SMDA

PROJECT: SMDA

SUBPROJECT: FREEMDA

Siegfried Nolte
http://www.siegfried-nolte.de/forum/mda/smda.html
siegfried@siegfried-nolte.de
October 2009

INHALTSVERZEICHNIS

1 freeMDA - Overview 4
1.1 Content 4
1.2 Help 5
2 Architecture Views 6
2.1 freeMDA Architecture Models 6
2.2 EMW 7
2.3 freeMDA 8
2.4 freeMDAX 9
3 Project Organization 11
3.1 Project-Organization 11
3.1.1 Elements of „Project-Organization" 12
4 Sample MDA-Process 16
4.1 MDA Process Models 16
4.2 Prozess: MDA Process - freemda sample 16
4.2.1 Elements of MDA Process - freemda sample 18
4.2.1.1 Partitions 18
4.2.1.2 Actions 18
5 Other Process Views 26
5.1 1-SWE Process - classical view 26
5.2 2-MDA Transformation Pattern 27
5.3 3-sample MDA Pattern 28
5.4 4-M2M-M2T Differentiation 29

ABBREVIATIONS

Abbreviation	Meaning
ATL	Atlas Transformation Language
BPMN	Business Modeling Notation
CIM	Computation Independent Model
DB	Database
DDL	Data Definition Language
EJB	Entity Java Bean
EMF	Eclipse Modeling Framework
EMW	Eclipse Modeling Workbench
ERM	Entity Relationship Model
GEF	Graphical Editing Framework
GMF	Graphical Modeling Framework
HTML	Hypertext Markup Language
IM	Implementation Model
JET	Java Emitter Templates
JMI	Java Metadata Interface
M2M	Model-to-Model
M2T	Model-to-Text
MDA	Model Driven Architecture
MOF	Metaobject Facility
MTL	Model-to-Text Transformation Language
OCL	Object Constraint Language
OM	Operational Mappings
PIM	Platform Independent Model
PSM	Platform Specific Model
QVT	Query Views Transformation
RL	Relations Language
SDK	Software Development Kit
SMDA	Simple MDA
UML	Unified Modeling Language
VTL	Velocity Template Language

1 FREEMDA - OVERVIEW

1.1 CONTENT

freeMDA

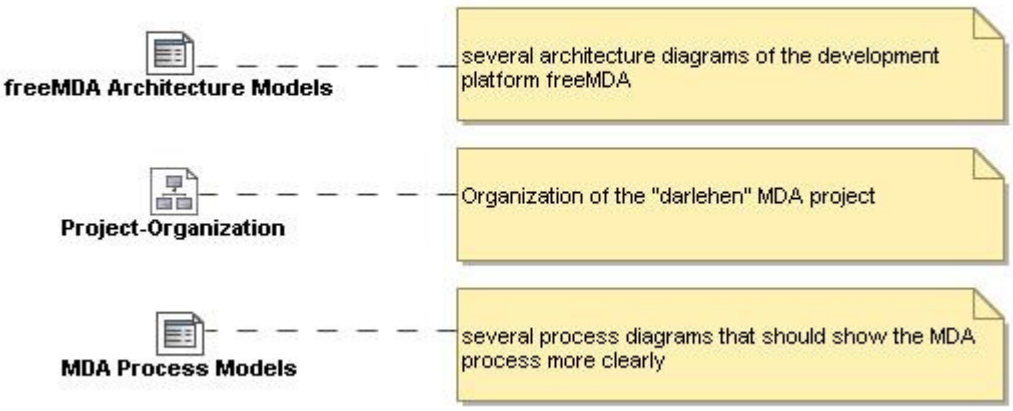
This documentation shows the architecture and usage of a free MDA platform. The MDA platform project is part of the SMDA project that deals with MDA techniques generally, especially with metamodeling and model transformation.

freeMDA has been constructed based on the Eclipse Modeling Workbench project. Information and references about and around SMDA can be found on ->

This documentation has been created with a model driven HTML-creation process. For some help information on navigating through the model take a look at the "help" option.

Note:
If there are build-errors while opening this project, this project should work with Acceleo MTL 0.8. The expected Workbench Path is "D:/projekte/MDA/MDA/fMDA/TC23"; change it to your path. For further information take a look at the architecture.

Hamburg, Oktober 2009



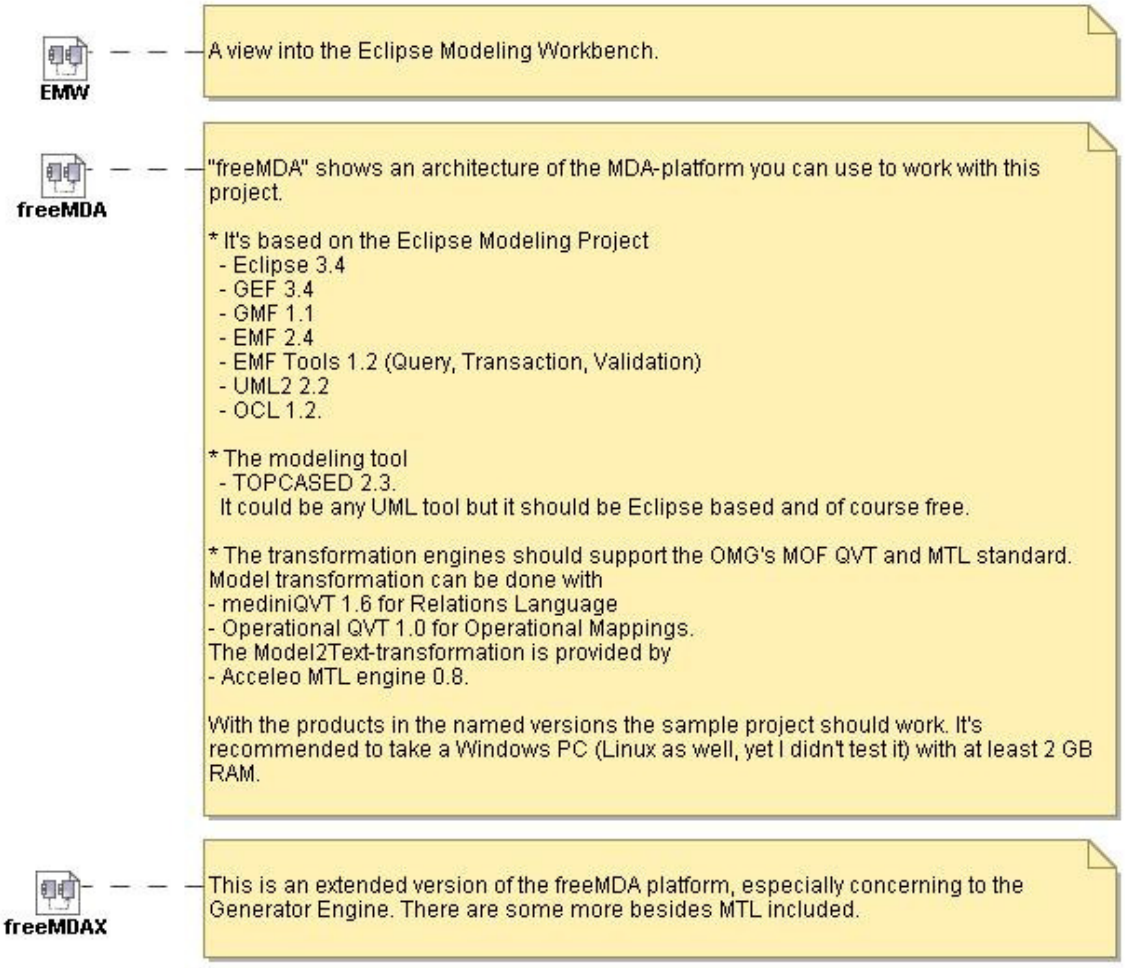
1.2 HELP

freemda - Navigation Hints

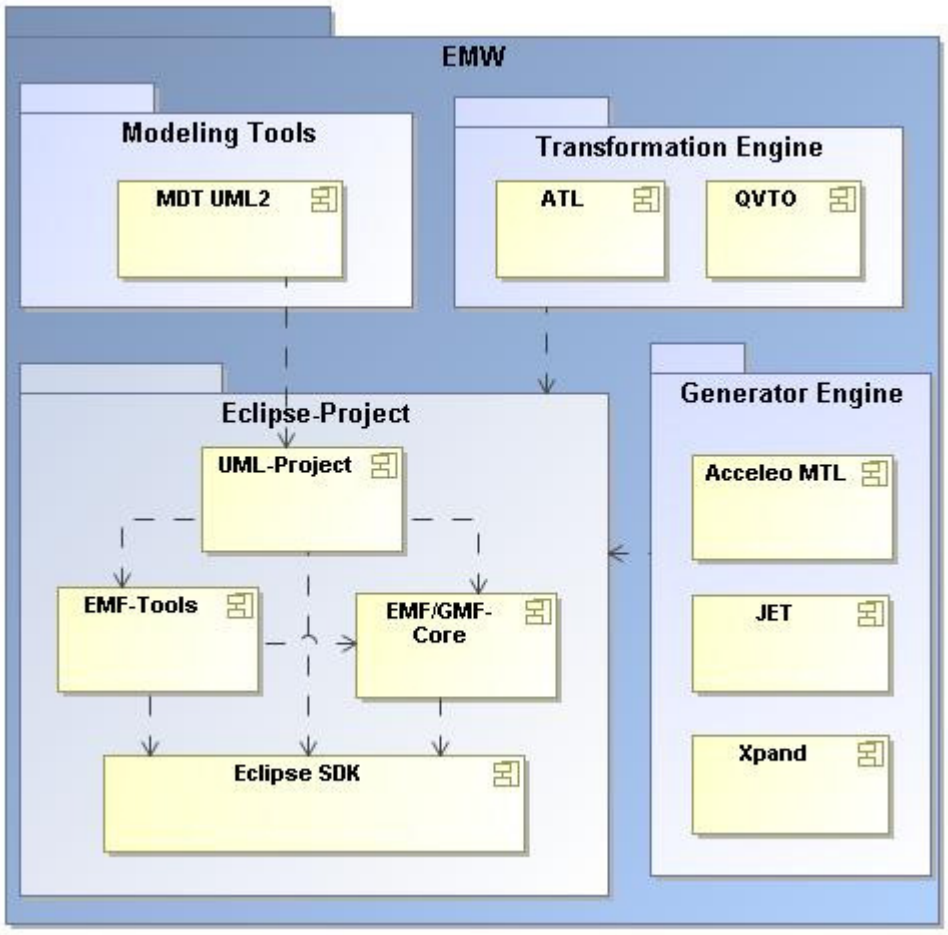
- This documentation has been generated with a model driven HTML-creation process.
- You see 3 frames from which you are able to navigate through the model
  - The big one left is the diagram-frame with some UML diagrams in it.
  - Right above is the model-tree-frame.
  - Right below is the information- or property-frame.
- All frames allow navigation by clicking on elements (diagram), links or symbols (other frames).
- Clicking diagram-symbols, that represent diagrams, shows the other diagram within the diagram-frame and its properties rigth below.
- Clicking on any other element in a diagram or elsewhere only shows the element's properties rigth below.
- Clicking on an element, that is connected with some other element of diagram type, for instance call behavior actions, shows the element's properties. The connected diagram is a property that could be selected from the properties-frame.
- Properties may consist of hyperlinks to other sites. Hyperlinked sites will be shown in a new browser window.
- Above there is a titlebar menu with the options
  - ***start*** leads you to the welcome- or content-diagram
  - ***+Frames*** display with all frames
  - ***-Frames*** display with diagram-frame only (for big, complex diagrams)
  - ***Help*** shows this Help-Window
  - ***Search*** allows you to search for keywords

2 ARCHITECTURE VIEWS

2.1 FREEMDA ARCHITECTURE MODELS

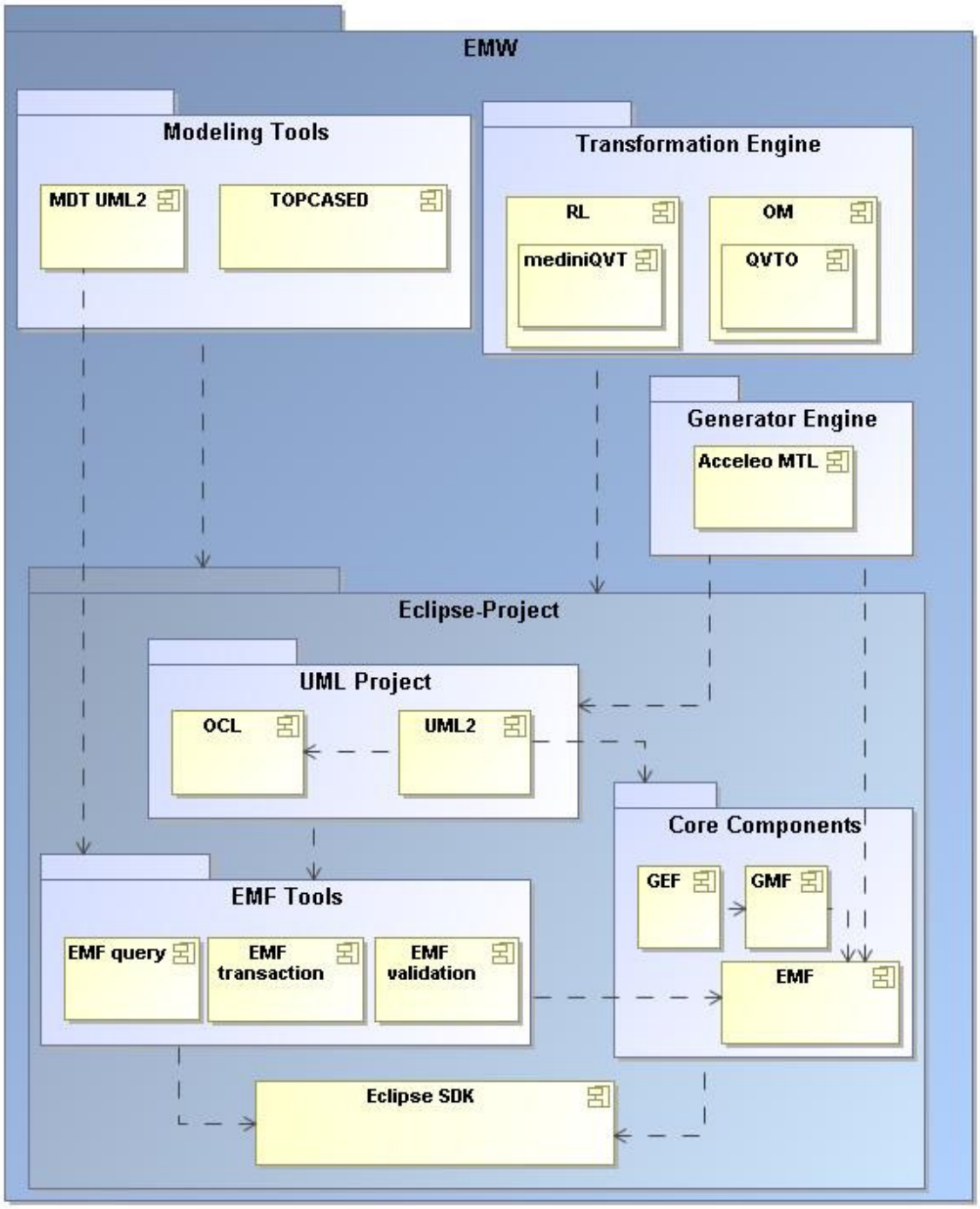


2.2 EMW



This diagram gives an overview of the Eclipse Modeling Workbench (EMW) in its original construction. It's not complete, though representative. The Eclipse Modeling Project gathers modeling and model transformation components and integrates them into a complex MDA workbench, of course Eclipse based.

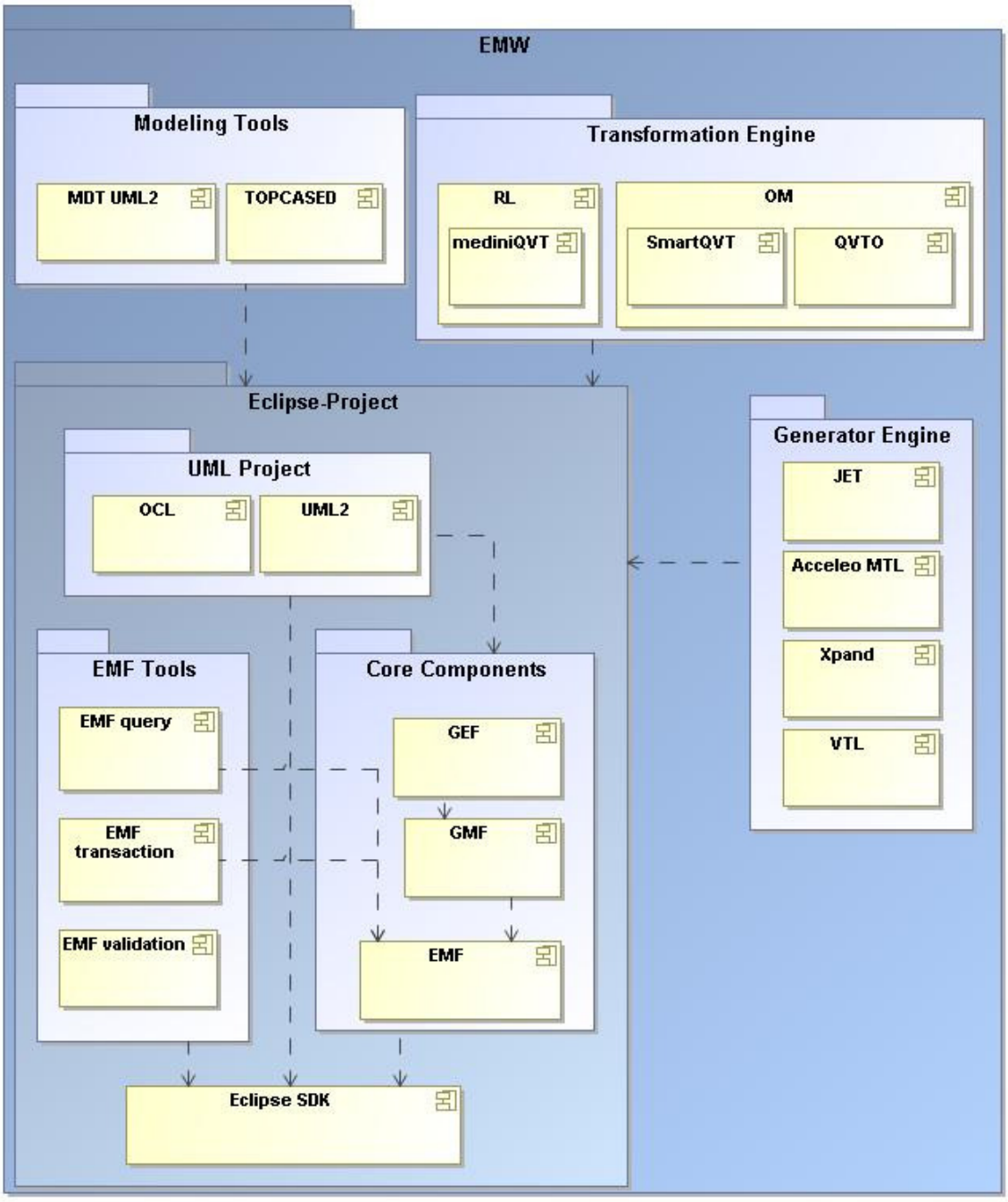
2.3 FREEMDA



"freeMDA" is our EMW based integrated MDA platform. The modeling tool is TOPCASED, for model transformation we provide a Relations Language tool, mediniQVT, and an Operational Mappings tool,

Operational QVT. Code Generation can be one with Acceleo MTL. The rest is Eclipse Modeling Framework.

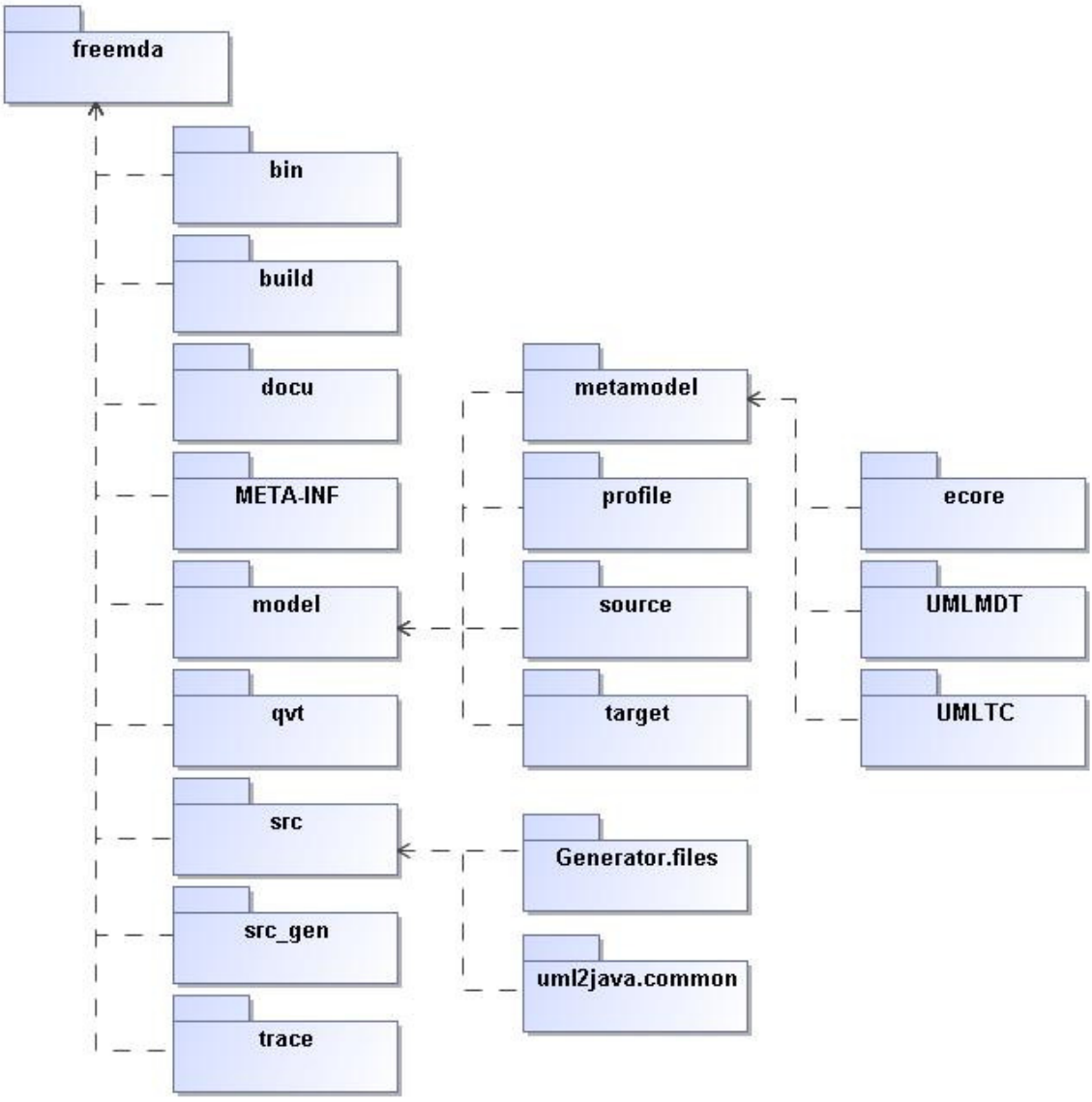
2.4 FREEMDAX



"freeMDAX" is the freeMDA workbench in an extended edition. SmartQVT has been added for Operational Mappings transformation. It's a bit nearer to the QVT specification, though the handling is a bit more complicated. Besides Acceleo MTL code generation can be done with JET, Xpand and VTL. JET is the classical tool from EMF, Xpand comes from openArchitectureWare and VTL from AndroMDA. VTL is also provided with MagicDraw, if you like modeling with this tool, so it could be integrated in freeMDAX as well.

3 PROJECT ORGANIZATION

3.1 PROJECT-ORGANIZATION



This diagram shows the folder organization of the freemda project. Mind that this project needs a freeMDA workbench, that is installed in "D:/projekte/MDA/MDA/fMDA/TC23". You may build another one for your own, but you should modify the project's classpath file accordingly. MTL should be Acceleo 0.8 or newer.

3.1.1 ELEMENTS OF „PROJECT-ORGANIZATION"

bin

"bin" is for the Java binaries. Those class files result from Eclipse's incremental compile. So if luckily there are any, we needn't bother them.

build

"build" contains some ant build files for MTL code generation using an ant batch process. It should work provided the classpath definition is correct, but I rather do my launching the scripts with the Eclipse runtime environment.

This ant build file is taken from the generated build tasks that can be found in the "tasks" folder. It is modified a bit so I hope it works. By the way this build generates the Java code from the PSM model (model/target).

docu

This documentation - it is done with MagicDraw 16.5 and generated from MagicDraw model using a model driven Doc-Generator. If you want to learn something about the document generation ask the author.

ecore

Our simple metamodels modeled with TOPCASED Ecore tools.

freemda

The development project "freemda" shows the complete way of a MDA process using a free MDA workbench. The MDA life cycle generally consists of modeling, transformation and code generation. You may learn some more details about the process from the process diagrams (follow the reference link).

This project will show some work of business modeling ("model/source/*"), building architecture models ("model/metamodel", "model/profile") and writing transformation ("qvt") and code generation scripts ("src").

A short run:
(1) We will start our way beginning with PIM modeling of a financing domain, the "darlehen" project. You will learn a little about it having a look at the source model "model/source/darlehen.umldi".
(2) Then we will transform the source model into a PSM EJB model using QVT techniques (with both QVT Relations and QVT Operational). This leads to "model/target/darlehen*.uml" and "model/target/darlehen*.umldi".
(3) For this task a special UML profile is needed ("model/profile/UMLEJB.uml").
(4) The PSM model will be modified a bit, defining datatypes and that like.
(5) At last Java code will be generated with a MTL model-2-text generation process that results in "gen_src/darlehen/*".

References:
- mdel://_16_5_4_5d80203_1256739454995_56375_1598

Generator.files

This folder contains the main template script and its "generator.java"-file. If you create a new Acceleo template, you will get a "Generate<templatename>.java", too.

You may change the template files "<templatename>.mtl" only and not the generator-file, unless you know, what you are doing. Well, and modify the template files as you like. If you want to learn something about MTL scripting, take a look at the specification MOF MTL. Again, references can be found on the SMDA project site.

References:
- http://www.siegfried-nolte.de/forum/mda/smda.html

META-INF

"META-INF" is for the usual manifest information. I usually don't touch it.

metamodel

"metamodel" contains our metamodels, in the first instance the simple ones, SimpleUML and SimpleRDBM.

We have different options of modeling metamodels:
- Ecore for modeling them using the TOPCASED Ecore tools.
- UMLMDT does the trick with tools of the UML2 Modeling Toolkit.
- UMLTC is for modeling metamodels with TOPCASED UML2 class diagram.

For deploying them into an Eclipse context, you will always need the "genmodel" file representation and generate the EMF-plugins. For more information about how to build and deploy metamodels you should follow the hyperlink to my project site - Part2 - or else the EMF tutorial sites.

The metamodeling stuff is only for some demonstration of living metamodeling in an Eclipse environment. This sample project doesn't really need the metamodels; we use UML profiles for our domain specific purposes.

References:
- http://www.siegfried-nolte.de/forum/mda/smda.html

model

"model" is the folder for our modeling business. In here should be the models of our platform independent and platform specific modeling and the architecture models as well:
- "metamodel" consists of metamodels, if there are any
- "profile" is a folder for the UML profiles
- "source" contains the source models, or rather the PIM models
- "target" contains the PSM models

profile

As I've said in the metamodel info, I won't need my own metamodels so much. I will work with an UML2 sample model. But it may be useful to have a little "domain specific" annotation. So it may be useful to have an UML profile. At least I want to show how to model a profile. Again this one, UMLEJB, is really simple. I only want to demonstrate how to work with a profile in transformation. This UMLEJB-profile is modeled with TOPCASED UML model using a profile template.

qvt

The folder "qvt" contains the QVT scripts. Those with filetype "qvt" are Relations Language scripts written with mediniQVT. The "qvto" ones are Operational QVT scripts.

UML2EJB should do the PIM/PSM transformation of business class diagrams to UML EJB class diagrams. The CheckProfile scripts check the UML-Profile and the stereotyped class diagrams.

The transformation is being performed with the Eclipse run configurations
* "QVTO_Darlehen" for the operational transformation
* "QVTR_Darlehen" for the relational transformation

Model transformation with QVT is handled deeply in my books (German language); again follow the link.

References:
- http://www.siegfried-nolte.de/forum/mda/smda.html

source

"source" is the folder of source models or in other words PIM models.
Well, there is a "darlehen.simpleuml", modeled with the "SimpleUML model editor" that is generated from the metamodel SimpleUML. So you see, modeling with SimpleUML can be done. But I will go on with the "darlehen.uml" model.

"darlehen.uml" is simple as well, but it is modeled with TOPCASED UML. It's a PIM class diagram that shows a bit of a house financing domain. The classes have German names and notions, though that should not bother, it's an example.

src

In "src" folder the source code files of the generator can be found. Most of them are generated themselves while creating a new Acceleo template with the "Acceleo/MTL generator engine". At least all of those Java files. You better do not touch these ones.

The files with "mtl"-suffix are the model-2-text template files which can be edited and completed for M2T transformation. The main template file is "Generator.files/generateJava.mtl". Start with this and try to do some changes. More templates you will find in the folder uml2java.common. Coding MTL is done according to the OMG's MOF MTL specification.

src_gen

"src_gen" is the folder for the generated Java code. With some luck you will find the "darlehen" EJB components in here. You may take any model from the model folder, but you may get some compile errors in the generated code because there is a wrong package name.

target

"target" is the folder of target models or in other words PSM models.

The PIM model "darlehen.uml" has been transformed to different PSM models:
- "darlehen-om.uml", having done the M2M with QVT Operational
- "darlehen-rl.uml", having done the M2M with QVT Relational
- "darlehenX.uml" is an extended version with generated getter and setter methods, but we rather want to do that with M2T-transformation.
- "darlehen.uml" is a copy of one of those models because the MTL script works with a model named "darlehen.uml".

The transformed uml models can be loaded into a TOPCASED class diagram again, take "New UML model with TOPCASED > Create from an existing model". This leads to a diagram that shows some EJB SessionBean components, each packed within a package that consists of the bean class and the interfaces as well. You may do some more PSM modeling now, but be aware, if you transform again to "target > darlehen*.uml", your work is gone. That's a problem but it's a problem for the tool developers, I think. Beware, we work with free tools.

Note:
There has been some own primitive datatypes generated and all attributes should be of one of that datatypes so they can be transformed to better Java code. Putting datatypes to attributes is a bit of typical PSM work in my eyes. At least we want to give the UML "Unlimited Natural" an own type "Double".

trace

"trace" is an optional folder for the QVT traces and others.

uml2java.common

This is an arbitrary folder that contains some more MTL templates. It's something like a MTL-module library.

UMLMDT

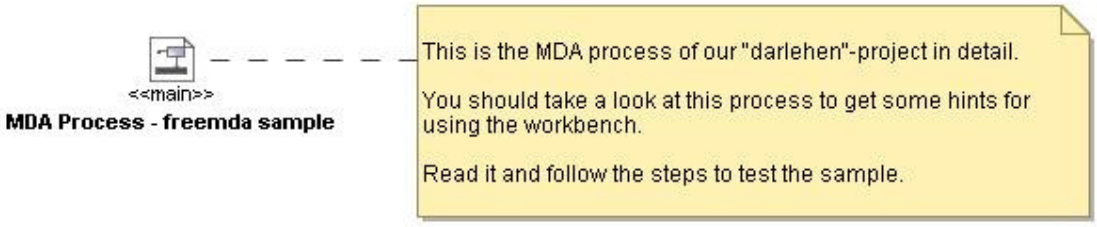
Our simple metamodels modeled with UML Modeling Toolkit.

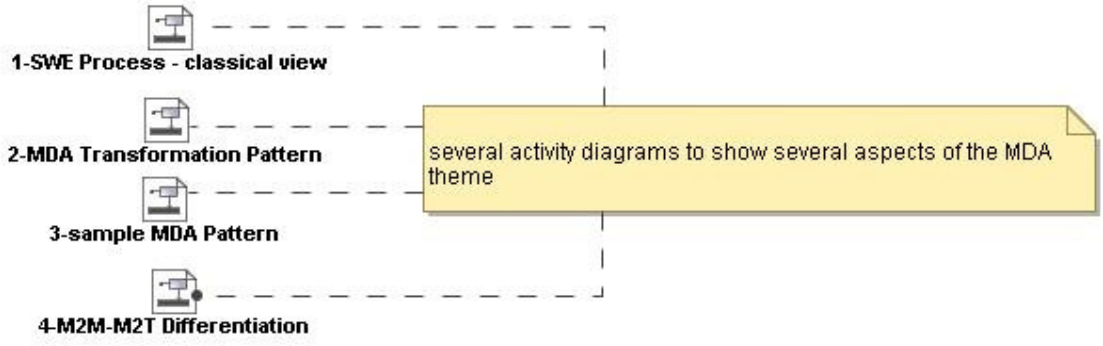
UMLTC

Our simple metamodels modeled with TOPCASED UML.

4 SAMPLE MDA-PROCESS

4.1 MDA PROCESS MODELS





4.2 PROZESS: MDA PROCESS - FREEMDA SAMPLE

This diagram shows the development process of this project.

In short:

(1) "Business Modeling"
First a bit of business modeling is done to get the business class diagram "model/source/darlehen.uml".

(2) "M2M Transformation"
The next step is model transformation from "model/source/darlehen" (= umlfk) to "model/target/darlehen" (= umlejb). The target model realizes some aspects of an EJB based architecture. The model transformation is done
- either with a Relations Language script "qvt/UML2EJB.qvt"
- or with an Operational Mappings script "qvt/UML2EJB.qvto".

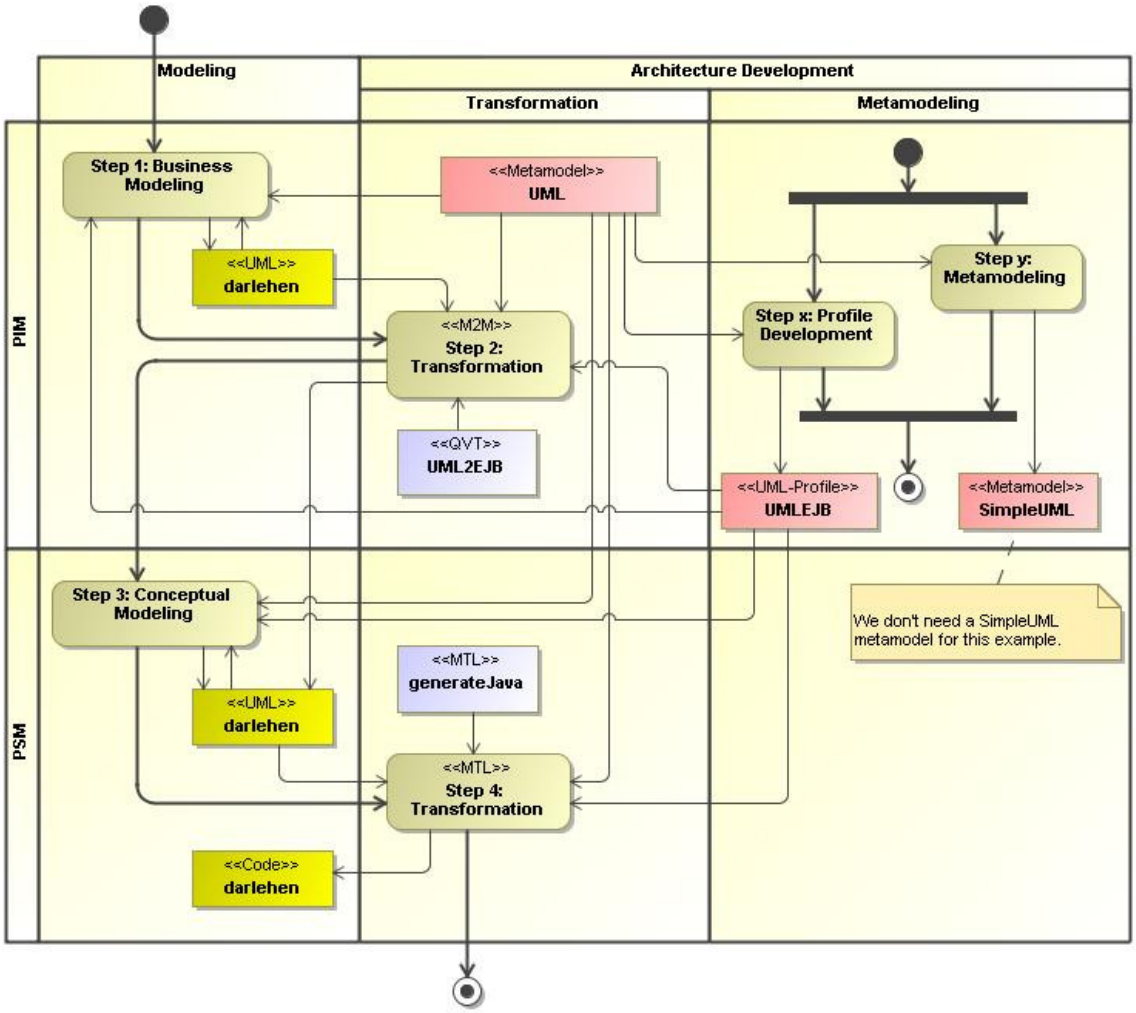
(3) "Conceptual Modeling"
Now you can do some more design and PSM modeling on the target model, for instance defining data types. That modifies the umlejb model "model/target/darlehen" in some way.

(4) "MTL Transformation"
The umlejb model is the input of a model2text transformation, "src/Generator/files/generateJava.mtl" and the modules "src/uml2java/*.mtl", to get the Java sourcecode, "gen_src/darlehen/*".

We may need some additional architecture modeling activities to build a metamodel and/or a profile:

(5) "Metamodeling"
The class modeling is done with UML2. It is also possible to build a modeling language "SimpleUML" for modeling simple class diagrams. We did it, "model/metamodel/Ecore", but we won't use it.

(6) "Profile Development"
We need an UML-Profile "model/profile/UMLEJB-Profil" for marking the EJB candidates of the source-model with the stereotype "entity".



4.2.1 ELEMENTS OF MDA PROCESS - FREEMDA SAMPLE

Let's step into details and let's have a deeper look at the diagram.

4.2.1.1 PARTITIONS

Architecture Development

Architecture Modeling is building and performing transformations and providing metamodels and profiles.

Metamodeling

Metamodeling includes the activities of modeling metamodels or profiles based on UML.

Modeling

the Modeling activities, Business Modeling and Conceptual Modeling

PIM

the Platform Independent Modeling activities

PSM

the Platform Specific Modeling activities

Transformation

Transformation generally deals with M2M- and M2T-Transformation.

4.2.1.2 ACTIONS

Step 1: Business Modeling

"Business Modeling" takes the application requirements and delivers some business UML models, for example business classes, processes and so on.

(0) The modeling language should be UML, for instance class diagrams, activity diagrams. Our tool is TOPCASED.

(1) Change to the TOPCASED perspective and go to the folder "model/source".

(2) Create a new model
"New > UML model with TOPCASED", "Create model" with "common approach" and "class diagram"

(3) or open an existing one "model/source/darlehen.umldi".

(4) model

(5) Before running the transformation there is some work on the PIM model left, for instance marking with stereotypes.
(5.1) Apply the UML profile "model/profile/UML2EJB-profile.uml" to the model.
(5.2) Add the available stereotype "entity" to the candidate classes.
(5.3) Put values to tags, if there are any.

Incoming Objects

darlehen

The result of Business Modeling is for example a business class diagram "model/source/darlehen.uml" and "model/source/darlehen.umldi".

UML

UML is our modeling language for Business Modeling, Model Transformation and for defining profiles and metamodels. The metamodel is provided by Eclipse plug-in "http://www.eclipse.org/uml2/2.1.0/UML".

If we would like to do some SimpleUML modeling, we would need the SimpleUML (or MyUML) metamodel as plug-in. But we don't have to do that. See the referenced SMDA project for it.

References:
- http://www.siegfried-nolte.de/forum/mda/smda.html

UMLEJB

UMLEJB ("model/profile/UMLEJB-Profil.uml" and "model/profile/UMLEJB-Profil.umldi") is a special profile, that provides at least (in our case) the stereotype "entity". Business classes may be marked as "entity", so we may recognize them as being entity bean candidates.

Outgoing Objects

darlehen

The result of Business Modeling is for example a business class diagram "model/source/darlehen.uml" and "model/source/darlehen.umldi".

Step 2: Transformation

The transformation action means building a conceptual UML model from a business UML model. The transformation is done with one of our QVT scripts UML2EJB.

(1) So for looking at the scripts, go to folder "qvt".

(2) Open them with double click or "Open with QVT editor" (Relations Language) or "Operational QVT editor" (Operational Mappings).

(3) Or try to make a new one for your own with simply copying an existing file:
- type "qvto" is for Operational QVT
- type "qvt" is for Relations (medini)
(You may create a new qvto or qvt file as well.)

(4) mediniQVT needs the metamodels being defined with "windows > Preferences > QVT Metamodels". You may select it from the "Registered Packages ...":

(5) For coding QVT there are two excellent books, that should help a bit - regrettably for German speaking folks only. Anyway, they and others will find several samples on my project site.

(6) Launching the transformation is done with a launch configuration. Change to Java or Acceleo perspective and check the "Run Configuration" options.
I've linked configuration images with this element. Follow the links for details.

References:
- file://../QVT-Relational.JPG
- http://www.siegfried-nolte.de/forum/mda/mdaproject/mda.html
- file://../QVT-Operational.JPG

Incoming Objects

UML2EJB

UML2EJB represents the QVT scripts
- "qvt/uml2ejb.qvt" for Relations
- "qvt/uml2ejb.qvto" for Operational

darlehen

The result of Business Modeling is for example a business class diagram "model/source/darlehen.uml" and "model/source/darlehen.umldi".

UMLEJB

UMLEJB ("model/profile/UMLEJB-Profil.uml" and "model/profile/UMLEJB-Profil.umldi") is a special profile, that provides at least (in our case) the stereotype "entity". Business classes may be marked as "entity", so we may recognize them as being entity bean candidates.

UML

UML is our modeling language for Business Modeling, Model Transformation and for defining profiles and metamodels. The metamodel is provided by Eclipse plug-in "http://www.eclipse.org/uml2/2.1.0/UML".

If we would like to do some SimpleUML modeling, we would need the SimpleUML (or MyUML) metamodel as plug-in. But we don't have to do that. See the referenced SMDA project for it.

References:
- http://www.siegfried-nolte.de/forum/mda/smda.html

Outgoing Objects

darlehen

The result of the M2M Transformation is a first conceptual UML class diagram for "model/result/darlehen.uml" and "model/result/darlehen.umldi". (It may be any model name, but it should best be "darlehen.uml" for the following prepared M2T transformation action -Step4.)

Step 3: Conceptual Modeling

"Conceptual Modeling" takes the generated UML/EJB models "target/darlehen.uml" (or any other of them) and delivers some conceptual EJB diagram.

For working with the conceptual model we need an UML diagram representation, again TOPCASED is the tool:

(1) "New > UML model with TOPCASED",
"Create from existing model"
with "Root Diagram: Class Diagram"
and "Initialize the diagram with existing model objects."

(2) model

(3) The result is a final "model/target/darlehen.uml" and "model/target/darlehen.umldi".

Incoming Objects

UMLEJB

UMLEJB ("model/profile/UMLEJB-Profil.uml" and "model/profile/UMLEJB-Profil.umldi") is a special profile, that provides at least (in our case) the stereotype "entity". Business classes may be marked as "entity", so we may recognize them as being entity bean candidates.

darlehen

The result of the M2M Transformation is a first conceptual UML class diagram for "model/result/darlehen.uml" and "model/result/darlehen.umldi". (It may be any model name, but it should best be "darlehen.uml" for the following prepared M2T transformation action -Step4.)

UML

UML is our modeling language for Business Modeling, Model Transformation and for defining profiles and metamodels. The metamodel is provided by Eclipse plug-in "http://www.eclipse.org/uml2/2.1.0/UML".

If we would like to do some SimpleUML modeling, we would need the SimpleUML (or MyUML) metamodel as plug-in. But we don't have to do that. See the referenced SMDA project for it.

References:

- http://www.siegfried-nolte.de/forum/mda/smda.html

Outgoing Objects

darlehen

The result of the M2M Transformation is a first conceptual UML class diagram for "model/result/darlehen.uml" and "model/result/darlehen.umldi". (It may be any model name, but it should best be "darlehen.uml" for the following prepared M2T transformation action -Step4.)

Step 4: Transformation

"MTL Transformation" does the model-to-text transformation using a M2T template language. The incoming model "target/darlehen" is UML or an instance of an UML profile.

It's best to work within an Acceleo perspective. You may work with the prepared MTL scripts, what is recommended, or create a new one. In the latter case you better should create a new Acceleo project.

(1) Create new MTL scripts: "New > Acceleo Template".
Create a new project: "New > Acceleo Module Project".
Read Acceleo help for more information.

(2) Open the template "src/Generator.files/generateJava.mtl"
with "Acceleo Template Editor" or more easily with double click.

(3) Write what you want to write. It should be MOF "Model-to-Text Transformation Language".
There are further modules doing some transformation job within "src/uml2java.common".

(4) Run the transformation with an Acceleo Application "MTL-UML2Java".
See the linked image for it.

(5) This should result in some "src_gen/darlehen/SB_*" packages.
The existing generated java-code will be complemented and not renewed.
So there may be some compile errors because of duplicates or like that.

References:
- file://../MTL-UML2Java.JPG

Incoming Objects

generateJava

"src/Generator.files/generateJava.mtl" is the model-to-text transformation script. The template language - maybe any template language, yet in this case - is MTL. There are further modules in the library "src/uml2java.common".

darlehen

The result of the M2M Transformation is a first conceptual UML class diagram for "model/result/darlehen.uml" and "model/result/darlehen.umldi". (It may be any model name, but it should best be "darlehen.uml" for the following prepared M2T transformation action -Step4.)

UMLEJB

UMLEJB ("model/profile/UMLEJB-Profil.uml" and "model/profile/UMLEJB-Profil.umldi") is a special profile, that provides at least (in our case) the stereotype "entity". Business classes may be marked as "entity", so we may recognize them as being entity bean candidates.

UML

UML is our modeling language for Business Modeling, Model Transformation and for defining profiles and metamodels. The metamodel is provided by Eclipse plug-in "http://www.eclipse.org/uml2/2.1.0/UML".

If we would like to do some SimpleUML modeling, we would need the SimpleUML (or MyUML) metamodel as plug-in. But we don't have to do that. See the referenced SMDA project for it.

References:

- http://www.siegfried-nolte.de/forum/mda/smda.html

Outgoing Objects

darlehen

"src_gen/darlehen/SB_*" represents the generated Session Bean packages.

If there are compile errors, delete the results and try again. Compile errors may result from duplicate generated code or the package names are different from the generated code.

Step x: Profile Development

"Profile Development" deals with modeling specific profiles based on UML. The result is an UML profile for further conceptual modeling.

"model/profile/UML2EJB-Profil.uml" and "model/profile/UML2EJB-Profil.umldi" demonstrate a really simple example a profile with a stereotype "entity", that has been modeled with TOPCASED: "New > UML Model with TOPCASED > Template: Profile Template".

Incoming Objects

UML

UML is our modeling language for Business Modeling, Model Transformation and for defining profiles and metamodels. The metamodel is provided by Eclipse plug-in "http://www.eclipse.org/uml2/2.1.0/UML".

If we would like to do some SimpleUML modeling, we would need the SimpleUML (or MyUML) metamodel as plug-in. But we don't have to do that. See the referenced SMDA project for it.

References:

- http://www.siegfried-nolte.de/forum/mda/smda.html

Outgoing Objects

UMLEJB

UMLEJB ("model/profile/UMLEJB-Profil.uml" and "model/profile/UMLEJB-Profil.umldi") is a special profile, that provides at least (in our case) the stereotype "entity". Business classes may be marked as "entity", so we may recognize them as being entity bean candidates.

Step y: Metamodeling

"Metamodeling" is the process for defining our own modeling language. Metamodeling generally means providing metamodels for other modeling languages besides UML.

(1) Metamodels are developed with UML class diagrams.
Let's see for example the SimpleUML metamodels modeled with
- Eclipse UML Modeling Tools - "model/metamodel/UMLMDT":
"SimpleUML.umlclass_diagram" and "SimpleUML.uml"
- TOPCASED - "model/metamodel/UMLTC":
"SimpleUML.umldi" and "SimpleUML.uml"

(2) Eclipse provides another tool for modeling metamodels, the Ecore Diagram Editor. This one delivers metamodels in "Ecore"-representation - "model/metamodel/Ecore":
"SimpleUML.ecorediag" and "SimpleUML.ecore"

(3) Building a modeling editor for our language can be done with means of Eclipse Modeling Framework. For this activity we need an EMF model representation "SimpleUML.genmodel" of our metamodel, from which we may generate the modeling plug-in. Take a look at the referenced SMDA site for more detailed information.

References:
- http://www.siegfried-nolte.de/forum/mda/smda.html

Incoming Objects

UML

UML is our modeling language for Business Modeling, Model Transformation and for defining profiles and metamodels. The metamodel is provided by Eclipse plug-in
"http://www.eclipse.org/uml2/2.1.0/UML".

If we would like to do some SimpleUML modeling, we would need the SimpleUML (or MyUML) metamodel as plug-in. But we don't have to do that. See the referenced SMDA project for it.

References:
- http://www.siegfried-nolte.de/forum/mda/smda.html

Outgoing Objects

SimpleUML

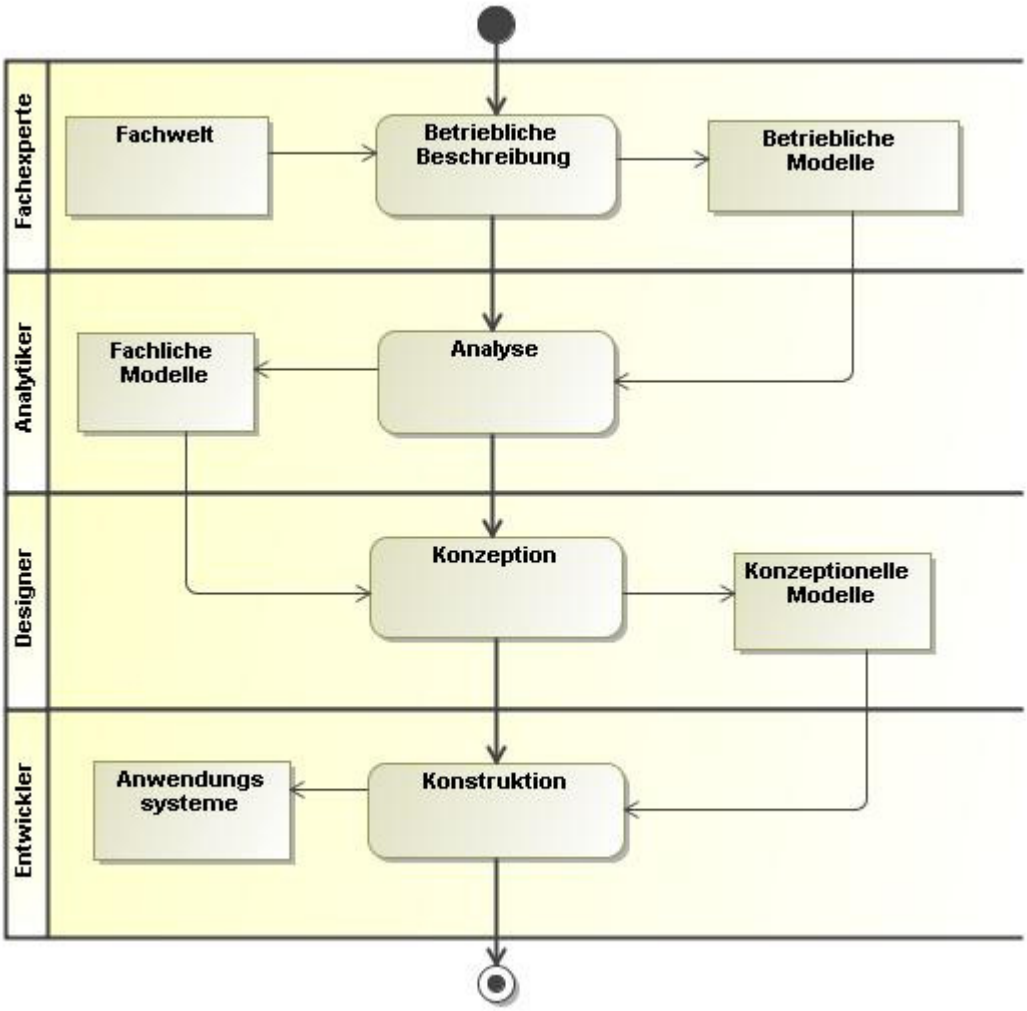
In our example we defined metamodels for a simple UML modeling language and a simple relational database modeling language:
- Eclipse UML Modeling Tools - "model/metamodel/UMLMDT":
"SimpleUML.umlclass_diagram" and "SimpleUML.uml"

- TOPCASED - "model/metamodel/UMLTC":
"SimpleUML.umldi" and "SimpleUML.uml"
- "Ecore"-representation - "model/metamodel/Ecore":
"SimpleUML.ecorediag" and "SimpleUML.ecore"

The modeling languages are represented by the editors "SimpleUML Model Editor" and "SimpleRDBM Model Editor", provided they has been integrated into our Eclipse workbench.
But, as mentioned before, we don't need it.

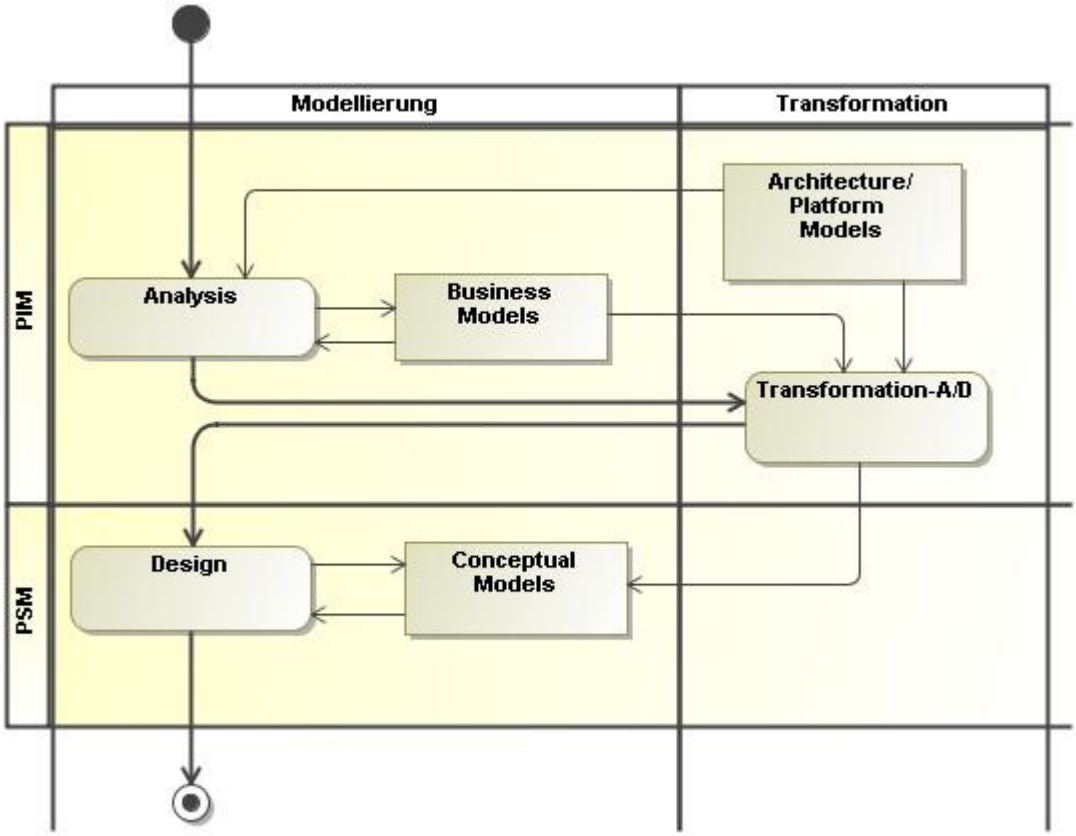
5 OTHER PROCESS VIEWS

5.1 1-SWE PROCESS - CLASSICAL VIEW



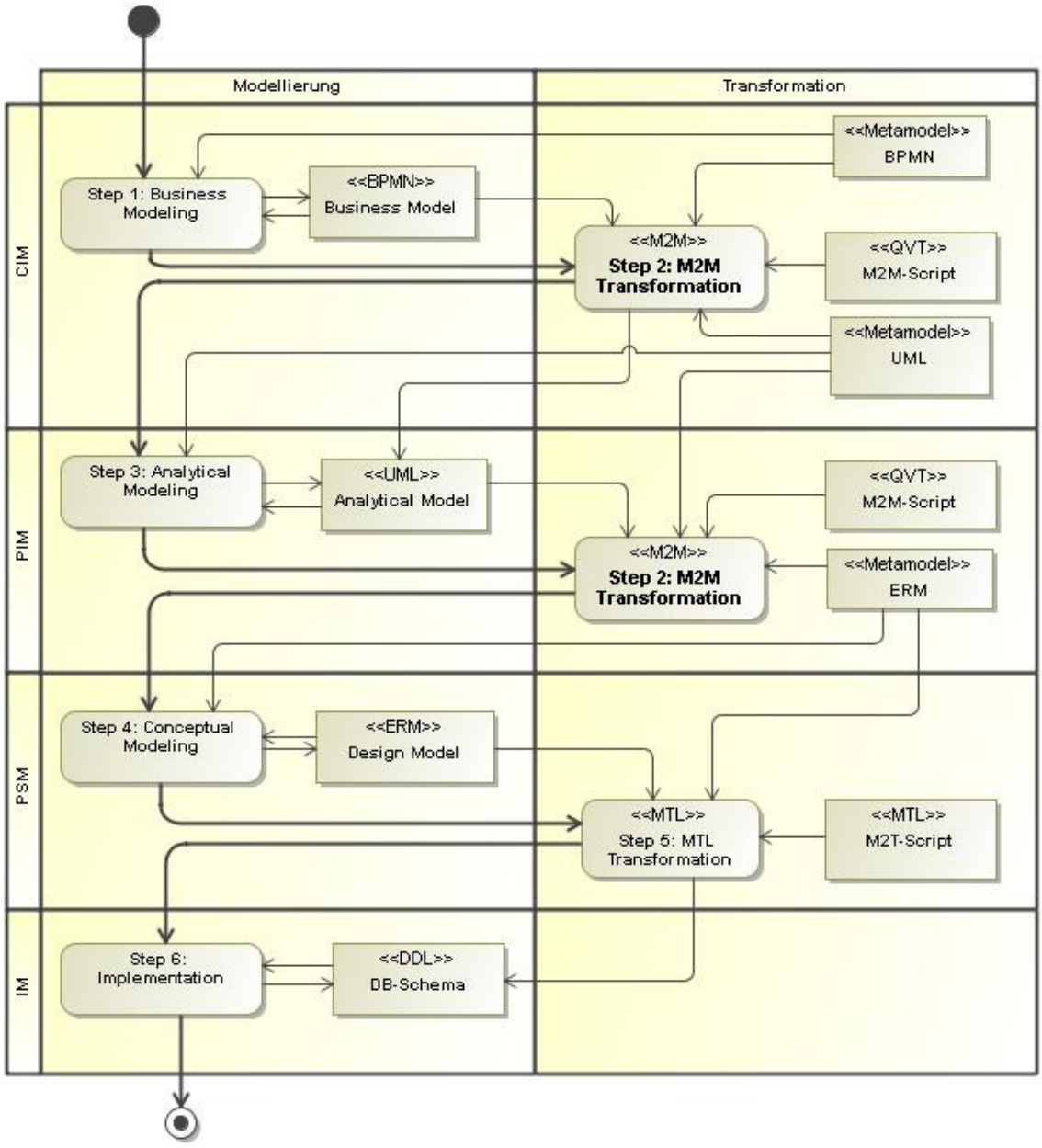
This is more or less the classical, maybe historical way of Software development, cascading from one process to another. The elements are with german names, so don't bother them to much.

5.2 2-MDA TRANSFORMATION PATTERN



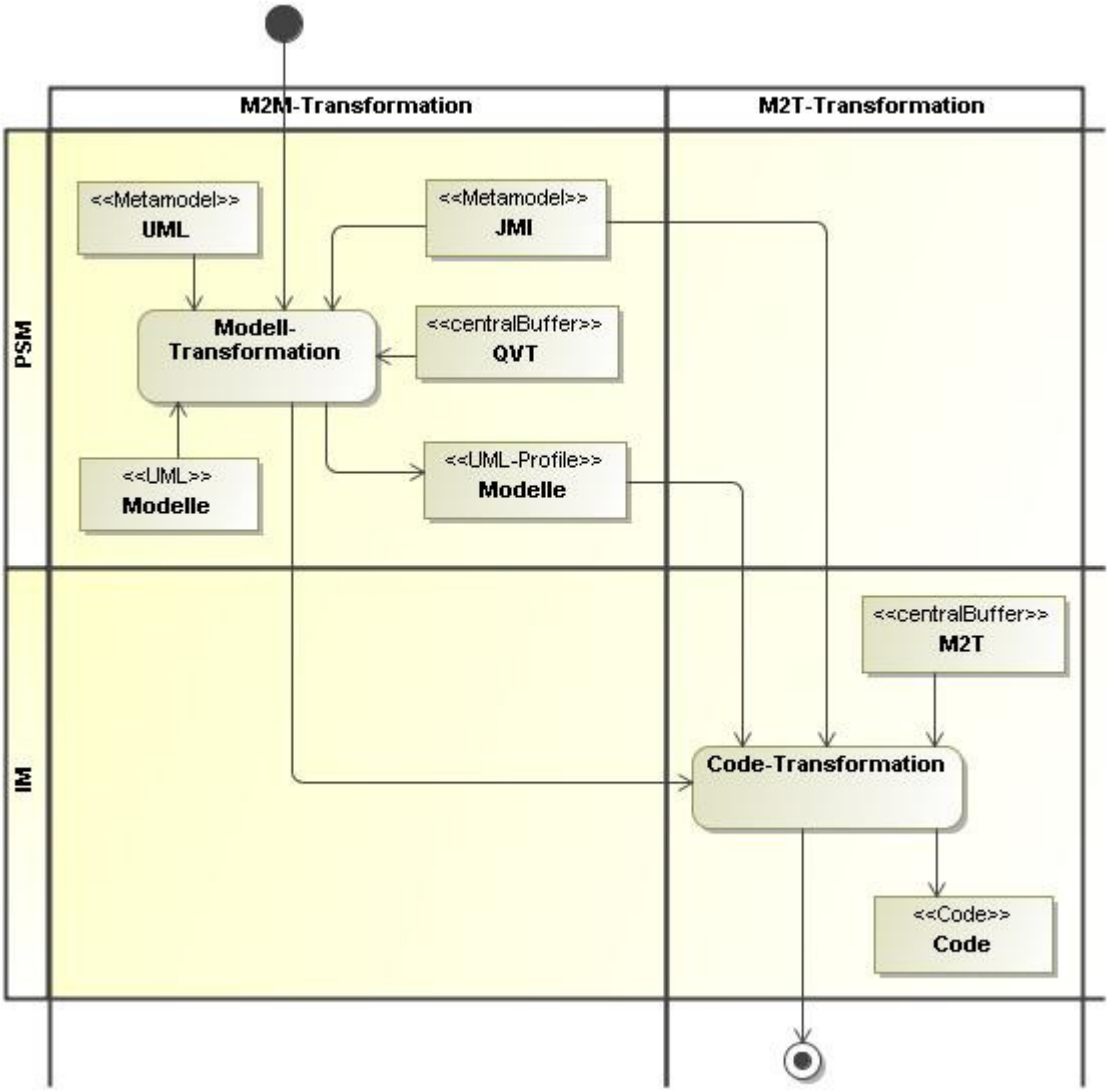
This is a sample for a MDA transformation pattern between PIM and PSM layer. Development is been done with changing between modeling and model transformation, in more general words: architectural modeling.

5.3 3-SAMPLE MDA PATTERN



This diagram shows the MDA development process across different development cycles. It's using the MDA pattern over all layers. The results of every lower layer define the platform of the above one.

5.4 4-M2M-M2T DIFFERENTIATION



"M2M-M2T-Differentiation" shows the difference between "model-to-model"-transformation (M2M) and "model-to-text"-transformation (M2T):
- The M2M takes a formal model, let's assume some UML model, and delivers another formal model, in this case a JMI model. The transformation language is QVT.
- The M2T takes a formal model, for instance a JMI model, and delivers some text file, Java code or that like. So M2T is code or text generation.