14

The QUEST-Approach for the Performance Evaluation of SDL-Systems

Marc Diefenbruch, Jörg Hintelmann and Bruno Müller-Clostermann
University of Essen, Dept. of Mathematics and Computer Science
D - 45117 Essen, Germany
phone (+49) (+201) 183 3915, fax (+49) (+201) 183 2419
electronic mail: {md,jh,bmc}@informatik.uni-essen.de

Abstract

Designing complex systems requires formal methods supporting validation and performance evaluation before implementation. We present an approach based on the adjunction of queueing stations and traffic sources to SDL-systems yielding integrated models assessable by quantitative evaluation. As a consequence SDL-specifications may be used as a basis for performance evaluations of systems during the design phase. Apart from the underlying concept we introduce the language QSDL (Queuing SDL), and the QUEST tool that implement our approach. As an example we show how the performance of an SDL specified communication protocol may be investigated by employing QSDL and QUEST.

Keywords

Extensions of FDTs, SDL, performance modelling and analysis,

1 INTRODUCTION

In addition to the functional correctness of complex systems like communication protocols and embedded systems another essential item is their performance behaviour. Hence system developers often require quantitative measures like throughput and response time to decide on implementation design alternatives. To obtain such performance values during early design phases an executable model has to be constructed that reflects implementation-dependent information like the concurrent and time consuming usage of limited resources, the choice of data structures and algorithms, and properties of the target hardware.

Since a formal specification has to be as implementation-independent as possible specification languages do not (and should not) cover performance aspects. To obtain performance measures in addition to the functional behaviour implementation-related quantitative properties have to be specified. These may cover issues like performance characteristics of hardware devices,

R. Gotzhein et al. (eds.), *Formal Description Techniques IX*

degree of available parallelism, concurrency due to non-sharable resources, scheduling strategies, processing speeds, channels bandwidths, buffer sizes, timeout values, and last but not least workload and traffic characterization.

Model based performance evaluation is associated with stochastic modelling using queueing networks, Markov chain techniques, timed Petri nets and discrete event simulation. To liberate the user from the necessity of a detailed knowledge of modelling techniques a great number of tools is available to support the performance evaluations of computer and communication systems. As representatives we mention the timed Petri net tools TimeNet (German and Mitzlaff 1995) and QPN-Tool (Bause, Buchholz and Kemper 1995), the queueing network tool RESQ2 (Lavenberg 1983) and the full-fledged modelling environment HIT (Beilner, Mäter and Weißenberg 1988). Nevertheless, performance evaluation has been (and is still) isolated from the methodology of system design and development. Hence, the integration of performance evaluation techniques with the main stream of computer and communication system engineering is an important requirement (Brœk and Haugen 1993, Ferrari 1986).

A summary on early work on FDT-based performance modelling has been given in (Rudin 1983), SDL-related approaches have been discussed in (Bause and Buchholz, 1991, Bause et al. 1995, Böhmer and Klafka 1994, Heck et al. 1991), performance evaluation methods based on Estelle, LOTOS and process algebras have been proposed in (Bochmann and Vaucher 1988, Kritzinger and Wheeler 1990, Marsan et al. 1990, Hillston et al. 1994). An overview of performance enhanced FDTs with an emphasis on the parallel processing of communication protocols is given in (Walch 1994).

The approach presented here introduces new elements for the specification of non-functional properties in SDL-systems such that aspects like competition for limited resources and the time durations may be included. Based on a more general framework proposed in (Heck, Hogrefe and Müller-Clostermann 1991, Heck 1992) we have recently developed the language QSDL as an extension to SDL (Diefenbruch et al. 1995). Here we introduce the QUEST-approach comprising the language QSDL and the QUEST-tool. The central part of QUEST is the language QSDL (Queueing SDL) extending the Specification and Description Language SDL by resources that are designed to reflect queueing stations with a waiting room and servers. The main objective of QUEST is to bring performance evaluation techniques to the working place of protocol designers and software engineers.

The rest of the paper is organized as follows. We start with an overview of the concepts, followed by an introduction to the QSDL-language and a brief description of the QUEST-tool itself. Finally its usability is demonstrated by a well-known standard protocol from Andrew Tanenbaum's textbook on computer networks (Tanenbaum 1996).

2 THE QUEST-CONCEPT

The starting point is the formal description of a systems behaviour in SDL, the Specification and Description Language of the ITU. Such a specification is usually called an *SDL-system.*
An SDL-system may serve as basis for validation, functional simulation and animation, code generation, prototyping, testing, and more. Nowadays there are SDL-tools like SDT, ObjectGeode and Melba supporting all these activities. The aspect of performance evaluation, howe-

ver, has still not been integrated in the SDL method and the SDL-tools (Brœk and Sarma 1995, SDT 1996). The closing of this gap is exactly the objective of the QUEST-approach.

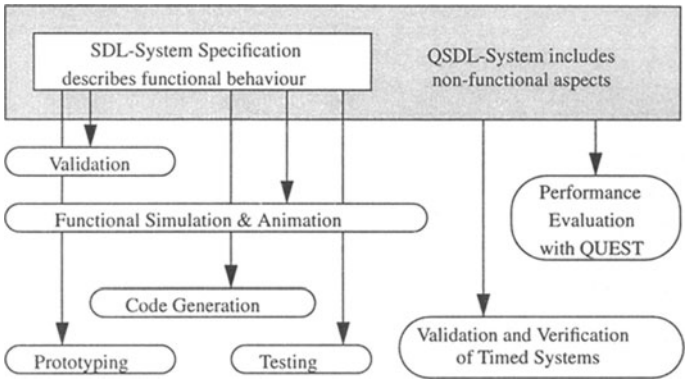

Figure 1 The Role of SDL- and QSDL-Systems in the Design Process.

Figure 1 shows the central role of an SDL-system specification in the development process. The so called non-functional properties covering time and resource aspects may be included in the extended SDL-specification called QSDL-system that is to serve for the automatic construction of a performance model. Note that we included in the framework given by figure 1 also the topic of timed validation and verification that is strongly related with the design of real time and reactive systems.

2.1 Overview of the QUEST approach

The QUEST approach is based on the adjunction of time consuming *machines* that model the congestion of processes for limited resources. By adding workload models and after defining a mapping of workload to machines finally an assessable performance model is automatically generated. The description and construction of performance models and their evaluation is supported by the language QSDL (Queueing SDL) and the tool QUEST.

We consider an introductory example. An SDL-specification including a two layered protocol and a medium form the core of the model. Attached to this core are specifications of the workload (traffic source) and the specifications of resources (machines) that model delays due to waiting, service and transmission. Signals sent by the traffic sources yield actions in the protocol instances that finally lead to service requests at the machines. The mapping of requests to machines is also displayed in figure 2.

The figure shows three typical usages of machines:

- Modelling of parallelism: The instances B1 and B2 have exclusive access to machines M1 and M2 respectively.

- Modelling of the competition for ressources: The instances B3 and B4 are sharing the machine M3.
- Modelling of time durations for the physical transmission by a "medium", here modelled by machine M4.

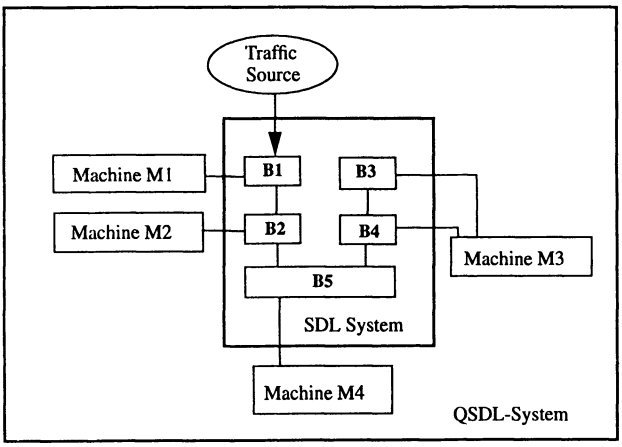

Figure 2 On the Role of Machines in a QSDL-System.

The language QSDL provides constructs for the description of the extensions sketched above. For the purpose of load modelling and for the resolution of non-determinism stochastic distribution functions are available; also the modelling of multiple state sources that describe complex traffic patterns is possible in a straightforward way. Machines are building blocks providing a waiting room, a number of servers, a scheduling strategy, and in particular a set of *services*. The processes may request the services provided by the machines. The mapping of requests to machine services is done with the help of a binding mechanism that resembles the signal transport mechanism of SDL.

The whole complex, as shown in figure 2 as QSDL-system includes all the information that is necessary to build an executable and quantitatively assessable model. The construction of this performance model in form of a simulation program is done automatically by the QUEST-tool. The execution of the simulation model finally yields performance values that may be visualized graphically or summarized as a report.

In order to construct a performance model we have to supplement the SDL-specification with definitions concerning the

- machines with their provided services,
- quantitative amount of requested service and priorities of the service requests,

- the characteristics of traffic sources,
- binding (interconnection) between the load and the machine.

2.2 Machines and machine services

Queueing models are a popular paradigm in performance modelling. They are used to describe and analyse the congestion of multiple requests to restricted resources. A system described with the help of queueing stations may be investigated by analytical, numerical or simulative techniques (Lavenberg 1993, Kant 1992).

In QUEST queueing stations are used as machines for the modelling of limited resources. Typically the so called Kendall notation like e.g. A/B/C/K/Z is applied to denote the parameters of a queueing station.

- The interarrival and service time distributions are given by A and B respectively.
- C denotes the number of servers.
- K is the maximum number of requests admitted at the station.
- Z is the scheduling discipline.

Depending on the type of parameters closed analytical formulas for utilization, wait and response time are derivable. Famous examples for analytically tractable models are the M/M/1-, M/M/1/K and M/M/m/m-systems, as known from the Erlang loss formulas, that have been successfully applied in designing telephone networks. There are other kinds of queueing stations like processor sharing stations, multi-queue/multi-server-stations and stations with priority scheduling disciplines. A simple example for a queueing station is a single server with a FIFO-queue, cf. figure 3 a).

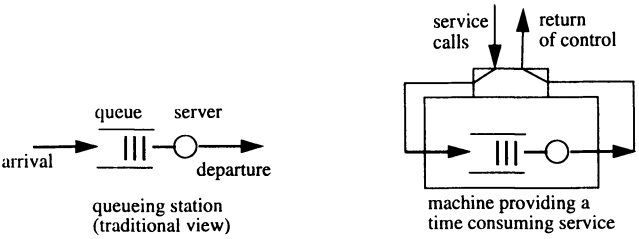

Figure 3 a) Queueing Station. b) Machine providing a Service.

The QUEST view of a queueing station is a *machine* providing a service to its environment, cf. figure 3 b). The provided service may be *requested* or *called* by entities from the environment via an interface. This viewpoint is consistent with modern design and specification methodologies and has been adapted successfully to the field of performance modelling some time ago (Beilner, Mäter and Weißenberg 1988, Heck 1992). The total time a request spends in a queueing station depends on the amount of required service, the speed of the server and additionally the wait time suffered in the queue. The amount of requested service is mostly described by a random variable, whereas the speed of the server is a real positive constant. The wait

time in the queue depends on the congestion due to concurrent usage of the machine. After service completion the control returns to the requesting process.

Machines have the following properties.

- A machine has a *name*.
- A machine offers *machine services* that are accessible by the processes. Each machine service has a unique name.
- Each machine service has real positive *speed*. The time progress of a service is determined by the specified amount of the service request and the service-specific speed of the executing machine.
- A machine has m identical servers, $m > 0$. The number of servers describe the degree of parallelism on that machine. The standard case, $m=1$, means strictly sequential processing. The case of "infinite servers" (service discipline IS) stands for maximum parallelism.
- A machine models the time delay that starts with a service request (call of machine service) and ends with the return of control to the caller. The suffered delay due to this service is the sum of wait time and service time.
- A machine has a scheduling strategy (called service discipline) that determines the order of service. The scrvice disciplines have unique names with a specific (intuitive) semantic. The service disciplines available in QUEST include FCFS (First-Come-First-Served or First-In-First-Out), Random- and Priority disciplines.
- A machine does not model any functional properties. Its internal structure is limited to provide a service specific and population dependent time delay.
- Machines can model limited resources *and* time consumption.
- The connection between requests and machines is established by a particular binding-mechanism.

Note that the arrival and service demand characteristics are a part of the workload description briefly sketched below.

2.3 Traffic and load modelling

Another important topic that highly affects system performance is the behaviour of the *traffic sources*. Hence the characterization of the workload using suitable traffic parameters is part of each quantitatively assessable model. QUEST supports the workload modelling by various features, in particular a number of random distribution functions are available. The traffic characteristics of current and future telecommunication systems may be described by load generators that may be implemented as QSDL processes. The signals are generated by these processes according to stochastic distributions for the signal interarrival times and finally lead to requests at the machines. An important option is the usage of multiple state sources, like Markov Modulated Poisson Processes (MMPP) or Markov Modulated Bernoulli Processes (MMBP) leading to a very flexible definition of the different traffic characteristics of file transfer, voice and video transmissions or even multimedia applications (Onvural 1992).

A typical example for a multiple-state source is an on/off-source, also called Interrupted Poisson Process (IPP). In the on-state the source will generate traffic (packets, cells) with a given rate. In the off-state the source is silent. Each event generated by the source may be interpreted as service primitive of the protocol system under consideration. Hence a signal will trigger actions in the SDL-system leading to service requests at the machines. In this way both congestion and time consumption may be modelled.

As displayed in figure 2 traffic sources may be attached to the processes and blocks of an SDL-system. In particular the attachment of traffic sources directly to the medium or to instances of lower layers may model the occurence of so called cross traffic that reduces the available bandwidth.

3 THE LANGUAGE QSDL AND THE QUEST-TOOL

After a first overview of the graphical representation of QSDL, named QSDL/GR, we introduce the syntax of QSDL and sketch briefly the QSDL-tool.

3.1 Overview of QSDL

Here we introduce the extension of SDL by syntactical constructs for *machines* and their connection to the processes and blocks of an SDL-system. The resulting specification and performance analysis language QSDL provides means for the specification of load, machines and their binding. These extensions strive for maximum structural and syntactical consistency with SDL.

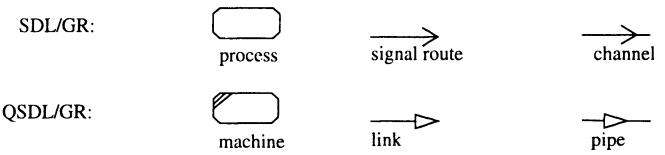

Figure 4 Elements of SDL/GR and QSDL/GR.

The binding between load and machines is managed by a *service call* transport mechanism, which is totally independent of the signal transport mechanism of SDL. *Load* occurs in the QSDL-processes by the instantiation of time consuming *requests* that are referred for execution to adjunct machines given by queueing stations. The graphical representation of the new constructs (QSDL/GR) is displayed in figure 4. An example is given in figure 5.

As shown in figure 5 QSDL-processes are bound to the machines via links and pipes. Processes and machines within the same block are only connected with a link. The processes serve as starting point and the machine as endpoint of the link. In the case that the process and the derived machine are in different blocks the binding has to be carried out in two steps .First of all it's necessary to define a pipe between these blocks and links from the process to the border of the block and from the border of the block to the machine. In the second step the links and

pipes are interconnected. This will be done in the same way as the connection of SDL-channels and signalroutes with the difference of using *„bind to"* instead of *„connect and"*.

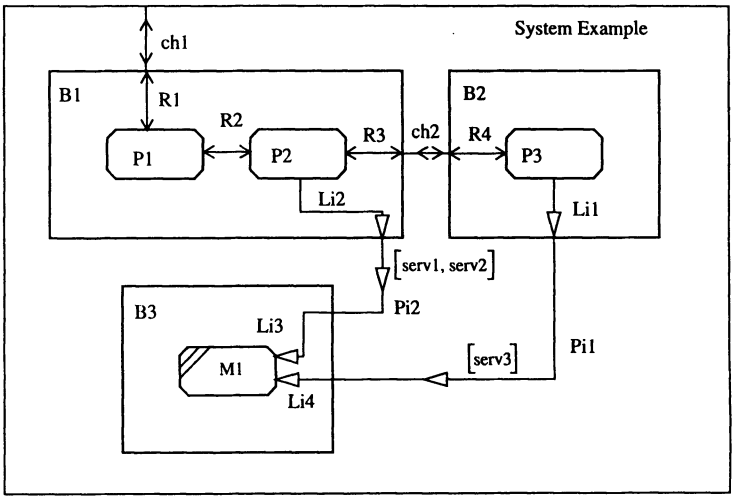

Figure 5 A System described with QSDL/GR.

3.2 QSDL-syntax for modelling load and machines

In QSDL we associate temporal durations (deterministic and probabilistic) and the use of resources with certain actions. The first step concerns the declaration of all service types used in the machine specifications. QSDL-service types are declared in the same style as SDL-signals. The keyword *machineservice* marks the beginning of the declaration followed by an enumeration of the declared services e.g.:

```
machineservice serv1, serv2, ..., servn;
```

Example 1 Declaration of Machine Services

The keyword *request* followed by at least one *machineservice* name instantiates a time consuming request. The requests have to be routed to the service providing machines. If the destination machine is unambiguously defined no further information is necessary. Otherwise the desired machine has to be identified by adding the path information after the keyword VIA. Every request instruction requires the service amount which can be a raw value specifying the cost of request execution. Examples for amounts are the number of instructions to be executed or the number of packets to be sent. The actual service amount is obtained by evaluating a real expression, in particular it may either be constant or random. The latter case requires a variable, whose actual value can be set by calling one of the random functions provided by appropriate predefined datatypes. A second (optional) parameter can supplement a request-

instruction to specify the priority of a request. A priority is a cardinal number and has the default value 0 (lowest priority).

We now turn to the specification of machines in QSDL. A QSDL-machine is constructed like a queueing station and has the parameters: *name*, the number of servers, *service discipline*, set of *services* offered and the service-specific *speed* values. Syntactically a QSDL-machine is specified as follows:

```
MACHINE example;
    SERVER  1;
    DISCIPLINE  FCFS;
    OFFERS      serv1: 3.0, serv2: 5.0;
```

Example 2 Machine Declaration

The *discipline*-expression specifies the service discipline of the machine, e.g. `FCFS`, `RANDOM` or priority. The `offers`-command specifies the services offered by a machine. Each service is specified by its name and a speed parameter S that quantifies the amount of work a machine can execute per time unit. The service time D of a request on a QSDL-machine is given by $D = A/S$, where A is the service amount.

After the formal specification of load and machines, the load has to be referred unambiguously to the machines. The *binding* of requests to machines preserves the SDL-hierarchy. The following example declares a bidirectional pipe between the blocks `B1` and `B2`, which can forward the machineservice types `serv1` and `serv2` from `B1` to `B2` and `serv3` vice versa, cf figure 5.

```
pipe Pi2
      from B1 to B2 with serv1, serv2;
      from B2 to B1 with serv3;
```

Example 3 Declaration of Pipe Pi2

Connecting processes in blocks *B1*, *B2* and *B3* to the block boundaries is done with *links*, in figure 5 named as *Li1*, ..., *Li4*.

3.3 On the semantic of time in QSDL

The exact semantic of time in pure SDL is a non-trivial topic, cf. (Mork et al. 1996) or the remarks in (Bütow et al 1996). SDL users are often involved in discussions concerning the semantic of timers, the role of the external clock "tick", non-determinism and similar aspects. When moving from SDL to performance enhanced versions of SDL the degree of abstractness decreases and the "formal" description is enriched by non-functional parts and hence develops towards a specification that reflects both the intented implementation and the available resources. In case of QSDL we have added "machines" that lead to the blocking of processes submitting calls to these machines. Blocked processes have to wait for a time duration given

by the sum of wait and service time suffered by a call at a machine, i.e. we have the "call semantic" known from programming langugages. A presentation of a (more) formal semantic of QSDL is beyond the scope of this paper.

3.4 Overview of the QUEST-tool

The transformation of the QSDL-description to an executable simulation program is performed automatically by a tool system that has been developed at the University of Essen. The tool system includes a QSDL-parser, a simulation class named SCL, and a compiler for translation of QSDL-specifications to a simulator in the programming language C++. A survey of the SDL-features supported by the QSDL-parser of QUEST is given in Appendix A.

Furthermore there are components for the evaluation and visualization of experiments by gathering statistical data as well as by *message sequence charts* (MSC) that are extended by timing informations. Future versions of QUEST will additionally provide techniques for timed validation and verification as indicated in figure 1.

4 USING QUEST FOR PROTOCOL EVALUATION

The application of the QUEST method is shown in a case study following a well known example. A sliding window protocol using selective repeat flow control and error recovery has been completely specified in SDL according to the semi-formal description given in Andrew Tanenbaum's textbook on computer networks, cf. the section on sliding window protocols in (Tanenbaum 1996). The protocol's behaviour is summarized below, see also figure 6 and figure 7.

Users may ask for a reliable unconfirmed service that transmits data packets in sequential order between directly connected equipments. The service primitives at the service access points are *UReq* to demand for data transmission and *UInd* to indicate correctly transmitted data. The service is provided by two protocol instances both using an unreliable service that may lose or corrupt data packets. Each protocol instance has an input buffer for incoming packets and an output buffer for outgoing packets respectively. The instances set a timer for each outgoing packet. If a packet gets lost or the acknowledge is too late the timer will expire causing a retransmission of that packet.

Correctly received packets will be acknowledged with the next outgoing packet by piggybacking or separately by use of acknowledge packets if no data for transmission in the other direction is available. The instances will forward the data to the layer above if they have been received in their sequential order. Each instance has a limit W (window size) for forwarded, not yet acknowledged packets. If the limit W is reached no more data will be accepted until new acknowledgements have been arrived. The complete QSDL system including the formal specification in SDL is shown at block level in figure 6.

In figure 6 also two machines (*m1* and *m2*) serving the protocol stacks are displayed. To keep the model simple these machines are not considered in the evaluation. A machine, called *m3*, modelling the time delay and the contention on the transmission line is attached to the medium

block. This machine offers the service *deliver* which is requested by the processes *m_inst1* and *m_inst2* using the statement *REQUEST deliver(paklength),* see figure 7. We assume a medium bandwidth of 1 Mbit/s resulting in a value for machine speed of 1048576 bps. The amount of requests is chosen as the frame length and set to 8192 bit (= 8Kbit). The frame length may also be specified randomly distributed according to the predefined distribution types.

The SDL code expresses that losses and errors during transmission may happen but it does not quantify their occurence. The frequency of loss events is expressed by random variables. The variable *loss* is of type *Bernoulli*. The procedure *sample* which is defined for all QSDL distribution types draws the value *true* with a given probability p and *false* with $1 - p$. The value of p may be chosen as the expected loss probability of a frame in transmission and has been set in the study to a value of 10^{-3}.

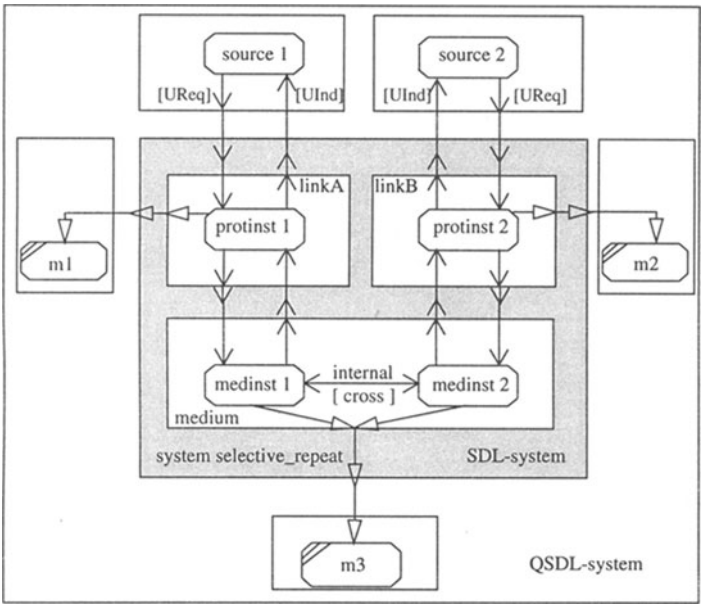

Figure 6 QSDL-System of a Communication Protocol (Block Level).

Bursty traffic sources are attached to the environment of the SDL-system, see figure 2 or figure 6. Defining desired measures like waiting periods at machines or in SDL input queues, state frequencies or queue length distributions completes the model description. It is also possible to define *actions* by a start event and a corresponding end event, e.g. a service request and the corresponding service indication. The time between these events may then be computed and evaluated as a user defined performance measure. All supplements concerning the experiment description are automatically included into a simulaton model.

SDL - code

```
DCL pak Frame;

STATE ready;
INPUT Mesg(pak);        /* SDU of medium */

DECISION any;           /* nondeterminism */
   () : DECISION any;
          () : TASK pak!crcf := FALSE;
          () : TASK pak!crcf := TRUE;
        ENDDECISION;
        OUTPUT cross(pak) VIA internal;
        NEXTSTATE ready;
   () : NEXTSTATE ready;
ENDDECISION;
```

QSDL - code

```
DCL pak Frame,
    err BERNOULLI,
    loss BERNOULLI;

STATE ready;
INPUT Mesg(pak);
REQUEST deliver(paklength);     /* delay */
DECISION SAMPLE(loss);
  (FALSE) : DECISION SAMPLE(err);
              (FALSE) : TASK pak!crcf := FALSE;
              (TRUE) : TASK pak!crcf := TRUE;
            ENDDECISION;
            OUTPUT cross(pak) VIA internal;
            NEXTSTATE ready;
  (TRUE) :  NEXTSTATE ready;
ENDDECISION;
```

Figure 7 Extract from the Process medinst1.

The timed behaviour is visualized during model execution, see figure 8 for transient behaviour, and summarized performance values may also be obtained, see figure 9. Of course, a large variety of design alternatives and parameter settings may be studied. We give just a sketchy overview on few experiments.

4.1 Visualization of transient measures

Performance related parameters express quantitative requirements of service users. Often short term "transient" values may be more interesting than average values over long observation periods. Furthermore, performance parameters should be observable to allow a service provider to react, if guaranteed parameters can not be satisfied, or if a service user does not keep to the negotiated traffic parameters. The QUEST approach provides the observation of such short term values, see figure 8. The upper windows show the number of signals per time, which were generated from *source 1* and *source 2* respectively. Figure 8 displays two bursts of source 1 and three bursts of source 2 occuring during the observation period. The middle windows show that the bursts may lead to longer waiting periods at the associated instances.

Periods of higher medium utilization are another effect caused by the bursts, see the lower left window of figure 8. Apart from the functional behaviour including time and resource constraints the QUEST-tool also allows to observe the delay due to waiting periods or the buffer sizes needed. Moreover it is observable how long it will take to deliver all signals which have been generated during such a burst.

A control panel allows for stopping and resuming the simulation, the variation of the level of detail of the displayed measures, the saving of snapshot information and the resetting of performance estimators.

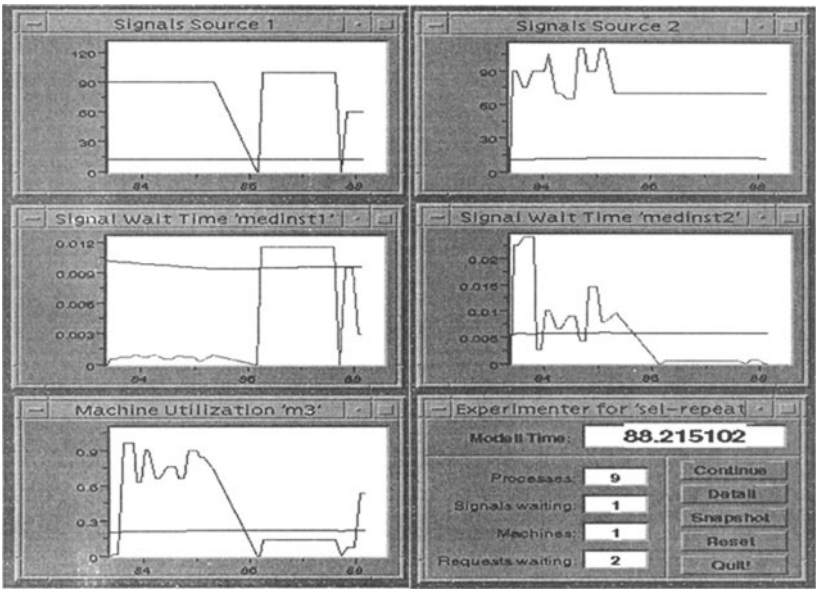

Figure 8 Short Time Behaviour of Traffic Sources and Performance Measures.
(The diagrams display the actual value as well as the long term average.

4.2 Steady state analysis of performance measures

Addditional to the transient behaviour the investigation of the long time behaviour of a protocol system may be helpful to adjust or even optimize configuration parameters. This requires the evaluation of steady state performance metrics.

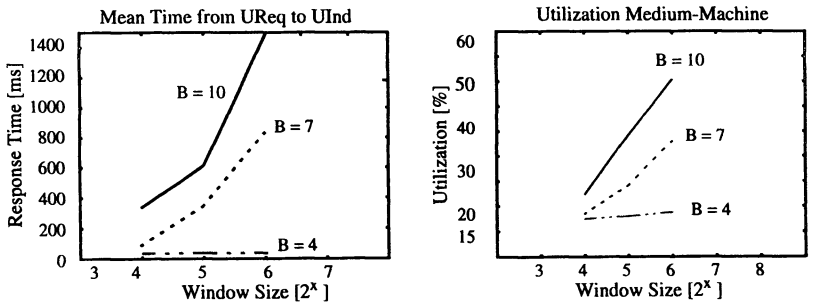

Figure 9 Steady State Measures for the Protocol.
(mean arrival rate 12 packets/s; bandwidth 1 Mbit/s)

The results of experiment series performed with the QUEST system sketched above are presented graphically, see figure 9. Response time and utilization for various settings of window size W and burst factor B of on/off sources have been investigated as an example to show the protocol's long time behaviour.Note that there is a rich literature on methods and techniques for the statistical analysis of simulation data that can be employed for more detailed investigations of protocol behaviour.

5 FUTURE WORK

The current version of QUEST covers SDL'88 and part of SDL'92. Some new features of SDL'92 like priority input, value returning procedures, spontaneous transitions or different continuous signals with identical priority are already supported by QUEST.

Until now the QUEST-approach has been used for relatively small examples or for specific protocol mechanisms (Hintelmann 1996). In a next step real world problems e.g. in multimedia applications or high speed networks are under consideration. Additional to the further development of methods and tools these analyses belong to the objectives of an ongoing research project named QUAFOS (Quantitative Analysis of Formally Specified Systems) which is supported by the Deutsche Forschungsgemeinschaft (DFG) during the years 1996 - 1998.

Demo-examples of executable QUEST-Simulators (for SunSparc) and QUEST-related documents may be downloaded via `ftp.informatik.uni-essen.de/pub/Quest`, or via `http://www.informatik.uni-essen.de/Sysmod/sysmod-engl.html`.

6 ACKNOWLEDGEMENTS

The various contributions of Axel Hirche, Christian Rodemeyer, Jörg Rühl, Wolfgang Textor and Reinhard Westerfeld to the implementation of the QUEST tool are highly appreciated. We also thank the unknown referees for their helpful hints.

7 REFERENCES

Bause, F. and Buchholz, P. (1991) Protocol Analysis Using a Timed Version of SDL. *3rd Int. Conf. on Formal Description Techniques (FORTE'90)*, Madrid (Spain), North-Holland.

Bause, F.; Buchholz, P. and Kemper P. (1995) QPN-Tool for the Specification and Analysis of Hierarchically Combined Queueing Petri Nets. in Bause, F. and Beilner, H. (1995).

Bause, F.; Kabutz, H.; Kemper, P. and Kritzinger, P. (1995) SDL and Petri net performance analysis of communicating systems, *15th International Symposium on Protocol Specification, Testing and Verification.*

Bause, F. and Beilner, H. (eds.) (1995). Quantitative Evaluation of Computing and Communication Systems, *Proc. of Performance Tools '95 and 8th GI/NTG Conference MMB '95*, Lecture Notes in Computer Science, Springer.

Beilner, H.; Mäter, J. and Weißenberg, N. (1988). Towards a Performance Modelling Environment: News on HIT. *4th International Conference on Modelling Techniques and Tools for Computer Performance Evaluation*, Palma de Mallorca.

Belina, F.; Hogrefe, D. and Sarma, A.(1990). *SDL with Applications from Protocol Specification*. Prentice-Hall.

Böhmer, S. and Klafka, R. (1994). A new Approach to Performance Evaluation of Formally Specified Protocols. *7th International Conference on Formal Description Techniques (FORTE'94)*, Berne, Switzerland, pp. 442-444.

Bochmann, G. v. and Vaucher, J. (1988). Adding Performance Aspects to Specification Languages. Proc. *8th Workshop on Protocol Specification, Testing and Verification*, p. 19-31, Atlantic City. North-Holland.

Brœk, R. and Haugen, Ø. (1993). *Engineering Real Time Systems*. Prentice Hall.

Brœk, R. and Sarma, A. (eds.) (1995). *SDL'95 with MSC in CASE. Proc. of the 7th SDL Forum*, Oslo, Norway, Elsevier Publishers.

Bütow, M.; Mestern, M.; Schapiro, C. and Kritzinger P.S.(1996). Performance Modelling from Formal Specifications. This volume.

Z.100 (1993), CCITT Specification and Description Language (SDL), ITU-T, Geneva.

Diefenbruch, M.; Heck, E.; Hintelmann, J. and Müller-Clostermann, B. (1995). Performance Evaluation of SDL Systems Adjunct by Queueing Models. In (Brœk and Sarma 1995), *Proc. of 7. SDL Forum,* Oslo, Sept. 1995, Elsevier.

Ferrari, D. (1986). Considerations on the Insularity of Performance Evaluation: *IEEE Trans. on Softw. Eng.*, Vol SE-12, No. 6.

German, R. and Mitzlaff, J. (1995). Transient Analysis of Deterministic and Stochastic Petri Nets with TimeNET. In (Bause, F. and Beilner, H. 1995).

Heck, E. (1992). On a Combination of SDL and HIT, in: K. Kronlöf (ed.), *Method Integration: Concepts and Case Study*. Wiley and Sons.

Heck, E.; Hogrefe, D. and Müller-Clostermann, B. (1991). Hierarchical Performance Evaluation Based on Formally Specified Communication Protocols. *IEEE Trans. on Computers, Vol. 40, No. 4.*

Hillston, J.; Hermanns, H.; Herzog, U.; Mertsiotakis, V. and Rettelbach, M. (1994). Stochastic Process Algebras: Integrating Qualitative and Quantitative Modelling. In Hogrefe, D. and Leue S. (1994) *7th International Conference on Formal Description Techniques (FORTE'94)*, Berne, Switzerland, pp. 439-441.

Hintelmann, J. (1996). Integration of SDL-Based QoS-Evaluation in Protocol Design. In Javor, A. and Lehmann, A. (eds.). *Modelling and Simulation. Proc. of the European Simulation Multiconference ESM'96 (Budapest), Workshop on Analytical and Numerical Modelling Techniques with Emphasis on Quality of Service Modelling*. 894-898, SCS International.

Kant, K. (1992). *Introduction to Computer System Performance Evaluation*. McGraw Hill.

Kritzinger, P. and Wheeler, G. (1990). A Protocol Engineering Workstation, *2nd Int.Conf. on Formal Description Techniques (FORTE'89)*. North-Holland.

Lavenberg, S. (1983). *Computer Performance Modelling Handbook*. Academic Press.

Marsan, M.; Bianco, A.; Ciminiera, L.; Sisto, R. and Valenzano, A. (1994). A LOTOS Extension for the Performance Analysis of Distributed Systems. *IEEE/ACM Trans. on Networking, Vol. 2, No. 2.*

Mork, S.; Godskesen, J.C., Hansen, M.R. and Sharp, R. (1996). A Timed Semantics for SDL. This volume.

Onvural, R. (1992). *ATM Networks - Performance Related Issues.* Artech House.

Rudin, H. (1983). From Formal Protocol Specification Towards Automated Performance Prediction, Proc. *Third Workshop on Protocol Specification, Testing and Verification*, Rüschlikon, Switzerland, May 1983. North-Holland.

SDT 3.0 Reference Manual & User's Guide. TeleLogic Malmö AB, 1996.

Tanenbaum, A. (1996). *Computer Networks.* Prentice Hall.

URL of the Authors: http://www.informatik.uni-essen.de/Sysmod/sysmod-engl.html

Walch, M. (1994). *Evaluating Parallel Processing of Communication Protocols.* Oldenbourg.

Appendix A: SDL-features supported by QUEST

The QSDL parser translates a textual SDL or QSDL specification into the internal format of the QUEST toolset, named QUEST depository. The depository serves as a basis for the code generation process.The parser of the current QUEST-version supports most of the SDL'88 standard and also some features of SDL'92. The table below gives an overview of SDL-features supported by QUEST.

supported	not supported
System	Services
Blocks	Packages
Processes	Block Partitionings (Substructures)
Procedures	Channel Partitionings
Channels	Viewed/revealed Variables
Signalroutes	Imported/Exported Values
Connections (of Channels and Signalroutes)	Macros
Signals and Timer	System Types, Block Types, Process Types
Variables	Generator Definitions
Actions (Task, Create, Call, Output, Decision, Join, Stop, Return)	Remote Procedures and Remote Procedure Calls
Input, Save, Pritority Inputs	
Continuous Signals	
Enabling Conditions	
Spontaneous Transitions	
Nondeterministic Decisions	
Asterisk Save, Asterisk Input, Asterisk State	
SDL-Datatypes (Syntypes, Synonyms, External Synonyms, Structure Sorts)	
Predefined Generators (Array, Powerset, String)	