Modeling Accelerates Optical Networking System Development

by

Jay Case
Staff Engineer – Optical Networking Group
Tellabsjcase@tellabs.com
www.tellabs.com

and

John R. Wolfe
President & CEO
Project Technology, Inc.
john.wolfe@projtech.com
www.projtech.com

In a traditional embedded system design process, the software and hardware teams begin with a set of requirements. The two teams then separate and implement their interpretation of the agreed requirements. Subsequently, the two teams then work out their different interpretations during prototyping, which frequently involves at least two or three significant design iterations to make a system work. The prototyping costs are expensive, but perhaps more significantly, traditional embedded projects experience significant delays in time-to-market.

By 1996, it was already clear that telecommunications demand projections were skyrocketing due to accelerated Internet traffic, exploding cell phone usage, and increased long-distance calling by consumers and businesses. As a producer of next generation optical networking equipment that also provides broadband access, Tellabs elected to build an advanced multi-service transport switch for large central hubs. Target customers for the new equipment included large telephone companies and network service providers.

At that time, a design team at the company embarked on a product development project to build a new, protocol-independent central office networking switch system called the TITAN® 6500 Multiservice Transport Switch (MTS). The fiber optic digital cross-connect switch would provide a much-needed solution for large communication hubs to groom and tailor bandwidth allocations to different outbound cable requirements.

Realizing that the design process for such a system would be a multi-year project, the team sought a design methodology that would help manage the development process and speed time-to-market. One key objective of the project was to eliminate the costly, time-consuming redesign steps during the integration of hardware and software at the prototyping stage. To help meet this objective and achieve optimum time-to-market, Tellabs selected the BridgePoint® tool set from Project Technology. The tool set

provides an advanced, executable-modeling environment that utilizes Object Oriented Analysis (OOA) for generating target-independent source code from models. The tool set allows analysts to use either Unified Modeling Language (UML) or Shlaer-Mellor (SM) notation.

System Expertise vs. Programming

In development organizations, a spectrum of skills exists across the team. On one end of the spectrum are subject-matter experts, people who best understand the requirements of the customer. The other end is comprised of technical experts who focus on the implementation and sometimes have little or no knowledge of the application itself. Most conventional programming projects require team members to span this entire range by writing application-specific code in a particular target language for a specific operating system.

On this project, however, the system was built quite differently. Through the support of the modeling tools, the team could take advantage of the experts that existed on both ends of the spectrum by allowing them to focus on their areas of expertise. The 40 subject matter experts, who also developed the earlier iteration of the switch system, used the modeling tools to abstract the invariant subject material prior to any hardware definitions. The technical programmers were then able to concentrate on building a model compiler to turn the application models into optimized C++ for pSOS™ and Solaris™.

When completed, the product included over 2,000 advanced PowerPC® processors on a fully loaded TITAN 6500 MTS. To manage the enormity of this task, the system was organized into 14 separate subjects, or domains. With such a complex system, the analysts relied on the tools to connect and synchronize these multiple domains via bridges.

Modeling Over 6,000 Terminations

Initially, the design team faced a challenge of how to describe a digital cross-connect system, regardless of any underlying protocols or technologies. Further, the target processors were not available until the final years of the project.

The design required that the fiber optic switch system handle the connections for over 6,000 terminations with varying incoming and outgoing bandwidths and transmission protocols. The device needed to switch inbound and outbound connections from these terminations to multiple destination cities. It's important to note that bandwidth requirements vary between inbound and outbound cables to and from destinations.

Due to the dynamics of this environment, the first task was to define what it meant to switch bandwidth and groom traffic (meaning tailor and shape the signal to meet the bandwidth requirements of the outbound termination). Then the system analysts used the tools to create models that simulated the needed behavior of the as yet specified target processors. The results allowed the hardware and software teams to concurrently develop a system independent of the final platform.

Building a Fault-Tolerant System

The target customers for these switches, large service providers, must provide continuous quality of service, regardless of network conditions. For this reason, one of the original system requirements was a redundant fault-tolerant architecture. Although easily stated on paper as a requirement, the design team needed to examine and develop a feasible fault-tolerant methodology for such a complex switch. Further, the approach needed to account for both equipment (internal) and facility (external) failure modes.

To describe this more simply, imagine a call originating in Chicago and routed on an optical termination to Philadelphia. Now, suppose a snowplow in Philadelphia breaks the cable. The system was required to react to the facility line breakage and continue the thousands of conversations without any of the callers even noticing a service "hiccup" in their conversation. The analysts needed to preserve the rules applicable to normal operation, yet they had to find ways to model the unexpected external events, such as the snowplow.

After much consideration, the design process split into two development paths. One development path focused on modeling the normal system. The other development path used the modeling approach to implement methods that provided almost simultaneous re-initialization and recovery. In this design process, the idea was to correctly model normal system behavior first, then account for potential service interruptions. This permitted the majority of the system analysts to focus on normal system behavior.

Fig. 1 provides a generic view of the modeling method.

Test Procedures

For the first three years, the analysts were able to execute the models, or simulate the system, to test and verify the behavior without requiring target hardware. The modeling tools then identified any behavioral failures (called "exceptions") that subsequently needed to be corrected.

The modeling tools first detect these exceptions in the state space and then provide the analysts with an opportunity to choose the correct execution profile (also referred to as "threads" or "paths"). Working strictly with the models, the behavior of the application was fully tested and corrected without relation to the eventual hardware platform. This capability was crucial in achieving final system transmission connections for the thousands of terminations.

When the analysts compiled the models and tested the automatically generated code with the simulated hardware, they were able to verify full system behavior. Being able to simulate the entire system, the analysts tested a then current model against two different test scenarios, one with a known outcome. Model execution allowed the analysts to test the fault-tolerant design requirement and observe the results in real time. If exceptions occurred, the analysts went back to the models and identified where the behavior needed to be remodeled. This methodology is analogous to advanced regression testing with the

difference being that the model is regression tested without the need for hardware availability.

This capability was also particularly useful for adding new features. By using the existing models, the analysts could either edit current features or add new features and simply execute the new models. When the analyst executed the models, the modeling tools again identified any exceptions that needed to be corrected before being compiled into code. This function alone allowed the design team to rapidly implement new feature sets and upgrades.

Summary

When completed and introduced in the fall of 2000, the proprietary model compiler automatically generated source code from the models for operation on both Solaris and pSOS operating systems. The design team succeeded in producing a fiber optic digital cross connect switch that handles varying bandwidth requirements for over 6,000 terminations. The design process also generated a fault-tolerant design such that even a broken cable or hardware failure in the system results in a seamless service transition for the telephony and data user.

From the application models, the model compiler generated approximately four million lines (excluding comments) of well-documented and beautifully formatted C++, using only POSIX-compliant system calls. The build environment generated 16 separate executables for the Solaris and pSOS operating systems.

The modeling tools also make it easy for the design team to multiply the system capacity to accommodate growth, both planned and the unexpected. The design team can readily configure the present generation model to multiply the terminations without changing the analysis models, a significant reduction in future development costs for subsequent generations of the switch.

The tools themselves offered an extensive range of benefits that enabled system development to run smoothly. From adding feature sets to establishing fault-tolerance, the modeling tools provided the wherewithal to easily manage the project's complexity and bring this product to market faster. The modeling tools also allowed the hardware and software teams to simultaneously execute models at each step of the project, perhaps the biggest advantage of all.

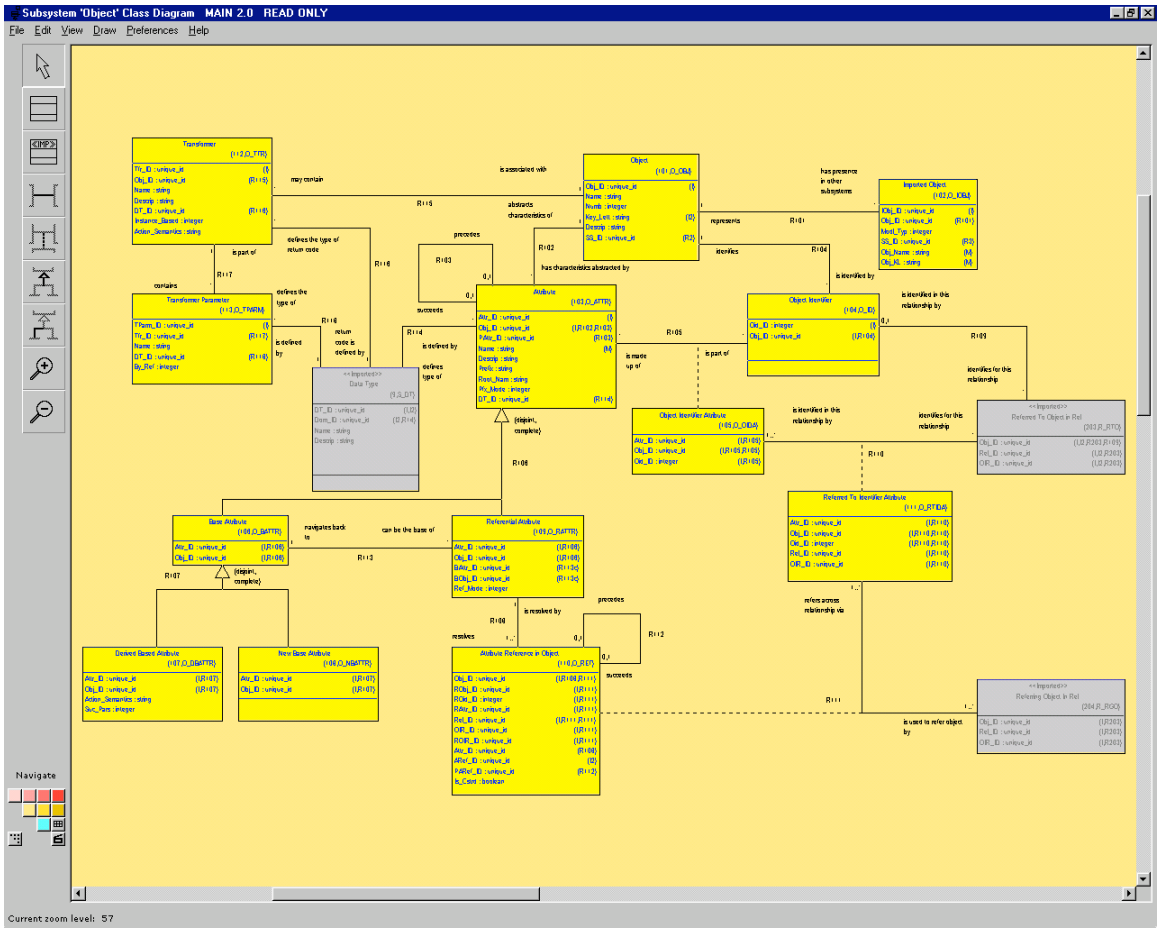

Figure 1: The components of the BridgePoint® Modeling Suite shown above provided the framework for development of the TITAN® 6500 Multiservice Transport Switch.

* * * * * END * * * * *

Word Count: 1,577