# Reading Report on VMTS

Dylan Kiss

*University of Antwerp*
*Middelheimlaan 1*
*2020 Antwerp*
*dylan.kiss@student.uantwerpen.be*

**Abstract**

In this reading report I will give a summary of what I read about VMTS (Levendovszky et al., 2005), the Visual Modeling and Transformation System, and give an overview of what I plan to do in my project.

*Keywords:* VMTS, metamodeling, metamodel-based transformations

## 1. Introduction

The Visual Modeling and Transformation System (VMTS) is a model editor, storage and transformation software package. It uses an approach in which model storage and model transformation can be treated uniformly, linked together by the notion of the metamodel. This metamodel-based approach provides a highly configurable visual modeling tool. Model transformations in VMTS can be used for model and code generation as well as for modifying models. VMTS makes use of XML for the representation of its (meta)models. It has been implemented in .NET using C#, which restricts its use to the Windows operating system.

## 2. Metamodeling environment

VMTS is an *n-layer* metamodeling environment. It uses a simplified UML class diagram as a metamodel language to define models. If the UML class diagram is instantiated, there are three layers involved: the UML object diagram, the UML class diagram and the metamodel of the UML class diagram. Besides that, VMTS says it has two more layers: the read-only meta-metamodel which specifies the metamodeling language, and the one in the internal structure (a labeled directed graph). The model storage part of VMTS is called *Attributed Graph Architecture Supporting Inheritance*

(AGSI). AGSI layers are designed such that every model can be a metamodel for others.

AGSI provides three basic graph constructs: nodes, directed edges and labels assigned to nodes and edges. In order to use this for metamodeling, some things need to be added. A first thing is that each node and edge holds a bidirectional connection to other nodes and edges, respectively: this is the type-instance mapping. A second thing is support for containment hierarchy through a parent-child bidirectional mapping. A third thing is inheritance support through a directed mapping from the descendants to the ancestors. A last thing is support for association classes through the use of pseudo-nodes with no semantic meaning.

AGSI stores the model attributes (labels in the directed graph) in an XMI-like format. Meta-attributes, which can be instantiated, are converted to an XSD file that provides the schema for the XML file storing the attributes on the instance level.

## 3. Model transformations

The simplest method to transform models is to traverse them using a specific programming language and changing appropriate parts of the input models or producing an output model. Traversing Model Processors (TMP) offer five basic graph operations for this: create node, connect nodes, delete node, delete edge and set label. The models and their elements in VMTS traversing processors are regular objects in an object oriented programming language. Creating a node is done by creating a new object with a specific type; deleting a node is done by destroying the object; attributes are set via member variables with corresponding attribute names; deleting an edge is done by removing references between objects; adding an edge the other way round.

Visual Model Processors (VMP) provide a visual alternative way of model transformation. In VMTS, VMP use graph rewriting as the transformation technique. A set of rules is defined consisting of a left hand side graph (LHS) and right hand side graph (RHS). Firing such a rule searches for an occurrence of the LHS in the host graph and replaces that subgraph with the RHS. In VMTS, the rewriting rules are specified in terms of the metamodel. As such, instead of finding a direct occurrence of the LHS, a part of the input model must be found which instantiates the LHS. Attribute transformation in VMTS is accomplished by XSLT scripts, since attributes are stored as XML files.

## 4. Planned work

In my project I will explore the capabilities of VMTS by modeling a role-playing game. In particular, this will consist of creating an abstract and concrete syntax for the RPG formalism and performing possible transformations on such a model to define operational and/or denotational semantics. I will compare this to my experiences in AToMPM and state the advantages and disadvantages of VMTS.

Levendovszky, T., Lengyel, L., Mezei, G., Charaf, H., 2005. A Systematic Approach to Metamodeling Environments and Model Transformation Systems in VMTS. Electronic Notes in Theoretical Computer Science 127 (1), 65–75.