

---

# **SOFTWARE REQUIREMENTS SPECIFICATION**

**for**

**XModeler**

**Release 1.0**

**Version 1.0 approved**

---

# Contents

<b>Contents</b>	<b>2</b>
<b>Revision History</b>	<b>4</b>
<b>1 Requirements</b>	<b>5</b>
1.1 Web Site . . . . .	5
1.1.1 Content . . . . .	6
1.1.2 Design . . . . .	7
1.2 Changes on Intrinsic . . . . .	8
1.2.1 Add . . . . .	9
1.2.2 Delete . . . . .	10
1.2.3 Change Level of Attribute . . . . .	11
1.2.4 Change Level Of Class . . . . .	12
1.2.5 Checks . . . . .	13
1.2.6 Addons-Essen Existing Implementation . . . . .	13
1.2.6.1 MetaAdaptor new() . . . . .	13
1.3 MeMo-Language . . . . .	17
1.4 Kernel . . . . .	17
1.5 X-Modeler . . . . .	17
1.5.1 Diagrams . . . . .	17
1.5.1.1 Multi-Level Diagram . . . . .	17
1.5.2 Forms . . . . .	17
1.5.2.1 FormsClient . . . . .	17
<b>2 Specifications</b>	<b>18</b>
2.1 Web Site . . . . .	18
2.1.1 Design . . . . .	18
2.2 XMF-Language . . . . .	18
2.3 Kernel . . . . .	18
2.4 X-Modeler . . . . .	18
2.4.1 Diagrams . . . . .	18
2.4.1.1 Multi-Level Diagram . . . . .	18
<b>3 Other</b>	<b>19</b>
3.1 git . . . . .	19
3.1.1 fetch . . . . .	20
3.1.2 pull . . . . .	20
3.1.3 commit . . . . .	20
3.1.4 push . . . . .	20
3.1.5 conflicts . . . . .	20
3.1.6 branches . . . . .	20

3.1.7	revert . . . . .	20
<b>4</b>	<b>Example</b>	<b>21</b>
4.0.8	Multilevel Draft Essen . . . . .	24
4.0.9	next Try . . . . .	24
4.0.10	next Try II . . . . .	24
4.0.10.1	numbering up . . . . .	25
4.0.10.2	numbering down . . . . .	25
4.0.10.3	allowing multi-level jumps . . . . .	25
4.0.10.4	Finding a unified numbering scheme . . . . .	26
<b>5</b>	<b>Literature</b>	<b>27</b>
5.1	Integrating Language Engineering [1]... . . . .	27
<b>6</b>	<b>Weekly Reports</b>	<b>28</b>
6.15	2015 . . . . .	28
6.15.37	7 Sep 2015 - 11 Sep 2015 . . . . .	28
6.15.38	14 Sep 2015 - 18 Sep 2015 . . . . .	28
6.15.39	21 Sep 2015 - 25 Sep 2015 . . . . .	28
6.15.40	28 Sep 2015 - 2 Oct 2015 . . . . .	28
6.15.41	5 Oct 2015 - 9 Oct 2015 . . . . .	28
6.15.42	12 Oct 2015 - 16 Oct 2015 . . . . .	29
6.15.43	19 Oct 2015 - 23 Oct 2015 . . . . .	29
6.15.44	26 Oct 2015 - 30 Oct 2015 . . . . .	29
6.15.45	ToDo . . . . .	29
	<b>Bibliography</b>	<b>30</b>

## Revision History

# 1 Requirements

Req-1.0.0.0.1	Some requirements may be optional. <a href="#">Spec-2.0.2.1.1</a>
---------------	---

## 1.1 Web Site

Req-1.1.0.0.1	The website is available online.
	A slightly different version is available in XModeler.
OR	The online version is accessed from XModeler, but unavailable when offline.
OR	XModeler tries to access the online version, but keeps a offline version as backup.
	The online version cannot use the XModeler's hijacking mode for references.
Req-1.1.0.0.2	The website will have several parts: The online home page, the in-tool home page, online content, in-tool content, online content which is available offline. In-tool content can access online content. Online content cannot access in-tool content.

### 1.1.1 Content

Req-1.1.1.0.1	<ul style="list-style-type: none"> <li>• XModeler <ul style="list-style-type: none"> <li>– What is XModeler?</li> <li>– Getting started <ul style="list-style-type: none"> <li>* Install guide</li> <li>* Tutorial</li> </ul> </li> <li>– Help <ul style="list-style-type: none"> <li>* API</li> <li>* Language concepts</li> <li>* Snippets <sup>X</sup></li> <li>* Bluebook <sup>X</sup></li> </ul> </li> </ul> </li> <li>• Literature <ul style="list-style-type: none"> <li>– Articles, Papers <ul style="list-style-type: none"> <li>* Our Articles, Papers</li> <li>* Conferences</li> <li>* List of others' literature</li> </ul> </li> <li>– Lecture Slides and other material for students <sup>O</sup></li> </ul> </li> <li>• External Links <ul style="list-style-type: none"> <li>– git</li> <li>– shu</li> <li>– ude</li> </ul> </li> <li>• Contact <ul style="list-style-type: none"> <li>– Who are we</li> <li>– How to contact us</li> <li>– Impressum</li> </ul> </li> </ul> <p><sup>X</sup>: Not available online  <sup>O</sup>: Only available online</p> <p>Is any material we use copyrighted? Are icons and logos in XModeler our work? If they are from any "free" source, are there any conditions on them?  Do we need any legal advice? Disclaimers?</p>
Req-1.1.1.0.2	The website should include a install guide
Req-1.1.1.0.2a	The install guide should be split into several parts, after each (and before first) some checks to see if step worked.
Req-1.1.1.0.2b	The install guide should contain common problems.
Req-1.1.1.0.3	The website should include a tutorial.
Req-1.1.1.0.4	The website should include a "API", explaining the basic data types and classes.
Req-1.1.1.0.5	The website should include an explanation of basic language concepts. (grammar, operators)
Req-1.1.1.0.6	The website should include snippets. (Small examples on a specific topic, like how to read from a file)
Req-1.1.1.0.7	The website should include a list of literature, with links if possible.
Req-1.1.1.0.8	The website should include a link to git, Maybe direct links to the Issues?
Req-1.1.1.0.9	The website should include information about contact. <i>Whom was it made by? How do I contact them, report a bug?...</i>

### 1.1.2 Design

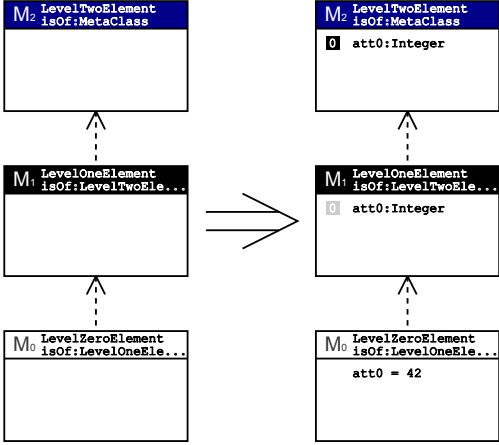
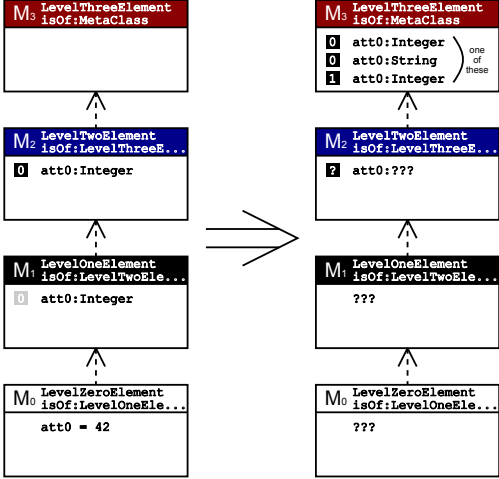
Req-1.1.2.0.1	The design should have a primary and a secondary colour. These, together with gray-scale colours should be used predominantly. <a href="#">Spec-2.1.1.0.1</a> , <a href="#">Spec-2.1.1.0.2</a>
Req-1.1.2.0.2	The website is split vertically: Title, Links, local Links, Content

## 1.2 Changes on Intrinsic

Req-1.2.0.0.1	The user may want the changes to affect all existing instances.
Req-1.2.0.0.2	The user may want the changes to affect all future instances.
Req-1.2.0.0.3	The user may want any alerts to be suppressed until he finishes his changes.



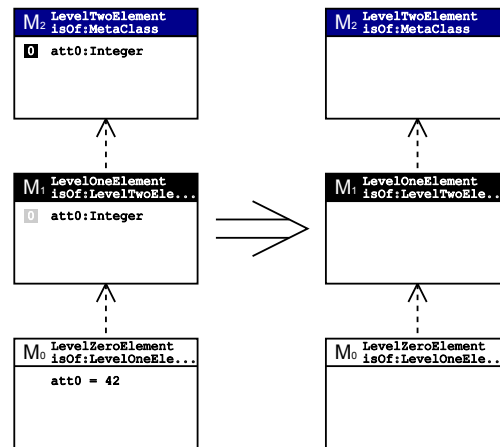
## 1.2.1 Add

Req-1.2.1.0.1	<p><b>The user adds a new intrinsic attribute.</b> Every instance on the appropriate level should get a slot for the Attribute. No slot will be provided before the given level is reached. <i>Isn't there a constraint that an element must have all slots for which the of() has an attribute?</i> Unless a default value is given, all instances have to be supplied with new data.</p> 
Req-1.2.1.0.2	<p><b>The user adds a new intrinsic attribute causing a name clash on lower levels.</b> This means that an attribute on a lower level would — intended or not — override or hide the newly created attribute.</p> 
Req-1.2.1.0.3	<p><b>The user adds a new intrinsic attribute causing a name clash on higher levels.</b> Basically the same possible conflicts as above, but a higher degree of intention can be assumed.</p>

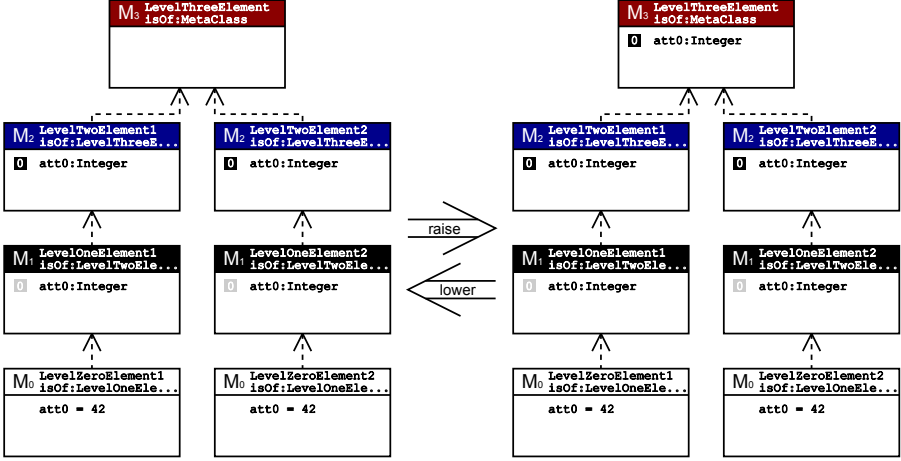
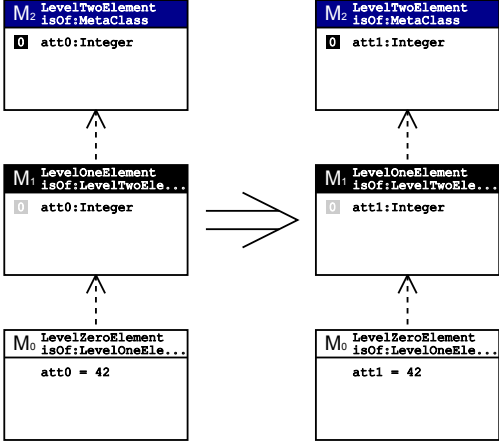
## 1.2.2 Delete

Req-1.2.2.0.1

**The user deletes an intrinsic attribute.** All usages should be removed as well. Otherwise they would lose their connection to their original definition.



### 1.2.3 Change Level of Attribute

Req-1.2.3.0.1	<p>The user adds a new intrinsic attribute intentionally causing a name clash on lower levels, therefore raising the level of the attribute. The intrinsic attributes in the lower classes must be removed. Opposite of <a href="#">Req-1.2.3.0.2</a></p> 
Req-1.2.3.0.2	<p>The user deletes an intrinsic attribute in a class and adds it to all that class' instances. Opposite of <a href="#">Req-1.2.3.0.1</a>. Would mean adding redundancy and is therefore unlikely to happen.</p>
Req-1.2.3.0.3	<p>The user renames an existing attribute. In addition to the potential problems of adding one attribute and deleting another one, this case is different as the attribute should remain unchanged apart from it's name. This makes this scenario a genuine one, that means it is not a combination of other scenarios.</p> 

#### 1.2.4 Change Level Of Class

Req-1.2.4.0.1	The user changes the level of a class: All instances have to be of lower level than the class. All intrinsic attributes have to be of lower level than the class. All classes above the class must comply or be changed as well. No gap must be left between levels.
---------------	--

### 1.2.5 Checks

- An Attribute, which is intrinsic, can only be in (or added to) Objects, which are intrinsic and are of Class.
- An Attribute, which is not intrinsic, can only be in (or added to) Objects, which are of Class.
- *An Object with isIntrinsic=false has no Attributes with isIntrinsic=true*
- *An Object which is not of Class has no Attributes*
- An Object which is intrinsic on level 2 or more must extend Class/MetaClass
- An Object which is intrinsic on level 1 must be of a something extending Class/MetaClass but must not extend Class/MetaClass
- An Object which is intrinsic on level 0 must neither be of a something extending Class/MetaClass nor extend Class/MetaClass
- The level of an Object that is intrinsic must be 1 less the of's level, unless the of is not intrinsic.
- An Attribute, which is not intrinsic, of an Object, which is of Class (implied by a previous check), must have a slot at all instances.
- *All instances of an Object, which is of Class (implied by a previous check), must have a slot for every of's non-intrinsic Attribute.*
- An Attribute, which is intrinsic, of an Object, which is of Class, must have a slot at all instances (or instances' instances...) which are on the given level.

### 1.2.6 Addons-Essen Existing Implementation

#### 1.2.6.1 MetaAdaptor new()

- 1: A class with isIntrinsic=false creates an instance, where isIntrinsic is false as well. Normal behaviour.
- 2: A class with isIntrinsic=false creates an instance, where isIntrinsic is true. The instance needs a level, which has to be provided in some way. This class must extend Class/MetaClass to be on levels 2 or higher
- 3: A class with isIntrinsic=true creates an instance, the level is decreased. If the class is higher than level 2(=the instance is at least at level 2) it must extend Class/MetaClass

If case 2 would provide the level by argument, then the function would be callable on level-aware objects as well. *An object of level 3 instantiating to level 5?* The newToLevel must check and throw an error in this case. Alternatively it could be unavailable by splitting classes which would be a lot of extra work and would eventually resulting in throwing an error anyway...

**MetaAdaptor** is of **Class**. Shouldn't it be **Adaptor**?

```
= @Operation new():Object
```

o is created by the Kernel function instead of super class Class' new(). To prevent the creation of slots, Class' new() function is copied and modified (+,-,=)

A are all Attributes, intrinsicA is an empty set

```
=      let o = Kernel_mkObj();
=      A = self.allAttributes();
+      intrinsicA = Set{}
```

The of link is set.

A loop, aimed at emptying A (all Attributes) is started

```
=      in Kernel_setOf(o,self);
=      @While not A->isEmpty do
=      let a = A->sel
```

for each attribute: if it is intrinsic, and the level too low, then it is meant to be skipped in creating slots. It is put in the set "intrinsicA". **For a new unified constructor we can assume that no attribute is intrinsic on non-intrinsics classes, so the exclusion does nothing, leaving the new set empty.**

```
+      in if a.isIntrinsic and a.instLevel < o.of().level - 1
+      then
+      intrinsicA := intrinsicA.including(a)
+      else
```

Otherwise, if the Attribute is to be instantiated. Then it is added, if available by the init function, as Class would do

In both cases it is removed from the list A.

```
=      if a.init <> null
=      then
=      Kernel_addAtt(o,a.name,a.init.invoke(o,Seq{}))
=      else
=      Kernel_addAtt(o,a.name,a.type.default())
+      end
=      end;
=      A := A->excluding(a)
=      end
=      end;
```

Now, the attributes are done. The parents are set. If it does not inherit from Classifier it returns false? It originally returned o instead. **Both seem to be pointless, so the else should be dropped altogether.** After that the object/class is being initialized.

```
=      if self.inheritsFrom(Classifier)
=      then
=      o.parents := o.defaultParents()
=      else
+      false
-      o
=      end;
=      o.init();
```

The following part is used for case 2, creating a level-aware class form a level-agnostic, and for case 3, reducing the level on level-aware classes. Case 1 would simply finish here (except for two ends). **For a new unified constructor, case 2 would not go through a no-argument function. Therefor a new function is needed.** It checks if of() is MetaAdaptor, in which case a dialogue will open and ask for a level. MetaClass will then be a parent. This should imply a level of at least 2, which is not checked here.

The level input by dialogue must be removed. A newToLevel(level:Integer) should be the replacement for these cases.

```
+      if self.of() = Extensions::MetaAdaptor
+      then
+          o.level := xmf.getInteger("Metaebene","Auf welcher Metaebene soll
+ sich diese Entitt befinden?",3);
+          o.addParent(FMML::MetaClass)
+      else
```

Otherwise (if of() is not MetaAdaptor) this is an instance of a level-aware class. If the level is more than 2 then the instance's level must be at least 2, implying it is a MetaClass. Then the instance's level is set to the next lower level. Unless a special case (Why?) requires it to be 2. If the level was less than 2. It is just decreased by one. Only level 1 and 2 should enter this path, resulting in level 1 or 0. Level 0 cannot access this function, as this is meant to be prevented elsewhere. Therefore levels lower than 0 cannot occur if prevented properly.

```
+      if o.of().level > 2
+      then
+          o.addParent(FMML::MetaClass);
+          if o.of().of().allAttributes()->exists(a |
+              a.name = "level".asSymbol())
+          then
+              o.level := o.of().level - 1
+          else
+              o.level := 2
+          end
+      else
+          o.level := o.of().level - 1
+      end
+  end;
```

The skipped intrinsic attributes are added (as Attributes) to the instance. As no intrinsics should remain on level 0, o's level must be at least 1.

```
+      @While not intrinsicA.isEmpty() do
+          let intA = intrinsicA.sel()
+          in o.add(intA);
+          intrinsicA := intrinsicA.excluding(intA)
+      end
+  end;
+?      if Root.contents.keys().includes("TargetPackage".asSymbol())
+?      then
+?          TargetPackage.add(o)
```

```

+?      else
+?      false
+?      end;
+?      o
=      end
=      end

```

### 1.2.6.2 newToLevel

```

@Operation newToLevel(level:integer):Object

    let o = Kernel_mkObj();
    A = self.allAttributes();
in Kernel_setOf(o,self);
    @While not A->isEmpty do
        let a = A->sel
        if a.init <> null
        then
            Kernel_addAtt(o,a.name,a.init.invoke(o,Seq{}))
        else
            Kernel_addAtt(o,a.name,a.type.default())
        end;
        A := A->excluding(a)
    end
end;
if self.inheritsFrom(Classifier)
then
    o.parents := o.defaultParents()
end;
o.init();

```

Now case 2 is done

```

    if self.of() = Extensions::MetaAdaptor
    then
        o.level := level;
o.Isintrinsic = true;
        o.addParent(FMML::MetaClass)
    end;

    if Root.contents.keys().includes("TargetPackage".asSymbol())
    then
        TargetPackage.add(o)
    else
        false
    end;
    o
end
end

```



Another approach would not copy them through the levels. Instead the allAttributes function would check all higher levels for skipped attributes.

## 1.3 MeMo-Language

Req-1.3.0.2.1	The MultiLevel language MeMo requires. . .
Req-1.3.0.2.2	The MultiLevel language MeMo requires. . .
Req-1.3.0.2.3	The MultiLevel language MeMo requires. . .

## 1.4 Kernel

## 1.5 X-Modeler

### 1.5.1 Diagrams

#### 1.5.1.1 Multi-Level Diagram

### 1.5.2 Forms

#### 1.5.2.1 FormsClient

Req-1.5.2.1.1	The FormsClient receives messages to add components.
Req-1.5.2.1.2	<p>The FormsClient must be able to display these components:</p> <ul style="list-style-type: none"><li>• Label</li><li>• Textfield</li><li>• Textarea</li><li>• Checkbox</li></ul> <p>The component is determined by the type of the value.</p>
Req-1.5.2.1.2a	A boolean is shown as a Checkbox
Req-1.5.2.1.2b	An enum is shown as a drop-down-list
Req-1.5.2.1.2c	A short(?define) Strings or a number is shown as Textfield
Req-1.5.2.1.2d	A long(?define) String is shown as Textarea
Req-1.5.2.1.3	Labels can be grouped with other components to form a key-value-pair.
Req-1.5.2.1.4	A form client has a listener for changes which will be transferred instantly to the underlying model. There are no Save, Cancel or Undo buttons.
Req-1.5.2.1.5	The FormsClient is linked to the underlying model.
Req-1.5.2.1.6	The FormsClient's components are linked to the underlying model's parts by an id.
<i>Comment</i>	Labels should not be empty. <a href="#">Git Issue #34</a>
<i>Comment</i>	Double click should not freeze the form. <a href="#">Git Issue #33</a>

## 2 Specifications

Spec-2.0.2.1.1	Some specifications may be optional. <a href="#">Req-1.0.0.0.1</a>
----------------	--

### 2.1 Web Site

#### 2.1.1 Design

Spec-2.1.1.0.1	The design's primary colour is #1155BB. <a href="#">Req-1.1.2.0.1</a>
Spec-2.1.1.0.2	The design's secondary colour is #22AA44. <a href="#">Req-1.1.2.0.1</a>

### 2.2 XMF-Language

### 2.3 Kernel

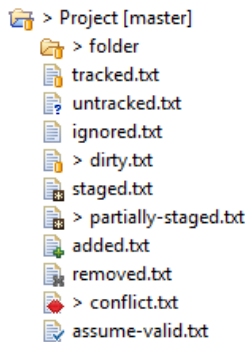
### 2.4 X-Modeler

#### 2.4.1 Diagrams

##### 2.4.1.1 Multi-Level Diagram

## 3 Other

### 3.1 git



- **dirty (folder)** - At least one file below the folder is dirty; that means that it has changes in the working tree that are neither in the index nor in the repository.
- **tracked** - The resource is known to the Git repository and hence under version control.
- **untracked** - The resource is not known to the Git repository and will not be version controlled until it is explicitly added.
- **ignored** - The resource is ignored by the Git team provider. The preference settings under Team > Ignored Resources, "derived" flag and settings from .gitignore files are taken into account.
- **dirty** - The resource has changes in the working tree that are neither in the index nor in the repository.
- **staged** - The resource has changes which have been added to the index. Note that adding changes to the index is currently possible only in the commit dialog via the context menu of a resource.
- **partially-staged** - The resource has changes which are added to the index and additional changes in the working tree that neither reached the index nor have been committed to the repository. See partial staging from the Git Staging view for how to do that.
- **added** - The resource has not yet reached any commit in the repository but has been freshly added to the Git repository in order to be tracked in future.
- **removed** - The resource is staged for removal from the Git repository.
- **conflict** - A merge conflict exists for the file.

- **assume-valid** - The resource has the "assume unchanged" flag. This means that Git stops checking the working tree files for possible modifications, so you need to manually unset the bit to tell Git when you change the working tree file. Also see Assume unchanged action.

### 3.1.1 fetch

Fetch from remote repository

### 3.1.2 pull

Incorporates changes from a remote repository into the current branch. In its default mode, git pull is shorthand for git fetch followed by git merge FETCH\_HEAD.

### 3.1.3 commit

Commit changes to local repository

### 3.1.4 push

Push Commits to remote repository

### 3.1.5 conflicts


[http://wiki.eclipse.org/EGit/User\\_Guide#Resolving\\_a\\_merge\\_conflict](http://wiki.eclipse.org/EGit/User_Guide#Resolving_a_merge_conflict)

### 3.1.6 branches

branch merge rebase cherry-pick

### 3.1.7 revert

If a commit which has not yet been pushed has to be undone:

Select folder  Team → Show in History  
 Note number of commit  
 then use command line  
`git revert <noted number>`

If a commit which has already been pushed has to be undone:

Is done the same way.

Note: Both reverts don't undo history. The wrong commit and the reversion will be logged.

## 4 Example

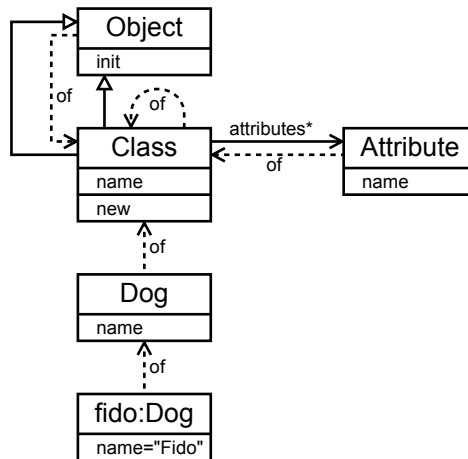


Figure 4.1: Basic Class diagram

Dog extends ( $\xrightarrow{\text{extends}}$ ) Object. It does not extend Class. It is an instance of ( $\xrightarrow{\text{of}}$ ) Class. Fido is an instance of ( $\xrightarrow{\text{of}}$ ) Dog. Therefore (commonly spoken): Fido is a Dog. Dog is a Class. But: Is Fido an Object? Is Dog an Object?

Differentiate between

- the *operation* of() which returns the target of the  $\xrightarrow{\text{of}}$  of-Arrow and
- the *operator* instance? which returns true if second argument is  $\xrightarrow{\text{of}}$  of() or anything of()  $\xrightarrow{\text{of}}$   $\xrightarrow{\text{extends}}$  inherited from
- isTypeOf
- isReallyTypeOf

```
parserImport XOCL;
```

The class OmniscientClass extends Class. Any instance of it will therefore be a class.

```
context Root
  @Class OmniscientClass isabstract extends Class
  @Attribute instances : Vector = Vector(12) end
  @Attribute numberInstances : Integer = 0 end
  @Attribute pluralName : String (?) end
```

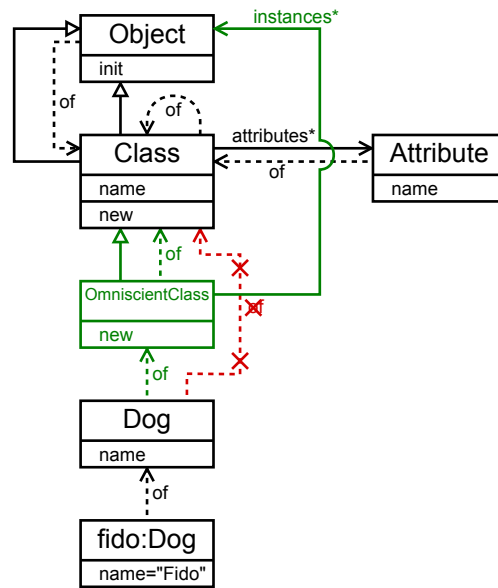


Figure 4.2: Extended Class diagram

```
@Operation getPluralName() : String
  name + "s"
end
```

```
@Operation new():Element
  if numberInstances >= 3
  then let
    o = super()
  in
```

This line is throwing an exception, when too many instances of the `OmniscientClass` are created. It uses the function `getPluralName()` to test about overwriting functions. In most cases the plural is formed by appending a `s`. This is done in the default function above. The word “Mouse” however has an irregular plural and therefore needs to overwrite the function, returning “Mice” instead of the default “Mouses”.

Instead of overwriting functions, attributes could be used. In this case it proves a worse choice as every slot has to be filled manually, and no useful default can be provided.

```
    self.error("According to popular belief, "
      + (numberInstances+1) + " "
      + o.getPluralName()
      // + pluralName
      + " are deemed unlucky.")
end
else
  let
    o = super()
  in let
    c = o.of()
```

```

        in
        c.instances.put(numberInstances,o);
        self.numberInstances := numberInstances +1;
        o
    end end
end
end
end

```

Class Dog is an instance of OmniscientClass. It is a class because it's of is Class or a any extension thereof.

```

context Root
  @Class Dog metaclass OmniscientClass
  @Attribute name : String = "nameless Dog" end
  @Constructor(name) ! end
  @Slot OmniscientClass::pluralName = "Dogs" end

```

As Dog does not extend Class, this function is not supposed to be called. However, it can be called by (Dog())(), naturally causing an exception to be thrown somewhere in super(). It can be called without an exception if Dog would extend Class.

```

  @Operation new():Element
    "Dog.new".println();
    super()
  end
end

```

```

context Root
  @Class Cat metaclass OmniscientClass
  @Attribute name : String = "nameless Cat" end
  @Constructor(name) ! end
  @Slot OmniscientClass::pluralName = "Cats" end
end

```

The class Mouse

```

context Root
  @Class Mouse metaclass OmniscientClass
  @Attribute name : String = "nameless Mouse" end
  @Constructor(name) ! end
  @Slot OmniscientClass::pluralName = "Mice" end
  @Operation getPluralName() : String
    "Mice"
  end
end
end

```

```

Root::dog1 := Dog.new();
Root::dog1.name := "Fido";
Root::mouse1 := Mouse("Mickey");

```



Objects have a type and slots. The type is returned by the `of()` operation and the slots are accessed by name either using the `'.'` operator or the `get(name)` operation. Values of slots can be updated using the `o.s := v` notation or the `o.set(s,v)` operator. Messages can be sent to an object using the `o.m(a1,a2,...)` notation or using the `o.send(m,Seq a1,a2,...)` operator.

Objects implement a MOP via their meta-class that determines how slot access and update is handled. The MOP is defined on an objects type via the `getInstanceSlot`, `hasInstanceSlot` and `setInstanceSlot` operations. Define a new meta-class (sub-class of `Class`) that implements a new MOP and therefore handles object storage completely differently to the VM.

**Slot storage** is usually allocated when a class is instantiated and **is always consistent with the attributes defined by the class**. You can modify this using `addStructuralFeature` and `removeStructuralFeature`, but you **must be aware that certain key execution features of XMF rely on the slots of an object conforming to the attributes of its class**. NB adding new slots to an object that do not clash with these attributes is safe and can be used to squirrel away information in particular objects.

#### 4.0.8 Multilevel Draft Essen

Extend Association, Attribute, Compiled Operation by `isIntrinsic`, `instLevel`, `isCore` and `Constraint` and `Doc` by `id`.

new class/interface `MetaAdapter` overriding `new`(decreasing level by one) with additional function `newAtLevel`(decreasing level by more than one)

???

Is a level  $\leq 1$  (class, object) automatically forbidden to extend `MetaAdapter/Class`? Is a level 0 (object) automatically forbidden to be of type `MetaAdapter/Class`? Is an element which does not extend `MetaAdapter/Class` but is of type `MetaAdapter/Class` automatically a level 1? Is an element which is not of type `MetaAdapter/Class` automatically a level 0?

Why start numbering levels at object level? This means: a classic object is always level 0, a classic class always level 1. Therefore it is always necessary to know before the number of the top layer. Or do the upper layers number automatically? But then the `instLevel` for attributes does not give an absolute level-distance. Maybe allow relative `intLevels`? Or use another concept than integer numbering.

!!!

#### 4.0.9 next Try

An element  $e$  is level 0, if and only if  $e.of()$  is level 1.

$Level_0(e) \Leftrightarrow Level_1(e.of())$

$Level_1(e) \Rightarrow \neg e \text{ inheritsFrom } Class.$

$Level_1(e) \Leftrightarrow Level_2(e.of()) \wedge \neg e \text{ inheritsFrom } Class.$

$Level_n(Class) \forall n \geq 2$

#### 4.0.10 next Try II

#### 4.0.10.1 numbering up

Elements can be numbered by their distance to an object, if that object is given. Object *o*'s Level in respect to *o* is 0. Object *o*'s *of()*'s Level in respect to *o* is 1. Class' Level in respect to *o* is the number of *of()*s one has to traverse, including but not counting *inherits*.

```
int levelUp(Element a, Element b) {
    if(a == b) return 0;
    if(b.of() != b) return 1 + levelUp(a, b.of());
    if(b.inherits() != b) return levelUp(a, b.inherits());
    throw Exception;
}
```

#### 4.0.10.2 numbering down

Elements can be numbered by their distance to *Class*. Object *o*'s Level in respect to *Class* is the number of *of()*s one has to traverse from *o* to *Class*, including but not counting *inherits*. Be this number negative for easier disambiguation.

```
int levelDown(Element e) {
    if(e == Class) return 0;
    if(e.of() != e) return -1 - levelDown(e.of());
    if(e.inherits() != e) return levelDown(e.inherits());
    throw Exception;
} <==> {
    return - levelUp(Class, e);
}
```

#### 4.0.10.3 allowing multi-level jumps

If we have a *of*-relation with a specified weight  $\geq 1$  we can amend the formulas above and replace 1 with the weight.

```
int levelUp(Element a, Element b) {
    if(a == b) return 0;
    if(b.of() != b) return b.ofWeight() + levelUp(a, b.of());
    if(b.inherits() != b) return levelUp(a, b.inherits());
    throw Exception;
}
```

#### 4.0.10.4 Finding a unified numbering scheme

Assume the following relation:

fido  $\xrightarrow{\text{of}}$  Dog  $\xrightarrow{\text{of}}$  Quadruped  $\xrightarrow{\text{of}}$  Animal  $\xrightarrow{\text{of}}$  MetaClass  $\xrightarrow{\text{of}}$  Class  
 squidward  $\xrightarrow{\text{of}}$  Octopus  $\xrightarrow{\text{of}}$  Animal  $\xrightarrow{\text{of}}$  MetaClass  $\xrightarrow{\text{of}}$  Class  
 where (Quadruped and Animal)  $\xrightarrow{\text{extends}}$  MetaClass  $\xrightarrow{\text{extends}}$  Class

x	levelDown(x)	levelUp(x, fido)	levelUp(x, Octopus)	levelUp(x, Animal)
fido	-5	0	N/A	(-3)
Dog	-4	1	N/A	(-2)
Quadruped	-3	2	N/A	(-1)
squidward	-4	N/A	(-1)	(-2)
Octopus	-3	N/A	0	(-1)
Animal	-2	3	1	0
MetaClass	-1	4	2	1
Class	0	5	3	2

The levelUp function can be extended downwards into negative results.

If we want a Slot name for every Animal, we would add an Attribute name to Animal to become a Slot on Level 0. But we cannot say which level Animal is on.  
 We cannot instantiate an animal object directly to level 0, as Animal inherits Class

## 5 Literature

- A Unifying Approach to Connections for Multi-Level Modeling [2]
  - Melanie: Multi-level Modeling and Ontology Engineering Environment [3]
  - Referenced 2
  - Referenced 3

### 5.1 Integrating Language Engineering [1]...

- XModeler
  - What is XModeler?
  - Getting started
    - \* Install guide
    - \* Tutorial
  - Help
    - \* API
    - \* Language concepts
    - \* Snippets <sup>X</sup>
    - \* Bluebook <sup>X</sup>
- Literature
  - Articles, Papers
    - \* Our Articles, Papers
    - \* Conferences
    - \* List of others' literature
  - Lecture Slides and other material for students <sup>O</sup>
- External Links
  - git
  - shu
  - ude
- Contact
  - Who are we
  - How to contact us
  - Impressum

<sup>X</sup>: Not available online

<sup>O</sup>: Only available online

Is any material we use copyrighted? Are icons and logos in XModeler our work? If they are from any "free" source, are there any conditions on them?

Do we need any legal advice? Disclaimers?

## 6 Weekly Reports

### 6.15 2015

#### 6.15.37 7 Sep 2015 - 11 Sep 2015

- Made browser tabs closeable
- Init Requirements and Specifications in LaTeX
- Added Panic button to GUI
- Image loading: XML file now derived from IMG file name and path instead of storing that information in the IMG file.
- Division of BigIntegers fixed, Multiplication of negative Integers fixed

#### 6.15.38 14 Sep 2015 - 18 Sep 2015

- Tested and documented some git features.
- Produced conflicts in git: Managed to resolve them finally. It was challenging. Further testing required.
- Tried to install safari. Waiting for Help desk...
- Made the browser work again. Several fixes: Removed layouter, changed locking mechanism, set to native browser

#### 6.15.39 21 Sep 2015 - 25 Sep 2015

Used XMF to do exercises with meta-classes

- to understand the concept of class, object, instantiation and inheritance
- to override the default meta-object-protocol, be redefining new/get/set etc.
- to understand the default behaviour of XMF

#### 6.15.40 28 Sep 2015 - 2 Oct 2015

- Continued experiments with meta-classes
- Gathered ideas for website; some experiments with html.

#### 6.15.41 5 Oct 2015 - 9 Oct 2015

- Preparation for discussions about intrinsics.
- Fr: Travel

#### **6.15.42 12 Oct 2015 - 16 Oct 2015**

- Mo,Tu,Fr: holiday
- Making documentation for scenarios for changes on intrinsic attributes
- Tiny bugfix in FormsClient

#### **6.15.43 19 Oct 2015 - 23 Oct 2015**

- Mo: Travel
- Added Monospace (with anti-alias) font to editor window.

#### **6.15.44 26 Oct 2015 - 30 Oct 2015**

- 

#### **6.15.45 ToDo**

- improve webpage
- Specify FormClient, fix only
- git commands in Eclipse
- (nextWeek) MeMo language requirements
- pw protected folder: [https://wincent.com/wiki/git\\_repository\\_access\\_control](https://wincent.com/wiki/git_repository_access_control)
- Snippets
- look at/play with Kernel (Object/Class/Attribute)
- HTMLViewer.xmf: Are ".o", ".xip", ".xto", ".xtd", ".xml" still in use? Welcome.xmf: requestURL ... requirements
- welcome page does not work in linux
- experiment: call java function
- fix documentation bug (over lunch)

# Bibliography

- [1] T. Clark and U. Frank. Integrating language engineering, conceptual modelling and programming: A multilevel development and runtime environment. 0000.
- [2] C. Atkinson, R. Gerbig and T. Kühne. A unifying approach to connections for multi-level modeling. *Proceedings of 18th ACM/IEEE International Conference MODELS*, 2015.
- [3] C. Atkinson and R. Gerbig. Melanie - multi-level modeling and ontology engineering environment. 2012.
- [4] H. J. Farnsworth and L. Simpson. Mathematics in the year 3000. *Proceedings of 318th IEEE International Conference*, 55(4), 3015, pp. 309–319.