Metamodel-Based Modeling and Model Transformation Framework Supporting Inheritance and Constrants

László Lengyel, Tihamér Levendovszky and Hassan Charaf

Because of the appearance of high level languages, object-oriented technologies and wide spreading of CASE tools metamodeling becomes more and more important. Metamodeling is one of the most central techniques both in design of visual languages, and reuse existing domains by extending the metamodel level. Visual modeling has become indispensable part of software engineering, with aim to raise efficiency in design period of software engineering. Visual modeling helps to realize systems at a higher abstraction level. It is important to provide solutions, which supports storing models building on each other, and which can handle models uniformly through optional number of layers.

Our implementation called AGSI (Attributed Graph Architecture Supporting Inheritance) is a multipurpose modeling and transformation system, which is able to store models, check them against their metamodel, traverse and transform them using graph rewriting (transformation) rules with tree-based notation, export and import them. In AGSI every model has its metamodel and every model can be the metamodel of its instances. The most important principle in AGSI is the following: traversing and modifying the layers must be transparent; every layer must be handled with the same functions without behaving in a layer-dependent way. In addition to these achievements in metamodel-based graph transformation, this presentation introduces how we extend this concept to support constraint handling and constraint propagation using Object Constraint Language (OCL).

In graph rewriting we have rewriting rules consisting of the left hand side graph (LHS) and right hand side graph (RHS). Applying a graph rewriting rule means finding an isomorphic occurrence (match) of the LHS in the graph the rule being applied to (host graph), and replacing this subgraph with RHS. Replacing means removing elements which are in the LHS but not in RHS, and adding elements which are in RHS but not in LHS. AGSI goes further: we have LHS built from metamodel elements, so an instantiation of LHS must be found in the host graph instead of an isomorphic subgraph. Causality is a relation between LHS and RHS elements, it makes possible to connect an LHS element to an RHS element and to assign an operation to this connection. Causality and its operation describe what must be accomplished during the application of a rewriting rule. It is possible to provide an ordering of the rewriting rules, in other words we control the transformation process by sequencing the rewriting rules. Searching is one of the key algorithms, AGSI always tries to use metamodel-based searching involving node type examination, to accelerate the rewriting algorithm. If it is not possible, because rewriting rule does not contain type information then the algorithm combines depth-first and breath-first searching strategies. Metamodel-based searching means, that we start searching in metamodel, firstly we identify the searched nodes by their type. And then we can immediately obtain an occurrence of the searched nodes in the model by model-metamodel relation. This immediate node identifying - which means only one database select - significantly accelerates the searching algorithm.

This presentation discusses the properties of the constraints and the implementation of their special key classes in detail. These constraints specified on meta-layer restrict the modeling possibilities on instance layer. OCL is a formal language used to express constraints. OCL considers the topology and requires the examination of the constraint propagation as well. It is possible to use OCL for different purposes: (i) to specify invariants on classes and types in the class model, (ii) to describe pre- and post conditions on operations, (iii) to describe guards (iv) and as a navigation language. It is shown how constraints can be used for rewriting rules and how they affect the transformation properties.

An illustrative case study based on constraint specification in rewriting rules is also provided.