UDC 004.4:004.738.5 (043.2)

Shestakov K.O.
National Aviation University, Kyiv

UML-TO-UML SOFTWARE MODEL TRANSFORMATION

Model-Driven Software Engineering makes models key artifacts in the software development process. During the Analysis, Design and Development stages of a general software development lifecycle, a variety of models is created. However, in a majority of software development methodologies, models, like UML (Unified Modeling Language) models, are built mainly for the purposes of documentation. Apart from the comprehension of a software architecture and design, models can be used to facilitate the creation of software. This can be achieved by executing model transformations.

Usually, models are obtained by generating them through the analysis of an existing application, but in a proposed method the models that were drafted during the Analysis and Design stages serve as a source base for the generation of new models that in turn represent the architecture of a program that can be formed using popular model-to-code transformations (such as UML to CORBA IDL, UML to Java, UML to WSDL, UML to XSD etc.). Instead of creating a necessary view all the time, Model-Driven Development approach employs the semantic associations between different UML models, which makes it possible to obtain new models from given models by means of enforcing specified model transformation rules. This approach saves time and effort needed to create the model and positively affects the overall project performance since there is more resources to invest into accomplishing more pressing tasks in the project.

Even though an effort needs to be applied to figure out the rules of required model transformations, that are suitable for a specific project (since every project is unique in its implementation), this effort is mostly fixed and the obtained results can be reused for further application development. Moreover, qualified professionals with many years of experience on a particular project already have the comprehension of a designed system and therefore might have ideas about appropriate relationships between different system representations, that is, models.

In this particular example, a transformation of a Use Case diagram into a Class diagram can be considered. The tool that facilitates model creation and visualization is Rational Software Architect. It creates EMX files that store the model and diagrams in XML Metadata Interchange format. The tool that facilitates model-to-model transformation is MediniQVT. It takes source and target models and performs a transformation based on defined QVT script. QVT script is the script that contains the rules which determine how to perform a transformation. The rules are represented as transformations that consists of a set of relations defined using QVT-R language (and additionally Object Constraint Language). QVT-R stands for Query/View/Transformation-Relations and it is the standard language proposed by the Object Management Group to specify both unidirectional and bidirectional model transformations.

The process of UML-to-UML software model transformation can be implemented by the following exemplary description. At first, the Use Case diagram is drawn in Rational Software Architect. This is how a source model is obtained. Then this model is prepared for usage in MediniQVT. For the transformation to happen, a QVT-R script must be present, that describes how to transform a Use Case model into a Class model. The rules are applied to a source model to obtain a target model. The target model is then prepared for usage in Rational Software Architect. To visualize obtained Class model, a class diagram is created in this model and generated class model elements are put onto the drawing surface. As a result, a class diagram was obtained that has classes with operations that correspond to actors linked with use cases using associations in a use case diagram.

In conclusion, UML-to-UML software model transformation is a technique that could advance the software development process by emphasizing the application of software models for facilitated code writing in a Model-Driven Development.

Scientific supervisor: Babiy G.V.,
Senior Lecturer

UDC 336.764.2(477) (043.2)

Skorliaieva K.V.
National Aviation University, Kyiv

HOW BINARY OPTIONS ARE PERCEIVED IN UKRAINE

During recent years, binary options (digital options) are becoming more popular as a format of financial trading.

All-or-nothing options (another name for binary options) generate either profit or nothing. Thus, the profit is fixed and such options are known as options with fixed profit (FROs-Fixed Return Options). Binary options allow you to know exactly the amount of benefits and risks before a contract is set up, that enables easy control of a large number of commercial transactions.

Binary options are similar to traditional options, in the sense that it`s payouts depend on the price of the underlying asset at the end of the contract validity. However, in binary options the value has only the trend of asset`s price change, but not the magnitude of the change.

The main difference between binary options and traditional options is the scope of the potential profit or loss. Binary options are also known as fixed return options