Integrating Performance Evaluation and Formal Specification

J. Martins* martins@tcom.epfl.ch
J.-P. Hubaux* hubaux@tcom.epfl.ch
T. Saydam◆ saydam@udel.edu
S. Znaty* znaty@tcom.epfl.ch

Abstract

We propose a methodology that intends to reuse formal specification effort to build a performance modeling. The methodology starts abstracting the relevant features of a real system in a formal specification. Then, we enhance the specification with performance information (quality of service, workload, processing design). At that point, we map the enhanced formal representation in a performance modeling that preserves the formal properties. Thereupon, we implement this modeling in a performance evaluation environment. We develop the methodology for the SDL and Estelle standardized formal techniques. Ultimately, we illustrate the methodology by an example: the Transport Control Protocol (TCP). Finally, we simulate the achieved executable, varying evaluation conditions.

1. Introduction

The situation of telecommunications systems is rapidly evolving towards a state characterized by the presence of multitude of services integrated in a single network structure. In order to ease their integration, it is important to provide unambiguous descriptions of the organization, of the functionality, and of the relationships existing among services. Since it is not possible to completely foresee the situation of services in the future, descriptions should be modular, so that the task of adding new services to existing environments is simplified. Formal description techniques are languages designed to handle such descriptions. Their goal is to produce unambiguous and correct specifications, understanding and modeling the system and the domain within which it operates.

On the other hand, the throughput of physical networks is constantly growing, reaching the order of Gbit/s. In the meantime, services needing high throughput are being designed (e.g., multimedia systems), creating a need for flexible, high performance communications systems. An important feature of new high speed services is their increased need for guarantees. Thus, the use of powerful performance modeling techniques becomes of prime importance. Their goal is to obtain some notion of how the system will perform under a given set of conditions. They facilitate analysis of proposals for new services or technology since they are more tractable and less expensive than field trials for evaluating alternatives.

Performance evaluation theorists have long been using system specifications to make a heuristic model of the system and then analyzing the model. When system specifications exist, the question arises whether a more direct path to performance analysis can be found. Our scope is to provide a methodology that consistently integrates formal specification and performance evaluation. Our methodology appeal consists of benefiting of the rigorous mathematical basis from a formal method (that produce high integrity systems) to build a performance modeling, saving time and avoiding functional errors.

Much effort has been invested during the last decade in the area of integrating performance modeling and formal specification. In [2], authors suggest how time and branching probability might be added to some standardized formal languages. In [3, 9], authors develop models combining Estelle and queueing networks. In [1, 6], methods for the integration of SDL and queueing networks are proposed. A combined formal description/performance evaluation model is implemented by means of interacting processes. Other interesting papers are [4, 5, 7, 8, 9, 11].

2. The Methodology

The purpose of a formal specification is to make sure that the information transfer between the producer of the specification and the user is smooth and unambiguous. Processing validation ensures that the model is error free and can then be used to elaborate implementations using an automatic code generation tool. A problem is that many of the bottlenecks and impediments to smooth data flow do with design issues not directly dealt in the system specification. Thus, a formal specification should be enhanced with the relevant performance information. It is essential to conserve the original semantics, and extend the semantics and syntax for new concepts.

As formal specification techniques are not intended for execution, we need to translate the extended formal model into a performance modeling adapting the ideal conditions

* Swiss Federal Institute of Technology,TCOM Laboratory, Telecommunications Services Group, DE-TCOM-GST, EPFL, CH-1015 Lausanne
◆ University of Delaware, Newark, DE-19716, USA

0-7803-3250-4/96$5.00©1996 IEEE

of the formal model to operating conditions (time kernel, scheduling). Nevertheless, this remains an abstract model that needs further refinement to reach an implementation.

Our methodology (Figure 1) is decomposed as follows:

❖*define* system requirements,
❖*formalize* functional requirements,
❖*enhance* formal specifications with relevant information,
❖*translate* an specifications into a performance modeling,
❖*implement* and *simulate* the performance modeling,
❖*evaluate* simulation results and *optimize* the system.

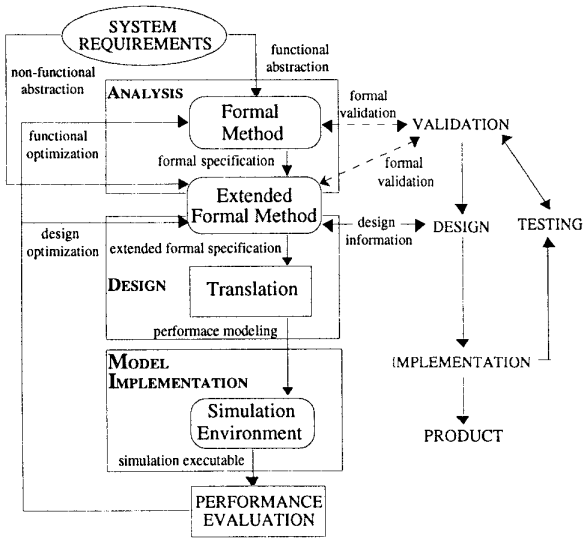

Figure 1: The proposed methodology.

The requirements specification defines the system interfaces (which services, which cost, what performance) and data (which messages, which workload) requirements.

The functional requirements formalization aims to structure the system separately of its implementation. It contains the information held in the system, the behavior which the system will adopt, and the details of the interfaces to the outside world. System logical properties can be verified, so one can see if we are building the model right according to the requirements. The abstract vision of the formal specification is suited because it provides a generic structuring that serves as basis for many different implementations, and because it avoids the complexity introduced when looking at details.

The extension of the formal specifications aims at refining and integrating relevant information for performance evaluation. It includes some design features such as processing, and a quantitative description of the interactions between the system and its environment.

Evaluating performance requires modeling the system, to get an early opinion of what the system will do. Using an extended formal specification has the advantage to integrate a functional description and selected design features, and thus, avoiding misleading complexity.

The performance implementation provides an executable code that mimics the behavior of the system. Executing it would provide statistical results that after analysis bring to an end optimization of the system. Simulations promote identifying bottlenecks and determining critical performance parts that require further design.

Applying our methodology to the SDL language gives an SDL specification that provides the structure (blocks, processes, communication links) and the functional description of the system. Then enhancing the formal model with performance information supplies an Extended SDL specification, that contains the SDL specification and adds the customer's quality of service requirements and workload, and the processing design. Translating the extended formal specification into a performance modeling requires replacing each SDL process by a performance process and determining which performance experiments to conduct. Then, the performance modeling is implemented on a performance evaluation tool (e.g., OPNET). Finally, the performance simulation is executed and its results are analyzed. When the system fails to fulfill a given quality of service, we should identify the system bottlenecks and optimize them.

3. Extending SDL Formal Specifications

The goal of a formal specification is to develop a model of what the system will do, including meaningful information from real-world perspective and presenting an external view of the system. It abstracts details in order to give an overview of the system, to postpone design decisions, and to allow all valid implementations. System properties can be verified through validation and testing:

❖*testing* checks that the external behaviour of a given implementation is equivalent to its formal specification.
❖*validation* checks that a formal specification is logically consistent.

3.1. The SDL Language

The basic idea of SDL [12] is to describe a system as communicating processes. A *process* is an extended finite state machine, that is either processing data or if no data are available for processing, is dormant in a state. The *state* defines what actions a process is allowed to take, which events it expects to happen, and how it will respond to those events. The processing (transition) performed depends solely on the state in which the machine was last dormant, on the data that become available and on the local conditions. Several items of data may become available during the machine processing, and a queue is associated with the machine to pile up the data. Communication is performed asynchronously, by way of connection paths. Blocks communicate by way of

channels, though processes can communicate with each other inside a block by way of signal-routes.

3.2. The Extended SDL

SDL is composed of abstract communication links and extended finite state machines, while a real-system is composed of physical components and networks. Thus, an SDL representation contains some differences with respect to the real systems. Some differences concern the nature of the components, and some the functioning. Thus, we need to define extensions to SDL that restrict the scope of these differences. We have defined our extension in a way that the SDL-92 standard is a subset of the *Extended SDL*.

Modeling time: A basic assumption in SDL is that the system is fast enough to process the offered load. In a real-system this assumption is not true since each signal transfer and each processing takes some time. To map the SDL model, the real-system should be fast enough to meet the load and response time requirements without destroying the validity of the SDL description. Thus, Extended SDL provides a transition concept that specifies processing duration by means of delay clauses. The extended execution model stipulates that all the actions before a delay clause are executed immediately, then the execution is suspended for the specified delay; when completed, the automaton resumes its execution.

SDL is equipped with timers and operates upon these. A timer stimulates a process as a function of a defined time by placing a timer signal in the input queue of the process. The timer execution model fails at verifying temporal *liveness* (something "good" will eventually happen within a given time). The reason is that as processes receive signals through a FIFO input queue, no assertion can be made on the duration it takes to consume earlier events. To prevent from it, we stamp Extended SDL timer signals with a priority. Thus, when the timer expires, the process receives the timer signal and starts its service once all earlier priority signals have completed. The time elapsed waiting for service is equal to the addition of the delay clauses of earlier priority signals.

Describing unreliability: SDL systems may suffer from specification errors, but the abstract representations they provide do not suffer from physical errors. SDL assumes that processes and communication links always operate according to their specifications. It is not assumed that processes will stop or that communication will distort the content of signals. But in the real world, errors manifest themselves as faults in operation of communication links and processes. Hardware errors, physical damage and noise are caused by physical phenomena entirely outside the realm of SDL. However, their effect is handled explicitly in SDL specifications. Besides, in SDL only the interfaces between a system and its environment are imposed by the considered problem; all internal structures are purely a means to express the behaviour. Thus, an unreliable communication link can be explicitly modeled by a channel substructure.

Modeling processing: SDL provides a model that abstracts from design details. Though, design decisions are important for performance evaluation. SDL provides a processing model consisting of a single server and a single infinite, FIFO queue. Even though this modeling captures the nature of many processes, it does not hold for all real processes. Extended SDL provides a more flexible model that allows specifying processing models consisting of a set of bounded queues and of servers.

A processing resource can be seen as customers arriving for service, waiting for service if it is not immediate and leaving the system after being served (Figure 2).

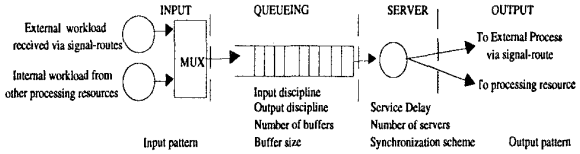

Figure 2: Processing resource.

❖ The *input* models the arrival of data to the resource.
❖ The *queueing* defines the number of distinct storage queues. Each queue is described by its *queueing discipline* (how customers are inserted and selected for service when the queue has formed) and its *buffering capacity* (a limited buffering capacity may arise with signal losses).
❖ The *server* defines the number of simultaneous service requests that the resource is able to perform. Each server is characterized by its service duration (time required to serve a request). Service duration is given within the process automata, by means of delay clauses within transitions.
❖ The *output* models the departure of data.

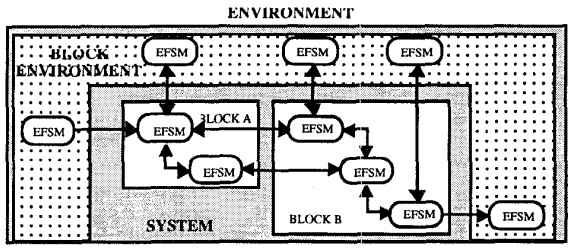

Figure 3: An Extended SDL specification.

Describing workload: The performance of a system depends on the workload it handles. SDL gives a workload qualitative description since each interface defines a list of valid signals. As we also need quantification, Extended SDL enhances the specification with a set of processes, one by environment interaction channel (Figure 3).

Describing quality of service: A customer sees the system as a black-box to which it applies requests and gets outputs. Each customer can quantify its satisfaction by means of some quality of service requirements. In performance evaluation, the QoS helps to monitor how these metrics evolve during the system computation, and helps to check if the system fits the required values. Thus, Extended SDL includes them in the specification.

4. Performance Evaluation

Performance evaluation facilitates comparing alternative designs and finding which is the best according to the chosen criteria. Even if there are no alternatives, it helps in determining how well the system would perform and which improvements need to be made. To get performance evaluation we must perform following steps (Figure 4):

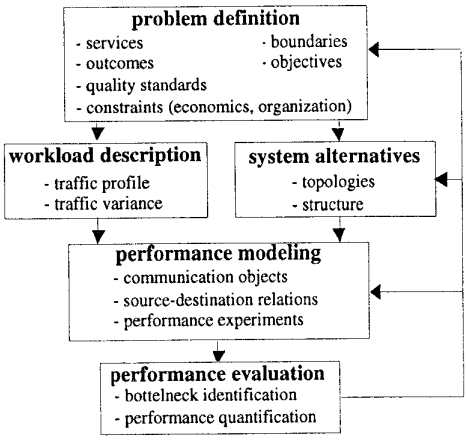

Figure 4: Performance evaluation process.

1. Define the problem.
2. Describe the system configuration.
3. Describe the system workload.
4. Create a consistent and correct performance model.
5. Analyze the performance model.

A Formal specification provides a good functional description and a high-level configuration of the system. Nevertheless, performance evaluation requires a more detailed specification. An extended formal specification provides, therefore, the system configuration with enough detail for performance evaluation. Besides, it provides the workload description. Therefore, formal world covers the steps 1 to 3 of the performance evaluation process.

The formal specification provides a good basis for starting performance modeling. Thus, we have defined a translation method (Figure 5) which principle is to map each formal EFSM into a combination of an EFSM and of a queueing network, where the EFSM models the behavior, while the queueing network describes the congestion of multiple requests to restricted resources.

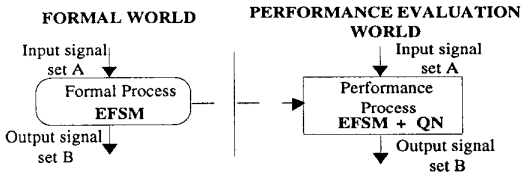

Figure 5: The process mapping technique.

Determining the performance of a system requires a careful selection of the parameters to evaluate. The extended formal specification provides us with such a selection through the quality of service specification.

The last step towards performance evaluation consists in executing the performance modeling, monitoring the quality of service parameters and determining if the observed values respect the specified limits. If these are not, we need to determine system bottlenecks and restart the evaluation process after a system optimization.

Executing the performance modeling requires a suitable performance evaluation environment. We chose to use a commercial tool, based on EFSMs, called OPtimized Network Engineering Tool (OPNET). Opnet is capable of developing and simulating communication systems with detailed behavior modeling and performance analysis. Opnet expresses processes as a combination of state transition machines, high level functions, and facilities of the C programming language.

5. A Case Study - TCP

Transmission control protocol (TCP) is a transport protocol. It provides a connection-oriented, reliable service. Two stations using TCP must establish a TCP connection before starting exchanging data. During the data exchange, TCP packetizes the data into segments, sets a time-out any time it sends data, acknowledges data received by the other end, reorders out-of-order data, discards duplicate data, provides end-to-end flow control, and calculates and verifies an end-to-end checksum. TCP is used by many popular applications: Telnet, FTP, SMTP.

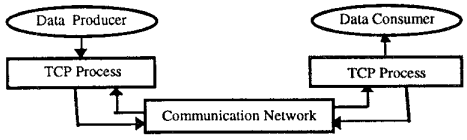

Figure 6: TCP illustration example.

In our example we focus on the exchange of data; we do neither consider the establishment nor the release of TCP connections. Figure 6 shows the outline of the system. "Data Producer" is a user generating data that should be reliably transmitted to another user: "Data Consumer". through an unreliable network. Figure 7 shows the SDL specification. Data enter the system by the channel "to_TCP" and leave by "from_TCP". Figure 8 shows the

extended specification in which we add the consumer's quality of service and workload description. Besides, we describe the processing of each process (structure, service), and how errors occur in the network.

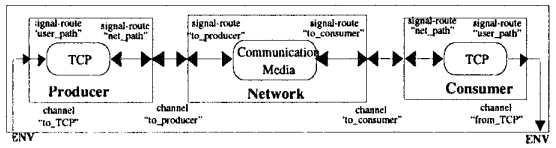

Figure 7: SDL specification of the TCP example.

The "Data Producer" gives the application performance expectations in terms of resource utilization, delay, losses and throughput. The workload description is used to generate data during simulations. The processing design and service are used to simulate how processes execute.

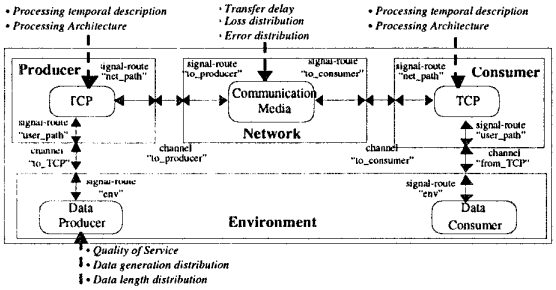

Figure 8: Extended SDL specification of the TCP example.

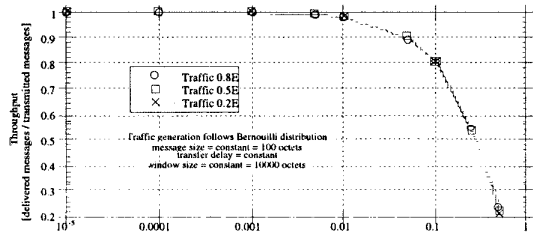

Figure 9: Throughput variation

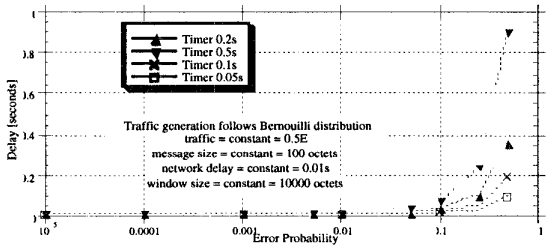

Figure 10: Delay variation

We have run simulations of the modeling of our TCP example. This modeling was implemented on the tool Opnet and obtained using our methodology. Simulation results (Figure 9, 10 and 11) can be used to verify under which conditions the required quality of service is fulfilled.

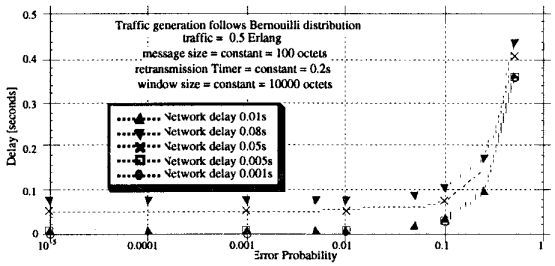

Figure 11: Delay variation as function of the network delay

5.1. Conclusion

We believe that modeling performance from formal descriptions is a powerful approach because it saves time and avoids errors. Furthermore, it combines the power of formal techniques (validation and conformance testing) with the power of performance evaluation.

Our contribution proposes a methodology reusing the formal specification, enhancing it with performance information and mapping it to a performance modeling. Analyzing the simulation results of the performance modeling allows optimizing the system. We are convinced that our methodology is simple, powerful and generic. Besides, we have no doubt that it can be automated.

References

[1] F.Bause, P.Buchholz, Protocol Analysis using a timed version of SDL, in IFIP Formal Description Tecniques 1991.

[2] G.Bochmann, J.Vaucher, *Adding performance aspects to specification languages*, in PSTV 1988.

[3] P.Dembinski; *Queueing Network Model for Estelle*; in Formal Description Techniques 1993.

[4] M.Diaz, *Modeling and Analysis of Communication and Cooperation Protocol using Petri Nets based Model*, in Protocol Specification, Test and Verification 1982.

[5] M.Hendaz, S.Budkowski, *An Enhanced Estelle Simulator for Performance Evaluation*, in Colloque Francophone sur l'Ingénierie des Protocoles 1995.

[6] E.Heck, D.Hogrefe, B.Muller-Clostermann; *Hierarchical Performance Evaluation Based on Formally Specified Protocols*; IEEE Transaction on Computers 1991.

[7] J.Quemada, A.Azcorra, D.Frutos, *A timed calculus for LOTOS*, , in IFIP Formal Description Techniques 1989.

[8] M.Sredniawa, B.Kakol, P.Gumulinski; *SDL in Performance Evaluation*; in SDL Forum 1987.

[9] C.Wohlin; *Performance Analysis of SDL systems from SDL Descriptions*, in SDL Forum 1991.

[10] S.Zhang, S.Chanson; *An Approach to Evaluating the Performance of Protocols based on Formal Specifications*; in International Conference on Network Protocols 1993.

[11] U.Herzog; Performance Evaluation and Formal Description; In IEEE Conference CompEuro 1991.

[12] UIT-T,; *CCITT Specification and Description Language*; Z.100, 1993.