Applying Model-driven Generative Programming to Communication Network Performance Evaluation
Amogh Kavimandan and Aniruddha Gokhale
Institute for Software Integrated Systems (ISIS)
Dept. of Electrical Engineering and Computer Science
Vanderbilt University Nashville 37235 TN
{amoghk, gokhale@dre.vanderbilt.edu}

Abstract.
Last few years have seen tremendous growth in large-scale, distributed, network applications across the internet. This has lead to considerable efforts in the area of mathematical analysis and simulation of communication networks \[1, 3, 4, 5, 6, 7, 8\]. Simulation-based study of a network is helpful to service providers (and researchers) for understanding resource utilization and service quality prior to final deployment, while analytical treatment provides algorithmic bounds on the feasible performance. A number of modeling and simulation tools exist to address this cause, however, no single tool is sufficient for a comprehensive study, which includes all aspects of performance analysis \[13\]. Further, since models written in these tools are textual, they are (1) not intuitive and (2) difficult to manipulate, since even small changes in the network topology or resource allocations introduce error prone, handcrafted changes to the textual descriptions. This paper describes the use of model-driven generative programming to address these challenges. First, we discuss the architecture and design of a domain specific modeling language called the Communication Network Modeling Language (ComNetML). ComNetML provides intuitive visual modeling artifacts for easy-to-use model driven development and synthesis environment for network simulation and analysis. Second, we illustrate how ComNetML decouples the back end simulation and/or analysis tool from the visual front end, while using generative programming to synthesize artifacts specific to the back end tool. The power of the tool is inherent in its capability to allow changes made to the models in the front-end from which the right artifacts are synthesized for the back end tools.

1. Introduction.
Modeling of communication networks for performance analysis and resource utilization is an area of considerable interest, owing to the ever increasing deployment of distributed applications, such as the internet-based m-commerce, peer-to-peer systems, and grid applications among others. Modeling of any medium to large network is necessary before the final deployment can be done. This helps in determining the optimal utilization and provisioning of limited network resources, such as routers, network bandwidth and end nodes, before it can be commissioned for service. Unless resources are optimally provisioned, the network applications and end users will experience deteriorated service and violations of service level agreements. Modeling the system is therefore of considerable importance for monitoring the health of network and for fault detection, even after it has been deployed.

Several tools exist for modeling and performance of networks. These can be categorized \[9\] mainly into (1) Simulation tools, such as the discrete event driven tools like NS-2 \[3\]. Event based tools are those in which an arrival or departure of a packet is treated as an event, thus the packets are tracked as they enter and leave network of interest; and (2) analytical tools, which perform mathematical and/or queuing theoretic analysis of network systems. Analytical tools are those which analyze a network mathematically and not necessarily depict the dynamic behavior of flows.

Each category of tool is thus useful for a specific network study. For example, if the space, time of simulation and memory are not a constraint then tools like NS-2 should be used for simulation. On the other hand, if the time is a constraint and we need only study the static behavior, then an analytical tool will be more appropriate since it is more suited for steady state study while still providing accuracy within acceptable limits. Thus, it is desirable to use a variety of tools to study network behavior and identify bottlenecks so that appropriate resource provisioning could be made.

Thus, no single performance evaluation tool is thus sufficient but a combination of tools is necessary for complete analysis of a given network. Topology generation, for example, is a non-trivial problem owing to the fact that each simulation tool uses a different language for representing network entities. Service providers use a number of tools during the network design and maintenance phase and have to make changes in the topology and study their effect on overall network performance. Any changes in topology will then have to be reflected into the inputs for each of the platform specific artifacts for these tools. Additionally, most of these tools are textual, which means that changes in topology are often tedious and error prone to incorporate manually. This forms the motivation for designing a graphical, domain-specific, multi-aspect, hierarchical modeling language for the study of communication network performance evaluation.

2. ComNetML Framework.
In this section we briefly introduce ComNetML, which is a domain-specific modeling language we have developed to describe network topologies and interactions for study of networks. ComNetML has
been developed using the Generic Modeling Environment (GME) visual language development environment. Figure 1 illustrates the ComNetML architecture. We then describe the various design choices we have made in the design of ComNetML.

2.1 Service Provider ComNetML Framework View.
At a higher level of abstraction, a service provider views the ComNetML framework as follows. The Modeler sees ComNetML framework as consisting of two high level blocks:

1. The Graphical Model Representation block, which is helpful in describing the network topology. This is the GME based modeling block shown in the figure, which defines all the network elements and their associations for a topology.
2. The second block, Graphical Model Interpretation, interprets the model developed in ComNetML and generates the target back end tool artifacts, such as scripts and topology for NS-2. The model interpreter synthesizes output that follows the syntax and semantics compliant with the target back end tool. These artifacts are used to drive the simulation on the target simulator. ComNetML allows a feedback path from the simulator, and the modeler may choose to modify the ComNetML model with minimal modeling effort and the whole process is repeated till optimal deployment and resource provisioning is designed for the given network topology.

The above framework is based on the principles of Model Integrated Computing, which supports application level evolution via the Multigraph Architecture. The application evolves as the feedback is received from the back end tool and the process is iterative. Thus, as the end of several cycles the application is ready for deployment since the resources specified in original model have been optimized for the topology under consideration.

2.2 Syntactic and Semantic Elements of ComNetML.
ComNetML is a graphical language developed within GME. In GME, a modeling language is defined in terms of meta-models that capture the abstract syntax of the language. Figure 2 shows the meta-model of ComNetML in GME. It consists mainly of three sub-diagrams for realizing each mapping described in the earlier sections. The diagram follows GME conventions, for example, a root folder contains all the models and atoms in the topology. Cross-references to an element in one diagram to another diagram are allowed by means of Proxy as shown in the diagram e.g., network_proxy shown in the figure. The network model is instantiated in diagram (B), while it is cross-referenced in diagram (A) using ModelProxy.

In the following section we describe the meta-model diagrams shown in Figure 3. For each diagram, there are a number of classes with a number of attributes. The attributes also follow the GME convention meaning their type is derived from the type hierarchy for attributes in GME. One observation here is that the meta-model captures abstract syntax for the modeling language. As such, some components for example, the link may appear to be graphical but their semantics are tied to the semantics of the underlying meta-modeling language. The visualization of each of these elements as such, is governed by the GME.

2.2.1 Topology view.
The top-level model network is contained in the root folder. network is semantically equivalent to a network topology in a simulation environment. They are hierarchical in that they contain a router, iointerface and event models. The link connection defines the relation between two routers, which can occur only though the use of iointerface contained in the router. The link is an association class that associates ports of routers. For more explanation the reader is referred to the GME documentation.

2.2.2 Events view.
The event class contains a number of attributes and is contained in the cross-referenced network model. The simulation_begin and simulation_end atoms are connected to events using number of connections. A GME entity can be connected to other, using a connection and connector, indicated by a stereotype <<connection>> and a dot.
2.2.3 Flow view.

The flow model is contained in network (cross-referenced model) and contains a number of attributes as shown.

3. Conclusion.

We have introduced ComNetML, a development and synthesis framework for communication networks. The framework provides a graphical language for describing a given network topology and generate platform specific simulation artifacts. ComNetML has shown great promise towards the goal of facilitating a platform independent, easy-to-use, hierarchical modeling language for network analysis.

References.