Component-Based Software Engineering	**Exercise Sheet No. 4**
Dipl.-Inf. Florian Heidenreich Office Hours: Wednesdays, 1430–1530 hours INF 2080 `http://st.inf.tu-dresden.de/teaching/cbse`	Software Technology Group Institute for Software and Multimedia Technology Department of Computer Science Technische Universität Dresden 01062 Dresden

Metamodelling with Eclipse EMF/Ecore

Task 1: Metamodelling with Eclipse EMF/Ecore (10 Points)

We have discussed the OMG Meta Object Facility (MOF) in the lecture on Metamodelling and Metaprogramming. Another metalanguage that is widely used for modelling is Ecore from the Eclipse Modeling Framework (EMF).

$\boxed{\text{1a)}}$ Install Eclipse and the Eclipse Modelling Framework and get used to its concepts and its overall structure.

Eclipse and EMF can be found at `http://www.eclipse.org/downloads/` where you can download the *Eclipse Modeling Tools*, which is an Eclipse distribution directly targeted to modelling.

$\boxed{\text{1b)}}$ EMF/Ecore can be used for modelling domain-specific modelling languages (DSMLs) that are targeted to a specific audience and capture domain knowledge in a more precise way than general-purpose modelling languages (GPMLs).

As an example, a forms language can be used for describing forms that are later filled in. A form usually has a title and consists of form-item groups. Form-item groups in turn consist of arbitrarily many form items. A form item has a question, an optional description and a certain type (e.g., free text, choice, option, decision, number, date). Form items can be dependent on options of form items (i.e., you have to fill in the form item iff a specific option has been selected).

Create an EMF/Ecore-based forms modelling language based on the description above.

Hint: There exist a plentitude of tutorials and how-to material that explains how you can create Ecore-based models with Eclipse. Make sure that you understand the generative facilities of EMF which serve you with an language-specific editor for your forms language.

Task 2: Using Metaclasses for Code Generation (10 Points)

In the lecture we have seen how metamodels and metamodelling can help you when transforming models from one representation to another. In this task, we will use the forms modelling language created in the previous task to develop code generators that allow for transformation of models described by the forms modelling language to textual representations (e.g., HTML, Text, PDF).

$\boxed{\text{2a)}}$ Make sure you have the Eclipse Modelling Workflow Engine (MWE) installed (formerly known as openArchitectureWare). Get familiar with its concepts and its overall structure. You are free to use other tools that allow for generating code from EMF/Ecore-based models, but I'll only present a solution that is based on MWE and Xpand in the exercise.

$\boxed{\text{2b)}}$ Using the Xpand template language from the Eclipse M2T project, you can generate arbitrary text from models. Create an MWE workflow that generates HTML-forms out of models described in your form modelling language. Use the language's meta-classes inside your code generator.

Task 3: Textual Concrete Syntax for Models (10 Points)

As you might have noticed while creating the actual forms models, using the EMF/Ecore tree-based editor is not always fun and not very productive. A domain specialist (i.e., a person who creates forms for a living) might prefer a more supporting representation for the language we have just developed. One option would be to create a graphical syntax (like, e.g., class diagrams are a graphical syntax for UML class models). But in this task we decide for a textual representation and build a textual concrete syntax—simply because the forms specialist loves working with text.

$\boxed{\text{3a)}}$ Install EMFText, a system for specifying textual concrete syntax for given metamodels and get familiar with its concepts and its overall structure.

EMFText can be found at `http://www.emftext.org`.

$\boxed{\text{3b)}}$ Define a textual syntax for your forms language in the CS specification language of EMFText. Generate the EMFText editor for your language and use it to create forms models. Inspect the models in EMF/Ecore's tree-based editor and compare both representation.

Hint: Make sure that you've watched the screencast that explains how EMFText is working.

$\boxed{\text{3c)}}$ What do you think are the advantages of treating text as a model?

$\boxed{\text{3d)}}$ * As an optional task, pick one of the languages from the EMFText Concrete Syntax Wish List, specify its syntax with EMFText and send the solution to `emftext-users@mail-st.inf.tu-dresden.de`.