SOFTWARE REQUIREMENTS SPECIFICATION

for

XModeler

Release 1.0

Version 1.0 approved

Contents

Revision History

1 Requirements

Req-1.0.0.0.1	Some requirements may be optional. Spec-2.0.2.1.1

1.1 MeMo-Language

Req-1.1.0.0.2	The MultiLevel language MeMo requires. . .
Req-1.1.0.0.3	The MultiLevel language MeMo requires. . .
Req-1.1.0.0.4	The MultiLevel language MeMo requires. . .

1.2 Kernel

1.3 X-Modeler

1.3.1 Diagrams

1.3.1.1 Multi-Level Diagram

1.3.2 Forms

1.3.2.1 FormsClient

Req-1.3.2.1.1	The FormsClient receives messages to add components.
Req-1.3.2.1.2	The FormsClient must be able to display these components: • Label • Textfield • Textarea • Checkbox The component is determined by the type of the value.
Req-1.3.2.1.2a	A boolean is shown as a Checkbox
Req-1.3.2.1.2b	An enum is shown as a drop-down-list
Req-1.3.2.1.2c	A short(?define) Strings or a number is shown as Textfield
Req-1.3.2.1.2d	A long(?define) String is shown as Textarea
Req-1.3.2.1.3	Labels can be grouped with other components to form a key-value-pair.
Req-1.3.2.1.4	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-1.3.2.1.5	The FormsClient is linked to the underlying model.
Req-1.3.2.1.6	The FormsClient's components are linked to the underlying model's parts by an id.
Comment	Labels should not be empty. Git Issue #34
Comment	Double click should not freeze the form. Git Issue #33

2 Specifications

Spec-2.0.2.1.1	Some specifications may be optional. Req-1.0.0.0.1

2.1 XMF-Language

2.2 Kernel

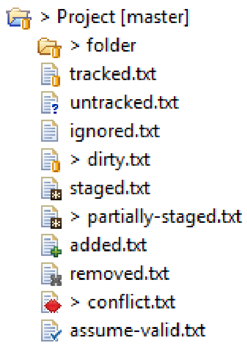
2.3 X-Modeler

2.3.1 Diagrams

2.3.1.1 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`

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

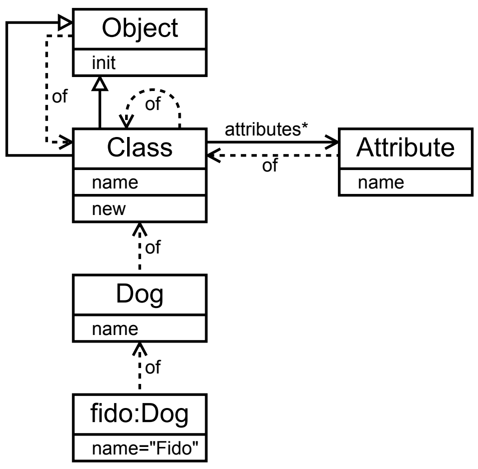


Figure 4.1: Basic Class diagram

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

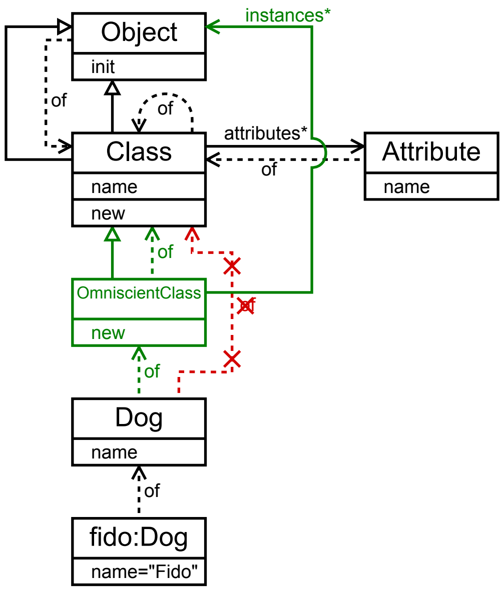


Figure 4.2: Extended Class diagram

```
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

  @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

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

5 Literature

- A Unifying Approach to Connections for Multi-Level Modeling [1]
  - Melanie: Multi-level Modeling and Ontology Engineering Environment [2]
  - Referenced 2
  - Referenced 3

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 con icts 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 metaclasses

6.15.40 ToDo

- improve webpage
- Specify FormClient
- git commands in Eclipse
- (nextWeek) MeMo language requirements
- pw protected folder: `https://wincent.com/wiki/git_repository_access_control`
- Snippets
- look at/play with Kernel (Object/Class/Attribute)

- Use Notepad++ with syntax highlighting for user defined languages: `http://weblogs.asp.net/jongalloway/creating-a-user-defined-language-in-notepad`
- HTMLViewer.xmf: Are ".o", ".xip", ".xto", ".xtd", ".xtml" still in use? Welcome.xmf: requestURL. . . requirements

Bibliography

[1] 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.

[2] C. Atkinson and R. Gerbig. Melanie multi-level modeling and ontology engineering environment. 2012.

[3] H. J. Farnsworth and L. Simpson. Mathematics in the year 3000. *Proceedings of 318th IEEE International Conference*, 55(4), 3015, pp. 309–319.