TDT4250 H2024 **Assignment 2 – Criteria**

	Weight	30%	40%	20%	10%
Score		**Metamodel Completeness**	**Metamodel Consistency**	**Constraints and Derived Features**	**Project Setup**
		The extent to which the metamodel accurately represent the target domain.	***The extent to which the metamodel is consistent. The extent to which the metamodel prevents creating invalid models.***	***The extent to which constraints and derived features have been included in the metamodel.***	***The extent to which the Eclipse project is correctly organized. The extent to which the repository is correctly organized.***
BEST	**4**	Potentially all the classes and relations of the domain have been identified correctly. Concepts are organized in an articulated way, using inheritance, abstract classes, composition, and references in an appropriate way.	Perfectly consistent metamodel. All its instances are valid instances in the target domain. The provided model instances use the entire metamodel in a consistnent way.	At least one non-trivial constraint and one non-trivial derived feature defined in the metamodel and implemented in the Java code. Unit tests have been written for constraints and derived features.	The repository contains all the needed projects. The code compiles correctly. All the conventions and best practices have been followed.
GOOD	**3**	Most classes and relations of the domain have been identified correctly. Use of inheritance when needed. Composition and references are distinguished appropriately.	Most of the metamodel is consistent. Occasional minor consistency problems. The provided model instances are valid but use only a part of the metamodel.	At least one non-trivial constraint and one non-trivial derived feature defined in the metamodel and implemented in the Java code. A constraint or a derived feature is “non-trivial” when its implementation combines values extracted from multiple base features.	The repository contains all the needed projects. The code compiles correctly. The projects follows most of the conventions. The Ecore metamodel does not have an URI or package name. Some occasional compilation errors. The manual modified code is not correctly identified as such.
NORMAL	**2**	The metamodel contains a fair amount of classes and relations. Concepts that could be extracted as separate classes or subclasses are instead represented as set of attributes within a class.’	Multiplicity and types used consistently. Occasionally missing multiplicity and types. Wrong use of composition prevents creating correct instances of the metamodel. The provided model instances are not valid.	At least one simple constraint and one simple derived feature defined in the metamodel and implemented in the Java code.	The repository contains all the needed projects. The code compiles correctly. Naming conventions for projects have not been used. Naming conventions for Ecore have not been followed.
POOR	**1**	The metamodel contains classes with many attributes, but no relations between classes.	Some use of multiplicity and types. Still, many attributes are redundant and the same information is entered multiple times and can be inconsistent in metamodel instances.	Constraints and derived features have been specified in the model, but they have not been implemented.	The repository contains all the needed projects. Still the code does not compile.
WORST	**0**	The metamodel contains only a few classes and almost no relations.	The metamodel does not set multiplicity for attributes and references. Attributes don’t have a type. The same information is entered multiple times and can be inconsistent in metamodel instances.	No use of constraints neither derived features.	The repository contains only part of the developed Eclipse projects. It is not possible to import projects in Eclipse. Code does not compile. There is no generated code for the “model”, “edit” and “editor” plugins.