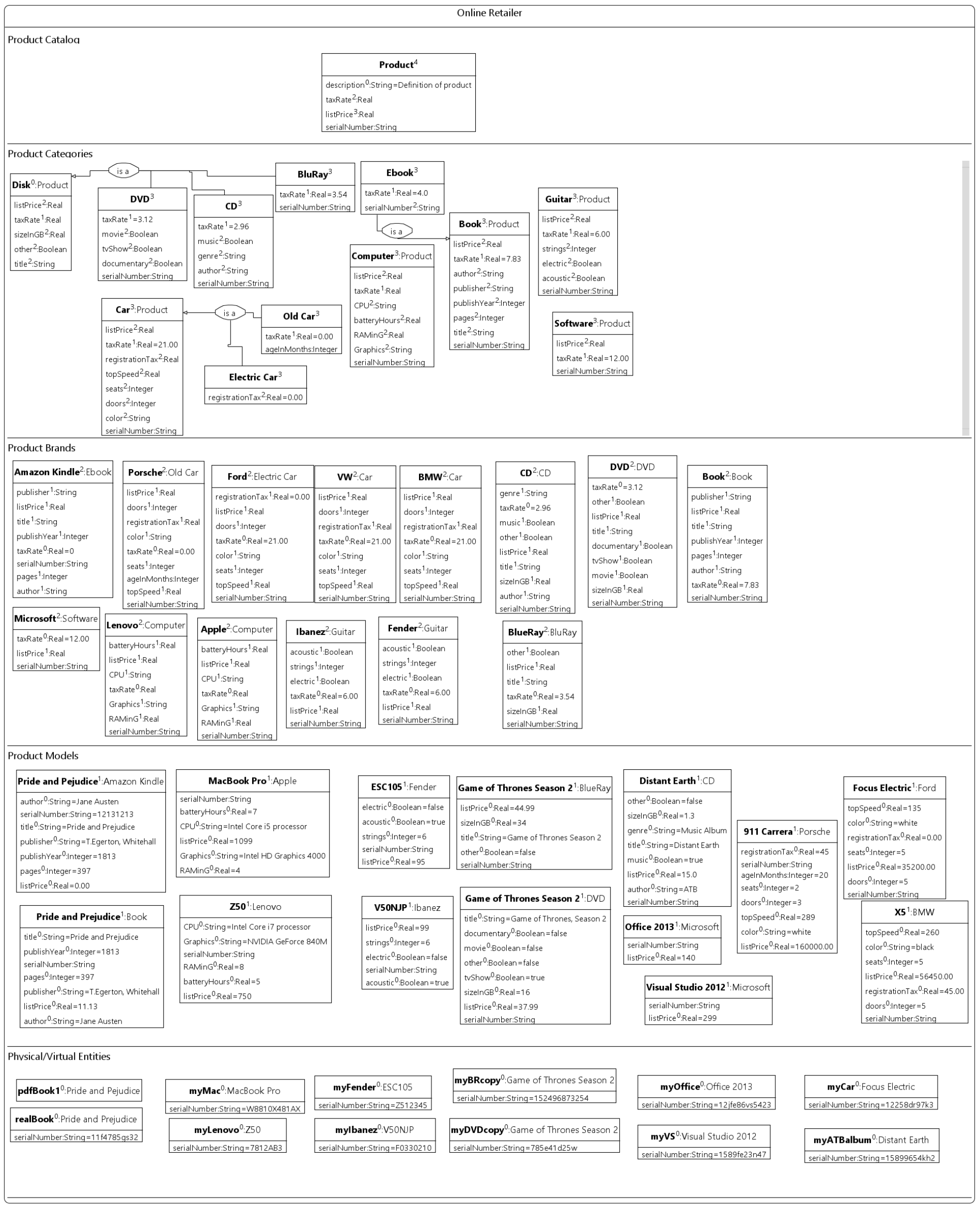

Online Retailer
Product Catalog
Product^{4}
description^{0}:String=Definition of product
taxRate^{2}:Real
listPrice^{3}:Real
serialNumber:String
Product Categories
is a
Disk^{0}:Product
DVD^{3}
CD^{3}
BluRay^{3}
Ebook^{3}
Book^{3}:Product
Computer^{3}:Product
Guitar^{3}:Product
Software^{3}:Product
Car^{3}:Product
Old Car^{3}
Electric Car^{3}
Product Brands
Amazon Kindle^{2}:Ebook
Porsche^{2}:Old Car
Ford^{2}:Electric Car
VW^{2}:Car
BMW^{2}:Car
CD^{2}:CD
DVD^{2}:DVD
Book^{2}:Book
Microsoft^{2}:Software
Lenovo^{2}:Computer
Apple^{2}:Computer
Ibanez^{2}:Guitar
Fender^{2}:Guitar
BlueRay^{2}:BluRay
Product Models
Pride and Pejudice^{1}:Amazon Kindle
MacBook Pro^{1}:Apple
ESC105^{1}:Fender
Game of Thrones Season 2^{1}:BlueRay
Distant Earth^{1}:CD
Focus Electric^{1}:Ford
911 Carrera^{1}:Porsche
Pride and Prejudice^{1}:Book
Z50^{1}:Lenovo
V50NJP^{1}:Ibanez
Game of Thrones Season 2^{1}:DVD
Office 2013^{1}:Microsoft
Visual Studio 2012^{1}:Microsoft
X5^{1}:BMW
Physical/Virtual Entities
pdfBook1^{0}:Pride and Pejudice
realBook^{0}:Pride and Prejudice
serialNumber:String=11f4785gs32
myMac^{0}:MacBook Pro
serialNumber:String=W8810X481AX
myLenovo^{0}:Z50
serialNumber:String=7812AB3
myFender^{0}:ESC105
serialNumber:String=Z512345
myIbanez^{0}:V50NJP
serialNumber:String=F0330210
myBRcopy^{0}:Game of Thrones Season 2
serialNumber:String=152496873254
myDVDcopy^{0}:Game of Thrones Season 2
serialNumber:String=785e41d25w
myOffice^{0}:Office 2013
serialNumber:String=12jfe86vs5423
myVS^{0}:Visual Studio 2012
serialNumber:String=1589fe23nh47
myCar^{0}:Focus Electric
serialNumber:String=12258dr97k3
myATBalbum^{0}:Distant Earth
serialNumber:String=15899654kh2

Figure 4: Multi level Modeling the case study with Melanee

- *Product* has the potency value of 4 because it needs to be initialized at all 5 level. It's attribute description has potency of 0 because it is not needed anywhere else. Taxrate 2 because it does not need to be accessed after the Brands level (because some products sold by Amazon for example are tax free). Listprice 3 because it need not to be changed at the real world level. Serial number is a unique value for each physical product so it is present until the last level.

2. **Product Categories**

    - *Disk* has the potency value of 0 because it needs not to be initialized at any other level since it is a general notion. It's attributes on the other hand, are inherited by CD, BluRay and DVD that need to be initialized in the next levels. Size in GB, Other Genre and title are present until the model level because different cd/dvd/bluray copies have the same title, size and info.
    - *DVD* Because the serial number has the same potency value as the instance it is not shown by Melanee.
    - *CD*
    - *BluRay*
    - *Book* has the properties inherited by EBook
    - *EBook* has the potency value of 2 for the attribute serialNumber because pdf or word copies of Ebooks don't have serial numbers. It is virtual property without any unique feature to identify it from the other copies.
    - *Car* is just normal cars one would buy and has to pay taxes for buying and registering
    - *Old Car* has the attribute taxRate overriden with 0 because appearantly in Belgium if you buy an old car of 6 months you don't need to pay that tax.
    - *Electric Car* has registration tax of 0 since it's a green car, environment friendly.
    - *Computer*
    - *Guitar*
    - *Software*

3. **Product Brands**
    - *Disk* is not in this level
    - *DVD* Because it doesn't make any sense for DVDs, CDs, Blurays and Books to have brands they just appear in here. This redundancy is one of the drawbacks of Melanee or MLM in general which is solved by M-objects and relationships.
    - *CD*
    - *BluRay*
    - *Book*
    - *EBook* Amazon Kindle book with tax 0, unlike other ebook brands which would have it 4.0 as described by the upper level Ebook instance.
    - *Car* BMW and Volks Wagen
    - *Old Car* Porche
    - *Electric Car* Ford
    - *Computer* Lenovo and Apple
    - *Guitar* Ibanez and Fender
    - *Software* Microsoft
4. **Product Models**
    - *Disk* is not in this level
    - *DVD* A DVD of the Second Season of Game of Thrones
    - *CD* A CD of the Distan Earth album of ATB
    - *BluRay* A BluRay of the Second Season of Game of Thrones
    - *Book* the hardcopy book of pride and prejudice
    - *EBook* Amazon Kindle ebook Pride and Prejudice
    - *Car* X5-BMW
    - *Old Car* 911 Carrera-Porche
    - *Electric Car* Focus Electric-Ford
    - *Computer* Z50-Lenovo and Mac Book Pro-Apple
    - *Guitar* a guitar from each of the Ibanez and Fender brands

- *Software* Visual Studio 2012-Microsoft and Office 2013

5. **Physical/Virtual Entities** The user has the following items:
   - *DVD* A DVD of the Second Season of Game of Thrones
   - *CD* A CD of the Distan Earth album of ATB
   - *BluRay* A BluRay of the Second Season of Game of Thrones
   - *Book* a hardcopy book of pride and prejudice
   - *EBook* Amazon Kindle ebook Pride and Prejudice
   - *Electric Car* a Focus Electric-Ford car
   - *Computer* a Z50-Lenovo and a Mac Book Pro-Apple
   - *Guitar* a guitar from each of the Ibanez and Fender brands
   - *Software* an Office 2013 Package with serial number and hopefully with a license key

After the case study is explained, a full picture is shown at Figure 4, a short tutorial of how to implement it in Melanee follows in the next section.

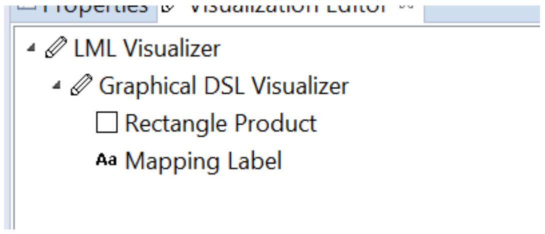
5. Implementation in Melanie

5.1. Install

Download the latest Melanee + Eclipse package at (1). Install it and run it. As explained above Melanee comes in packages. At the install moment only the core of Melanee is installed. Go to Help - Install new Software... and add more plug-ins to Melanee as shown here (7). Install both Graphical DSL and OCL and restart Melanee after each install. What it is not shown in the tutorial is that you have to go to Windows-Preferences-Melanee-Workbench and press apply to finish the installation. Not doing this will result in errors.

5.2. LML

One can start modeling using the Level agnostic Modeling Language as shown in this tutorial (8). There is a full diagram of the case study at Figure 4. Note that at this step only "classes" are created with attributes as explained at the case study section.

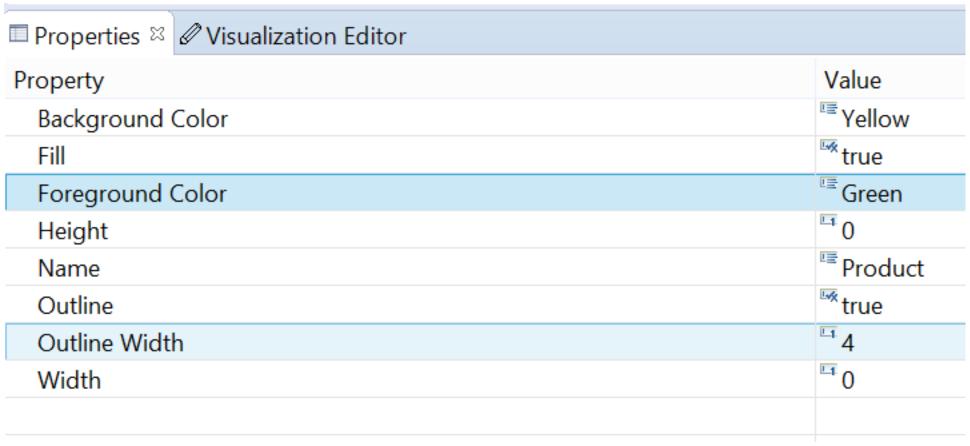

LML Visualizer

Graphical DSL Visualizer

Rectangle Product

Mapping Label

Figure 5: Product Visualization


Properties · Visualization Editor

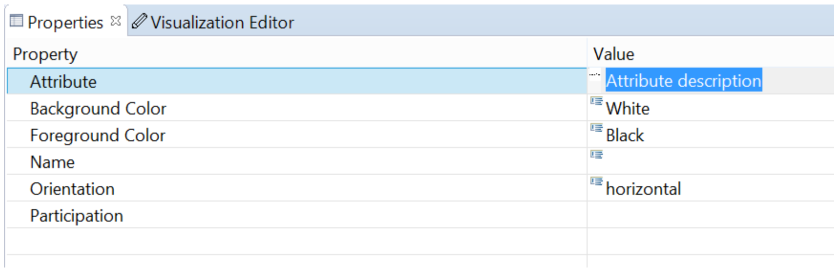
Property	Value
Background Color	Yellow
Fill	true
Foreground Color	Green
Height	0
Name	Product
Outline	true
Outline Width	4
Width	0

Figure 6: Rectangle Properties

5.3. G-DSL

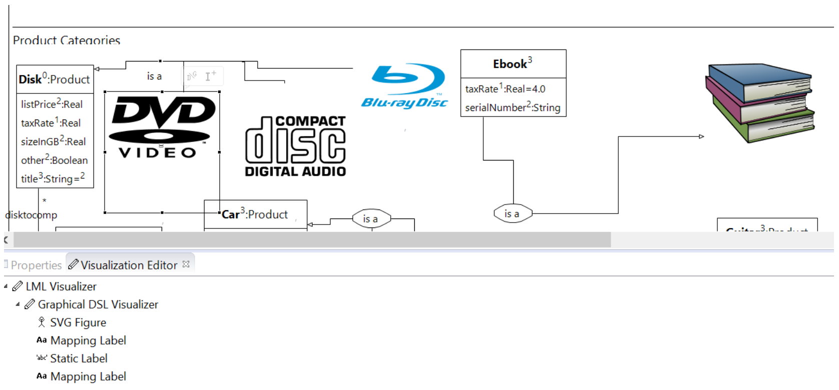
Graphical Domain Specific Language can be used to to give domain specific representation to the elements of the model. There is again a full diagram at the end of the paper with the worked case study. ((9) is the page with the tutorial when it can be learned how to use the Graphical DSL.) The Product element at the top level is created by using the default shapes of the tool. It's just a filled colorful rectangle with a thick border. Look at the pictures Figure 5 - Figure 7. Note that there is an other element looking like the product at a level lower. That is the software and since the Graphical representation is transferable between levels the software and any element without overriden visualization would look as such.

DVD, CD, BluRay and Book are declared at the categories level because brands do not matter for those. See Figure 8 with the visualization of DVD. It is composed of a SVG image, two mapping labels for title and serial number and a static label with a comma to divide the two mappings above. The SVG



Property	Value
Attribute	Attribute description
Background Color	White
Foreground Color	Black
Name	
Orientation	horizontal
Participation	

Figure 7: Mapper Properties



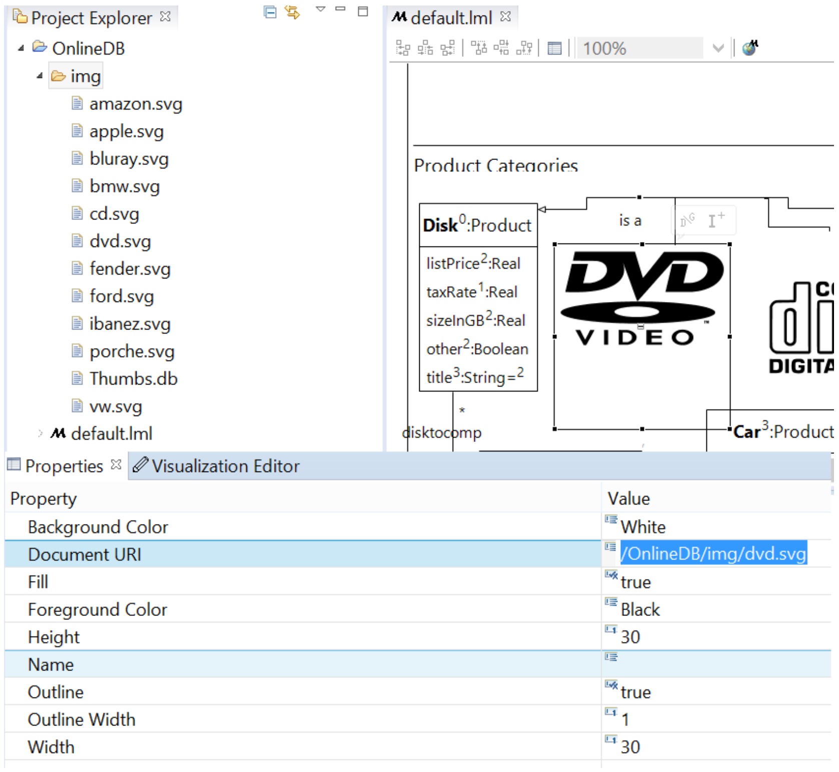
```
LML Visualizer
  Graphical DSL Visualizer
    SVG Figure
    Mapping Label
    Static Label
    Mapping Label
```

Figure 8: DVD Visualization

file's properties look as shown in Figure 9. Note that at this level a play on connection is created between disk and computer. This in order to show that graphical visualization can be added even at links/connections (Figure Figure 10). A blue arrow will appear in the next levels of the full Graphical diagram.

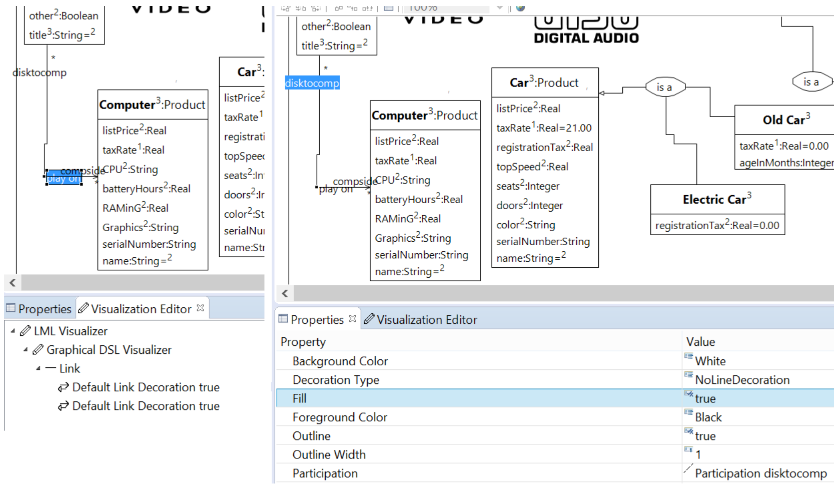
As done above SVG images and mapping and static labels were added at the other elements. Most of them have the icon of the brand. Special cases such as Microsoft Products have their own icon so they are represented by that. An attribute name is added to all the categories that do not have the attribute title, with mutability potency lower then the durability because this attribute is used until the last level to describe graphically with a mapping label the product. The elements with attribute title were changed so that the title can be durable until the last level and it can be shown at the graphical visualization.

- *Amazon Kindle* brand image and mapping for title of the book



Property	Value
Background Color	White
Document URI	/OnlineDB/img/dvd.svg
Fill	true
Foreground Color	Black
Height	30
Name	
Outline	true
Outline Width	1
Width	30

Figure 9: SVG Properties



Property	Value
Background Color	White
Decoration Type	NoLineDecoration
Fill	true
Foreground Color	Black
Outline	true
Outline Width	1
Participation	Participation disktocomp

Figure 10: Link and Default Link Decoration Properties

- *X5-BMW* brand image and mapping for name of car and serial number
- *911 Carrera-Porche* brand image and mapping for name of car and serial number
- *Focus Electric-Ford* brand image and mapping for name of car and serial number
- *Mac Book Pro-Apple* brand image and mapping for name of computer and serial number
- *Ibanez and Fender brands* brand image and mapping for name of guitar and serial number
- *Visual Studio 2012-Microsoft and Office 2013* model (specific software) image and mapping for name of software (because of the version differences)

5.4. T-DSL

See (10) for a tutorial on how to use the Textual Domain Specific Language modeling with Graphical DSL. Remember to install and apply changes as done before.

I was not able to make this part work. I was told it has bugs that need fixes but I was not able to make even the non buggy part work. ("Textual DSLs, however, had some bugs introduced through some updates to the Melanee-metamodel. It works fine for putting out text and changing attribute values. The instantiation feature however needs some fixes applied.") It is not possible for me to add children to the TDSL when only there is only textual visualizer. If I first create the graphical one as in Figure 12 then the graphical part is rendered, while nothing is visible from the textual visualizer. The textual editor is not editable, I can't do anything with it. Removing the graphical dsl visualizer is not possible if dsl rendering is true. When I do delete it and change the dsl visualizer at properties as in Figure 13 nothing happens.

5.5. Reasoning

Install and apply the changes. After setting The last Physical/Virtual level as a reasoning object and performing a classification check it shows

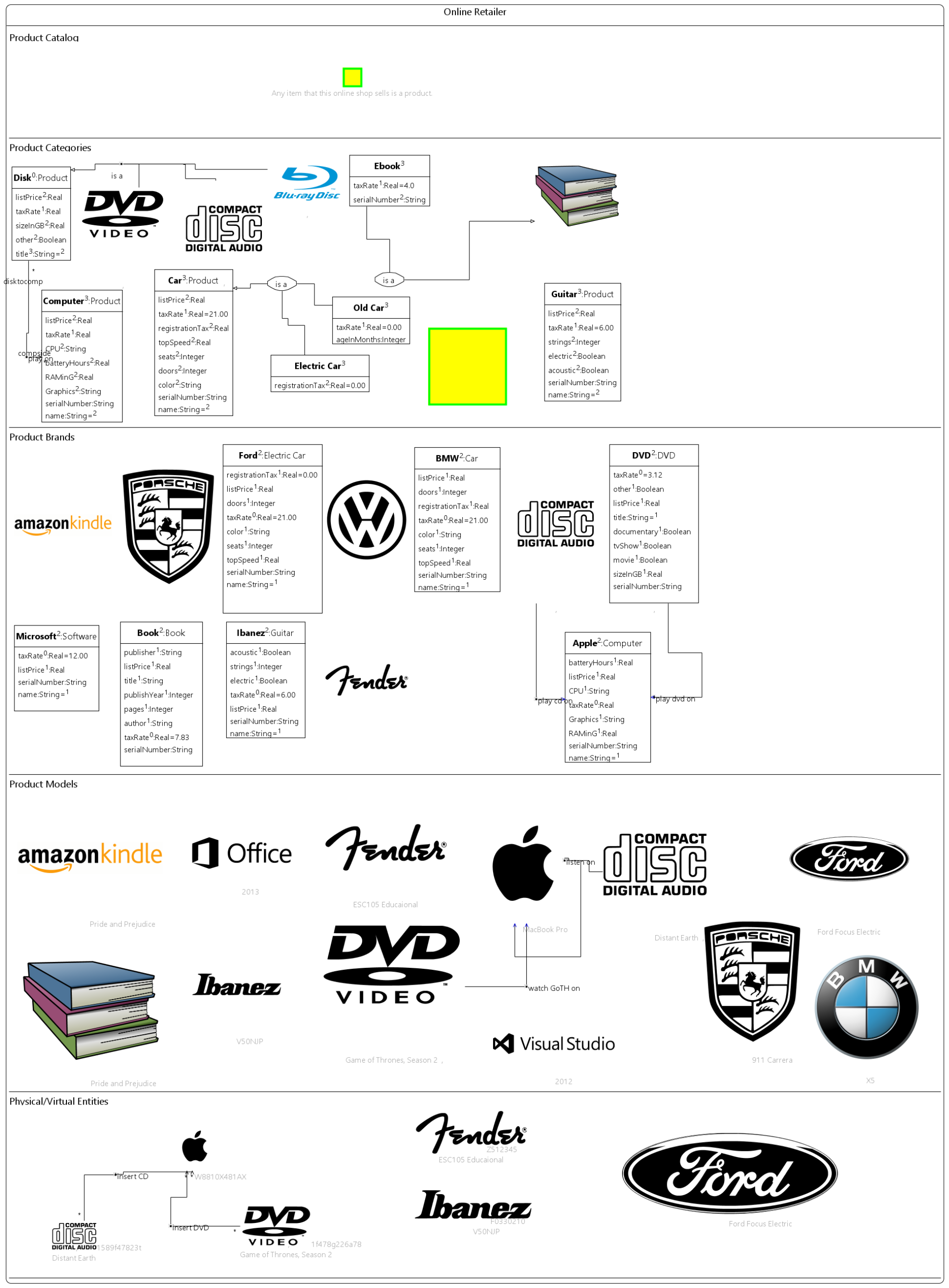


Figure 11: G-DSL of full Case Study

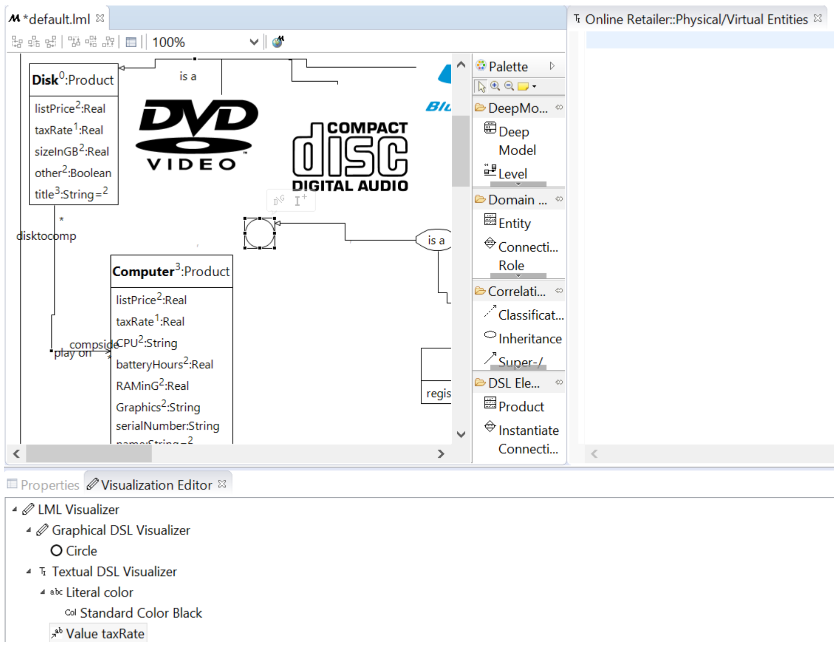


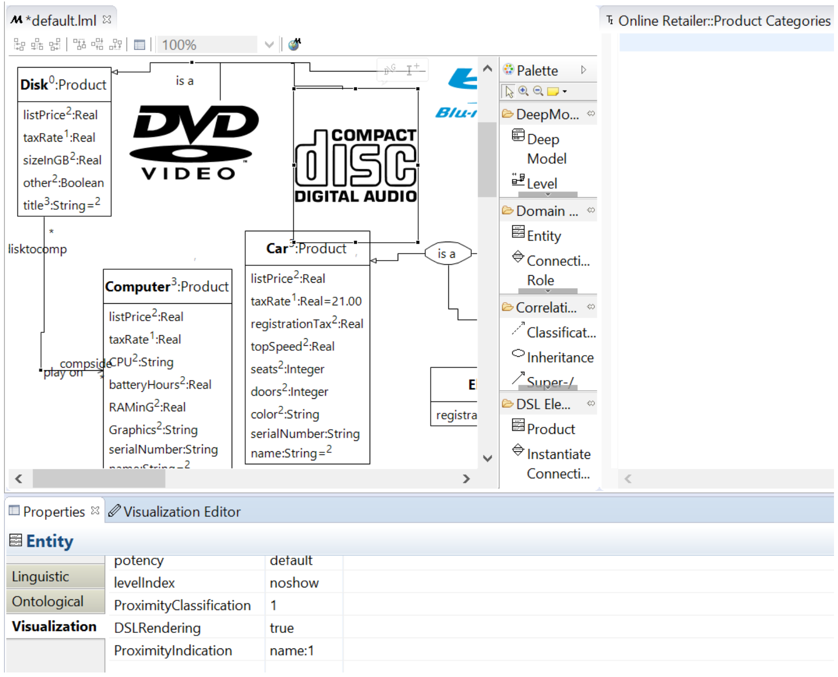
Figure 12: T+G DSL Visualizer

the Mac instance of the user has some classification error, Figure 14. This is because it was created before some new changes were made to the higher level definition of Mac. Removing this element and creating a new one results in Figure 15. For more about reasoning check the tutorial at (11).

5.6. Deep OCL

OCL was installed at the beginning together with GDSL. See tutorial (13) for more Deep OCL information. Figure 16 shows where to access the LML Deep OCL after opening the console at Window - Show View - Other... - General - Console. It also shows a basic example of how you can use Deep OCL on the model created so far. It is important to change all the spaces into "_" since the parser of the OCL does not recognize the spaces and gives out errors.

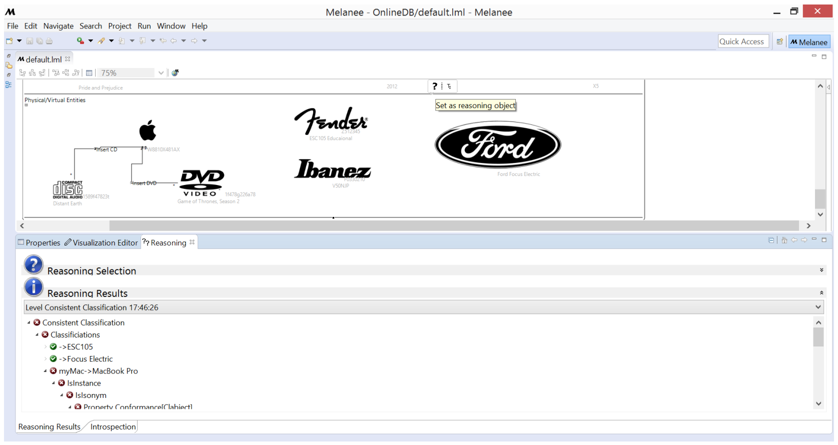
Even linguistic information is accessed from with feature. For example, click on the ford car of the user and write to execute "self._l_.feature" on the OCL console. The Results are: Attribute name and Attribute serialNumber. You can also travel through links/connections: self.insert.myMac.serialNumber gives the Results 'W8810X481AX'. Also travel between levels is possible,



Entity

	potency	default
Linguistic	levelIndex	noshow
Ontological	ProximityClassification	1
Visualization	DSLRendering	true
	ProximityIndication	name:1

Figure 13: T DSL Visualizer



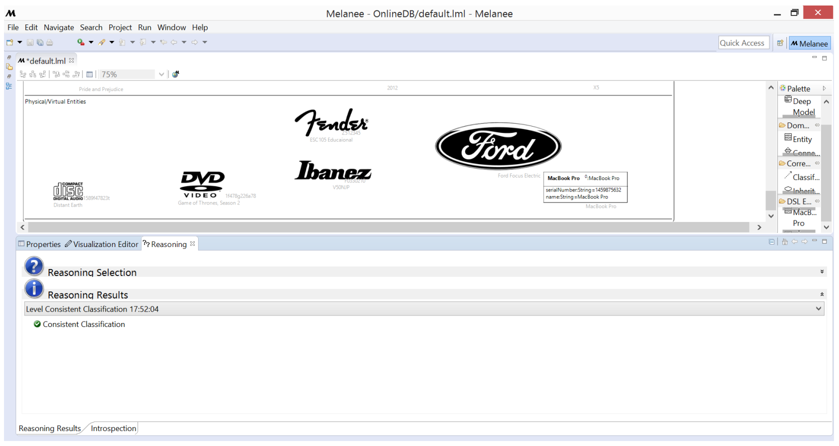
Reasoning Selection

Reasoning Results

Level Consistent Classification 17:46:26

```
Consistent Classification
  Classificiations
    ->ESC105
    ->Focus Electric
    myMac->MacBook Pro
      IsInstance
        IsIsonym
          Property Conformance[Clabject]
```

Figure 14: Error Mac

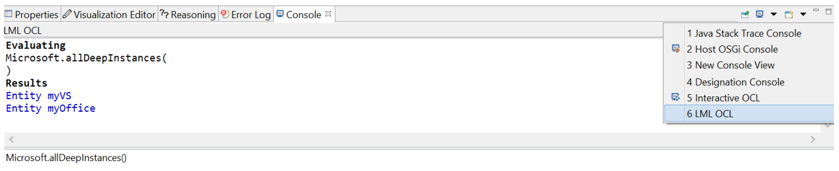


MacBook Pro :MacBook Pro
serialNumber:String = 1459875632
name:String = MacBook Pro

Level Consistent Classification 17:52:04

Consistent Classification

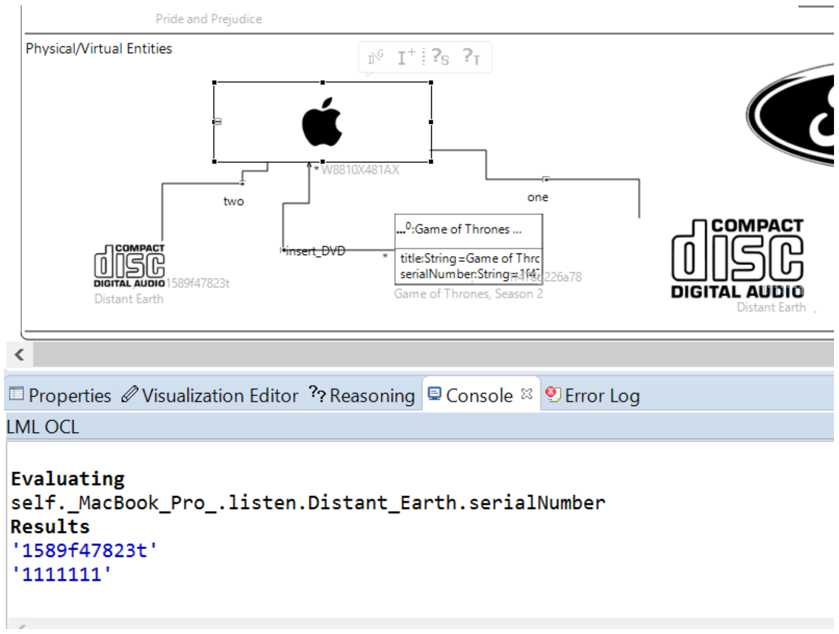
Figure 15: No Errors



```
Evaluating
Microsoft.allDeepInstances(
)
Results
Entity myVS
Entity myOffice
```

Microsoft.allDeepInstances()

Figure 16: Deep OCL Console



```
Evaluating
self._MacBook_Pro_.listen.Distant_Earth.serialNumber
Results
'1589f47823t'
'1111111'
```

Figure 17: Deep OCL through physical and product model levels to find the serial number of all cds this computer can open.

check Figure 17. You can go on higher levels the same way, for example self._MacBook_Pro_._Apple_._Computer_.play_on gives all instances of play_on (Connection insert_DVD, Connection one, Connection two). See Figure 18 for the changed names.

5.7. Model

At the beginning of this paper where the concepts of Multi Level Modeling are explained the Model looks very simple. A basic 3 level model with ProductType, Video and 2001. Because 2001 is just a representation of the real world a new level is introduced. One that includes a representation of the real world object with a unique serial id. More elements were introduced at the model to describe features of Multi Level Modeling and Melanee and the model has a new level, the brand level. Morover, during Graphical DSL links were introduced between computer and disks, indicating that dvds, cds and bluray can be played on a computer. The model goes through a transition as shown in the following pictures: Figure 1 to Figure 4 to Figure 11 to Figure 18.

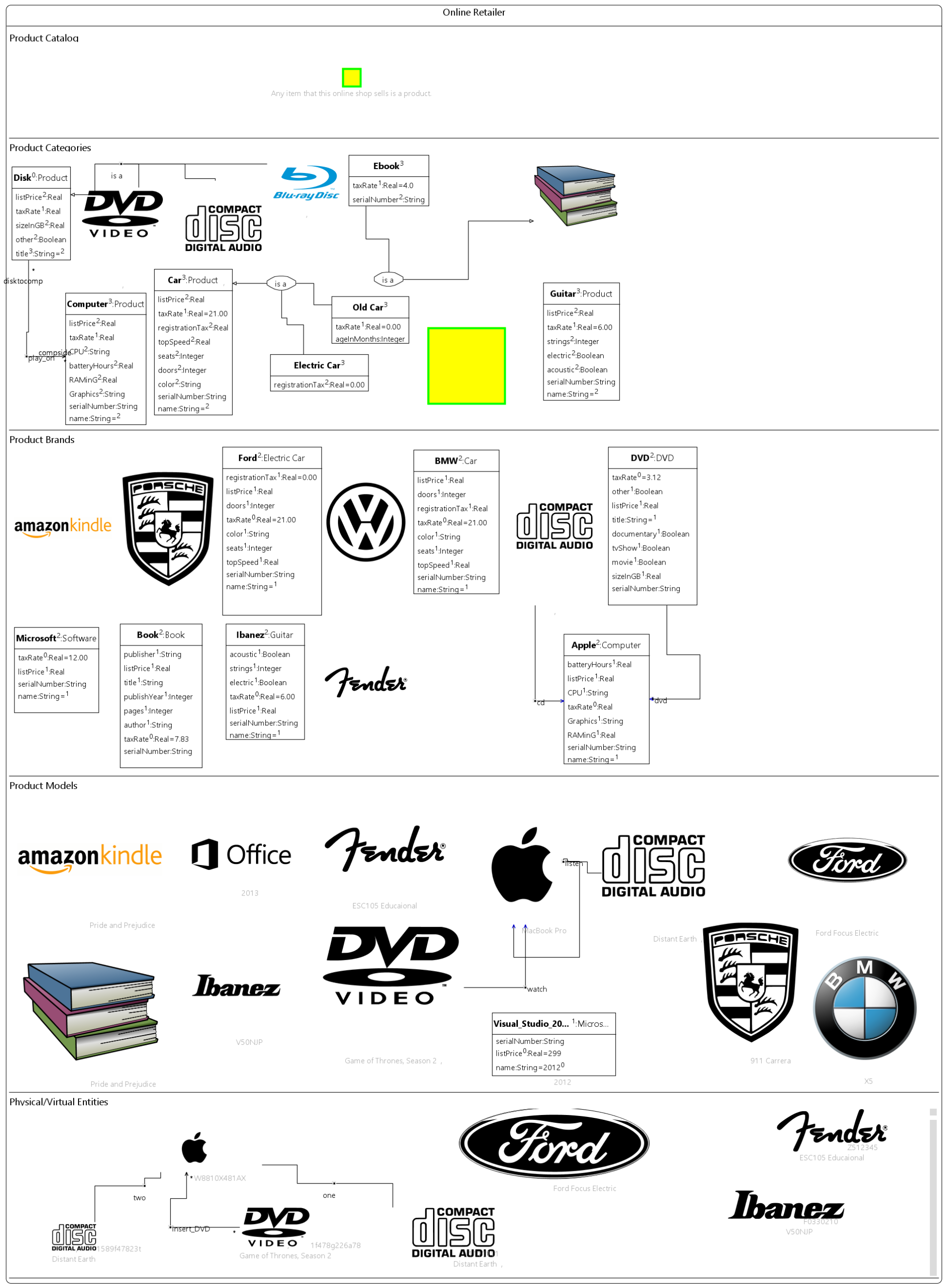


Figure 18: Final Model after some modifications because of Deep OCL and Reasoning

6. Future Work

It would be interesting to try the Textual DSL Visualizer of Melanee. Since the integration between Graphical and Textual DSL it's one of it's strongest positive points as a modeling tool. Continuing with the model, more attributes and methods should be implemented to take it closer to the real world objects. I could fo example create a top level entity Person. This person entity can be used to instantiate entities such as author or artist and create relationships with books or cds. (12) is a screencast that shows how Melanee and Deep ATL are used to realize Orthographic Software Modeling in a tool called Naomi. It would be interesting to try it. The same tool is used to generate movement to a robotic animation at the following video (14).

7. Conclusions

7.1. Melanee as an MLM tool

As you might have noticed at the last full model picture. The BluRay instance is missing. If you omit one element in one level you can not instantiate it on the lower levels in Melanee. It would be helpful if this was possible since instances of DVD, CD and Book are redundant in some levels. This would also be helpful if applicable on attributes and methods for the same reason. A solution is discussed in (6). Other then that Melanee is a very good tool implementing most of MLM concepts.

7.2. Melanee and AToMPM

One other modeling tool I have worked with is AToMPM. Althought AToMPM is not a MLM tool it has some features where it can be compared with Melanee. They're both graphical and textual Domain Specific Language Modeling Tools, but unlike Melanee that supports full textual DSL modeling AToMPM has it incorporated with G-DSL. But it was not possible while working for this paper to prove the full textual feature so this might not count as difference for some. AToMPM is a web based tool and Melanee is a stand alone application (ok, a plug-in on Eclipse). Both tools are open source, which means that they are still under constant change and development. Also anyone is free to modify, fix or suggest improvments. Error checking is difficult on both tools unless you have worked a long time with them. Melanee components, probably from the fact that it is a MLM tool offer less manual work (more compilation steps, file creation/naming in AToMPM) which makes it a more comfortable tool to work with. Both tools use some kind of xml files underneath to save the files. AToMPM is more comfortable while using pictures, in Melanee the only supported picture type is .svg and it is a little difficult to find free converters ((15), this one I found offers only black and white converting, there is an other where you can register and offers 2 free conversions, (16)). The graphical editor of Melanee is better, AToMPM's editor makes it difficult to work with the z-index of the elements and I personally had problems with that. But AToMPM has better examples to help beginners start working with the tool and also offers Transformations which I don't know if and how are used in Melanee, I only know they're based on EMF-transformations.

8. References

[1] Melanie HomePage *www.melanee.org*

[2] Colin Atkinson, Ralph Gerbig *Harmonizing Textual and Graphical Visualizations of Domain Specic Models*. 2013.

[3] Colin Atkinson, Ralph Gerbig *Melanie Multi-level Modeling and Ontology Engineering Environment*. 2000.

[4] Colin Atkinson, Thoman Kühne *Rearchitecting the UML Infrastructure*. 2002.

[5] Juan De Lara, Esther Guerra, Jesus Sanchez Cuadrado *When and How to Use Multi-Level Modelling.*

[6] Bernd Neumayr Katharina Grün Michael Schrefl *Multi-Level Domain Modeling with M-Objects and M-Relationships.*

[7] Installation Tutorial *http://melanee.informatik.uni-mannheim.de/melanee/screencasts/Download-Plugin-SWF/Download-Plugin-SWF.swf.*

[8] Create Digram Tutorial *http://melanee.informatik.uni-mannheim.de/melanee/screencasts/CreatingDiagram-SWF/CreatingDiagram.swf.*

[9] Create Graphical Model Tutorial *http://melanee.informatik.uni-mannheim.de/melanee/screencasts/GraphicalDSL-SWF/GraphicalDSL.swf.*

[10] Create Textual Model Tutorial *http://melanee.informatik.uni-mannheim.de/melanee/screencasts/TextualDSL/screencast.swf.*

[11] Reasoning Tutorial *http://melanee.informatik.uni-mannheim.de/melanee/screencasts/Reasoning/reasoning.html.*

[12] Deep ATL Tutorial *http://melanee.informatik.uni-mannheim.de/melanee/screencasts/DeepATL/ICMT2012.html.*

[13] Deep OCL Tutorial *http://melanee.informatik.uni-mannheim.de/melanee/screencasts/OCL-SWF/OCL.swf.*

[14] Deep Robot Model *http://melanee.informatik.uni-mannheim.de/melanee/screencasts/MORSE2014/DeepRobotModel.mp4.*

[15] Converter *http://image.online-convert.com/convert-to-svg.*

[16] Converter with colors only 2 times *http://vectormagic.com/.*

[17] AToMPM webpage *http://syriani.cs.ua.edu/atompm/atompm.htm.*