Extension of SDL and MSC to Support Performance Engineering: A Discussion of Design Issues [*]

Andreas Mitschele-Thiel
Universität Erlangen-Nürnberg
Email: `mitsch#informatik.uni-erlangen.de`

Bruno Müller-Clostermann
Universität GH Essen
Email: `bmc#informatik.uni-essen.de`

Abstract

Formal description techniques allow to formally reason about the functional aspects of systems under development. This allows to detect and remedy functional errors in early stages of the development cycle. In order to also support performance engineering activities in the early development stages, a study of the integration of performance aspects into the standardized formal description techniques SDL and MSC has been launched within the ITU-T study group 10 in 1997. The integration of performance aspects into the standards is important to promote the wide-spread use of performance tools. The paper reports on the results of the study reached so far and discusses the issues involved with the integration of performance aspects into SDL and MSC.

1 Introduction

Performance issues play a major role in the system engineering process. Nevertheless, the integration of performance aspects in the system engineering process has not been well studied. Instead, systems engineering and performance evaluation are often being considered as two rather independent areas. Domenico Ferrari once coined the term 'insularity of performance evaluation' [Fer86], even the phrase 'esoteric cult' is sometimes used in this context. As a result of this, each of the two worlds has its own specialists and uses its own models, methods and tools. Especially annoying is that different models are used to deal with different aspects of the system under development. The major drawback of the use of disjunct models is the extra effort needed

[*] **Copyright Notice:** The Copyright of this material is retained by the authors and it may not be copied without the explicit permission of the authors. The material can be used within ITU SG 10 for discussion purposes, but may not be distributed to other parties without the authors' permission.

to keep the models consistent. The extra effort necessary to derive and maintain a separate performance model in addition to the functional model is often shunned by putting off performance issues as long as possible in the engineering process. A survey of the risks of the 'fix-it-later' approach is given in [Smi90].

The 'fix-it-later' approach is especially dangerous in the major application area of SDL. In the telecommunication sector, product families are offered which evolve over many years and which have to be maintained and updated for a period of 20 years or more. With such systems, the 'fix-it-later' approach with respect to performance often results in a destruction of the system architecture. In order to quickly meet performance requirements of the system at a stage where the system is already fully implemented and integrated, solutions are selected that are not in accordance with the system architecture (kernel bypasses, etc.). If this approach is taken for several iterations of a product line, complexity is gradually added to the system. This typically results in enormous cost and delays for system integration and testing. In the worst case, the system runs completely out of control due to its enormous complexity.

A system specified with SDL may serve as a basis for verification, validation, functional simulation and animation, code generation, prototyping, and testing. Today there are SDL tools like SDT [Tel96] and ObjectGEODE [Ver96] supporting these activities. However, the integration of performance and time aspects into the SDL methodology and the SDL tools has not been fully established so far.

Therefore, the usability of SDL for performance-critical applications is limited. System developers often require quantitative measures like throughput and response time to decide on design alternatives, on target system architectures, and later on for the optimization of parameters like timer settings, or window and buffer sizes. 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 performance properties of the target systems.

Currently, SDL and MSC do not deal with these important aspects of the system. Especially, they do not allow for the specification of physical aspects of the system, including limited resources as queues, processing time and communication times. As a consequence of this, the current methodology does not support three important activities of the system development process,

- (model-based) performance evaluation,
- joint verification of functional and non-functional aspects, and
- support for the implementation of Quality-of-Service (QoS) requirements of systems.

In order to support dynamic QoS requirements, the dynamic reaction of the SDL system to performance figures has to be supported, i.e. to react to feedback from the underlying resources by dynamically changing the functional behavior of the SDL system. Another related issue is the automatic derivation of efficient and responsive systems (see [Hen97, Bra93] for a discussion).

In the paper, we concentrate on the issues involved with the integration of information needed for a performance evaluation. In order to allow readers not familiar with SDL and MSC to profit from the discussion, we focus on the general issues involved and refrain from going into details of specific language constructs. The paper is a result of the discussion within the SDL and MSC community to support and integrate performance and time aspects. In order to foster the integration of performance and time aspects into SDL and MSC, a workshop on the topic has been held in conjunction with the ITU Q.6/10 (SDL) expert meeting in Erlangen, Germany, on February 17-19, 1998 [Mit98]. The paper summarizes the major issues involved in the integration of performance and time aspects in the formal description techniques SDL and MSC and the results of the discussion so far.

The important tools that integrate performance aspects into SDL are SPECS [Bue96], QUEST and the language QSDL [Die95, Que98], EasySim-II [Ger97], DNAsty and SDLnet [Kab97], and SPEET [Ste97]. Tools that base their performance evaluation on MSC rather than on SDL are the DO-IT Toolbox [Mit96], the rapid prototyping project conducted at the University of Erlangen [Mit97a], the SPEED tool [Smi97], and the tool described in [ElS98]. A survey and classification of the approaches employed by the tools can be found in [Mit97]. The different approaches to describe and associate the additional information needed for a performance evaluation to the SDL and MSC specification are described in [Mit98]. To our knowledge, SDL and MSC are the first formal description techniques for which the tight integration with performance aspects is pursued in such a concrete manner. In addition, we are not aware of any other design method that supports performance engineering in a similar way.

2 Language Issues

An important prerequisite for the integrated support of performance-related activities in the software engineering process is the identification and formal specification (implicit or explicit) of the aspects relevant to performance. Note that performance aspects also comprise some functional aspects (e.g. functional dependences within the application or the scheduling strategy of resources).

In order to support a performance evaluation, the analysis or design model of the system has to be supplemented with performance-relevant information. Thus, implementation-related quantitative properties of the system have to be derived and added to the model, most notably information describing time and resource aspects. These include performance characteristics of hardware devices, concurrency, scheduling strategies, processing speeds, bandwidth of channels, size of buffers, timer settings, various aspects of the software architecture including the possible degree of parallelism, and last but not least workload and traffic characterization.

2.1 Missing Information in SDL and MSC

In order to support a model-based performance evaluation, the information described in the following has to be associated with the formal specification in some way. An overview on the additional information is depicted in figure 1. References that have discussed these issues in the past are [Hec91, Hec96]. For a recent overview and tutorial information see [Mit97, Mit98].

System stimuli The system stimuli describes the load imposed on the system i.e. the different service requests (type and intensity) issued to the system. In the performance evaluation world this is often referred to as arrival processes, i.e. the external process that generates the service requests arriving at the system interface. Often the workload of a system comprises a set of different scenarios. A scenario defines the load that is imposed on the system concurrently, i.e. the service requests the system has to deal with at some point in time. For the different scenarios,

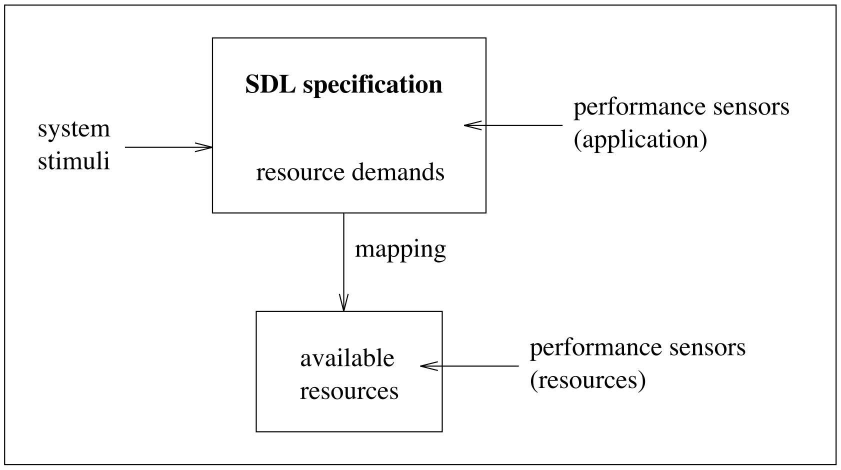

Figure 1: The additional information needed to support a performance evaluation

the number of requested services as well as its type and frequency may vary. Important parameters of the system stimuli are the requested service, the interarrival times of the service requests and their distribution. The system stimuli induce service requests, i.e. resource demands, at the components and subcomponents of the system (see below).

Available resources The performance of a system depends on the available resources. Available (limited) resources denote the units which are available for the application, i.e. to handle the load by serving the requests that are caused by the system stimuli. The most relevant resources are the processors and the communication channels of the system. Further important resources concern the available memory to hold code and data, most notably the size of buffers. Beside the capacity and time characteristics of resources, their service strategies are of importance.

Resource demands Resource demands denote the quantitative requirements resulting from the application or more specifically from the system stimuli that triggers service requests on the resources of the system. Thus, resource demands specify the cost caused by the implementation (or execution) of the parts of the system on the available resources. Examples of resource demands are the required processing times on processors and the required memory space. Note that the resource demands depend heavily on the design and implementation of the system. Thus, for a performance evaluation, the resource demands for the chosen implementation alternatives have to be derived.

Mapping Concerning the optimization of the system design and implementation, the mapping of the resource demands on the available resources is of importance. In the context of SDL, the mapping of the SDL units (e.g. blocks, processes, process instances, simple actions, and channels) on the available resources has to be specified.

Performance metrics and sensors Performance metrics denote the performance figures of the system under development that are object of the performance evaluation either as part of the goal function or as constraints. Examples of metrics are the response time to process a specific input signal, the system throughput or the utilization of some resources of the system, most notably the process input queues and the load of the processors and communication devices. Note that also a transformation of SDL-related measures may be necessary, e.g. the derivation of the throughput in bytes per second from the number and kind of transmitted signals.

In order to retrieve the values of the performance metrics from the evaluated system, sensors are needed. Sensors may range from simple probes that trace the execution to complex sensors that already provide aggregated performance figures to the outside world. In case simple probes are used, most of the evaluation of the information is done outside of the system. In the case of aggregated sensors, the evaluation is done within the system. This also allows to directly reuse this information, i.e. to influence the functional behavior of the SDL system based on this information.

Sensors may be divided in application sensors and resource sensors. Application sensors focus on the behavioral specification of the system, i.e. the parts visible within the SDL specification, while resource sensors monitor the underlying runtime system and the hardware.

Performance requirements The performance requirements specify the required values for specific performance metrics, i.e. the values or the range of the values expected from the implemented system. Thus, performance requirements can be considered as constraints of the performance optimization process. Note that performance requirements do not necessarily have to be defined formally for a performance evaluation. For different workload scenarios different performance requirements are typically given. Typical types of performance requirements are response time and throughput figures. This issue is also important when QoS requirements must be defined and observed.

2.2 General Issues

Performance-relevant issues are not completely orthogonal to other issues of the system development process. This is graphically shown in figure 2. For example, performance evaluation relies on some kind of functional or behavioral specification of the system. In addition, a performance evaluation requires information on the available resources which is also needed for other purposes. For example, the specification of the available resources is an important part of the implementation or platform description which is employed to derive the system configuration and implementation. In the following, we survey the most important issues involved.

2.2.1 Flexibility, Granularity and Ease of Use

A basic performance evaluation of an SDL specification should be easy to use even for laymen in the performance area. On the other hand, the selected approach should also support a detailed (fine-grain) performance evaluation where needed. This raises the question of flexibility versus ease of use. A flexible approach may not necessarily be very easy to use. In order to support both, flexibility and ease of use, object orientation, especially information hiding and inheritance of defaults appears to offer a remedy.

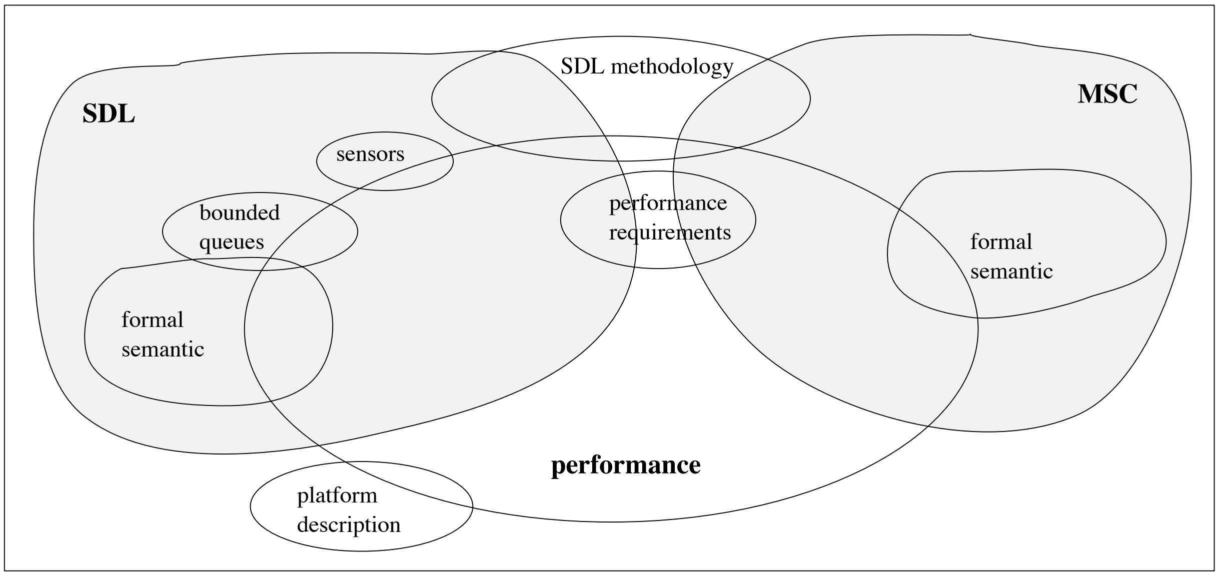

Figure 2: Issues involved with the integration of performance aspects into SDL and MSC

2.2.2 Flexibility and Reuse of the Mapping and the Available Resources

A single SDL specification may be mapped or implemented on different hardware and a chosen hardware configuration may serve for the implementation of various SDL specifications. In addition, it should be possible to assign an SDL specification (i.e. the application) on different hardware without changing the SDL specification itself. Another issue is fault tolerance which requires a dynamic change of the hardware configuration. In order to support these requirements, it seems highly advisable to keep the specification of the application and the available resources as separate as possible to support flexibility and reuse.

2.2.3 Supported Development Phases

The question of which development phases should be supported has been raised. Since SDL supports top-level design as well as detailed design, it seems that a flexible approach that supports both fine-grain as well as a more coarse-grain performance model is most appropriate.

In order to support early performance evaluation, i.e. at a stage where a single SDL process may represent a number of SDL processes in the implementation, a flexible mapping scheme is important that allows to map different parts of an SDL process on different resources.

2.2.4 Supported Application Area

The typical application area of SDL are distributed systems especially telecommunication system. Thus, performance modeling should focus on the peculiarities of these systems. This implies taking special care of the performance modeling of communications. In addition, issues concerning overload control, the dynamic reaction to overloaded resources within the SDL system, and (soft) real-time constraints are important and have to be dealt with.

Another application area where SDL seems to gain ground are small embedded real-time systems. In these systems often hard real-time conditions have to be met. In addition, these systems are often implemented on mixed hardware/software systems, i.e. systems where part of the functionality is implemented by special integrated devices as ASICs or FPGAs. In this case, parts of an SDL process may be implemented on special hardware devices. This raises the need to model the mapping of different parts of an SDL process on different resources. However, note that this depends on the granularity of the partitioning of the application in parts to be implemented in software and in hardware. Summarizing this, support of these applications requires a very flexible approach from the performance modeling viewpoint.

2.2.5 Limited Input Queues

Physical systems are finite, in particular limited buffer sizes or memory space are restrictions that may influence a system's behavior considerably. However, by definition, the input queues of SDL processes are unlimited. Thus, the results of a validation of a system under the assumption of unbounded queues may not be particularly useful. On the other hand, unlimited queues ease the formal validation. This is due to the fact that the limitation of queues results in additional actions which are not needed with unlimited queues.

The implications of introducing limited queues in SDL are far-reaching. Most important seems the question of how to deal with buffer overflow and the implication the selected solution has on the formal semantics. Alternatives for dealing with full queues are discarding the signal, throwing an exception, or restarting the system. Typically, the selected action depends on the application. Thus, a flexible, user-defined approach seems to be most appropriate from the user standpoint. However, this has serious implications on the language definition and the formal semantics of SDL.

2.2.6 Semantics of Time

The time semantics of SDL is very vague, which is due to missing agreement on the issue within Q.6/10. Currently, the time may or may not be advanced by an action or a transaction. A legal interpretation of this is to advance the time only when all queues of the system are empty (e.g. [Ver96]). As depicted in figure 3, this interpretation has the advantage of reducing the set of possible traces. Thus, the complexity of the (functional) simulation and validation is reduced. However, for a performance simulation, this interpretation is not appropriate.

From the implementational viewpoint, time is advanced be every action. This interpretation is also conform to the time semantics of SDL. However, note that the two interpretations of the time semantics are not equivalent, i.e. the two sets of system traces derived by the two interpretations of the time semantics are not necessarily equal. This is graphically depicted in figure 4.

2.2.7 Relation of Performance Evaluation to Object Orientation in SDL 92

All approaches to performance evaluation so far do not deal with object orientation in SDL. However, since object orientation is a basic concept in SDL, this needs further study. Most important is the study of the mutual interference of object orientation and performance modeling, i.e. questions concerning the inheritance of performance extensions. Important in this respect is

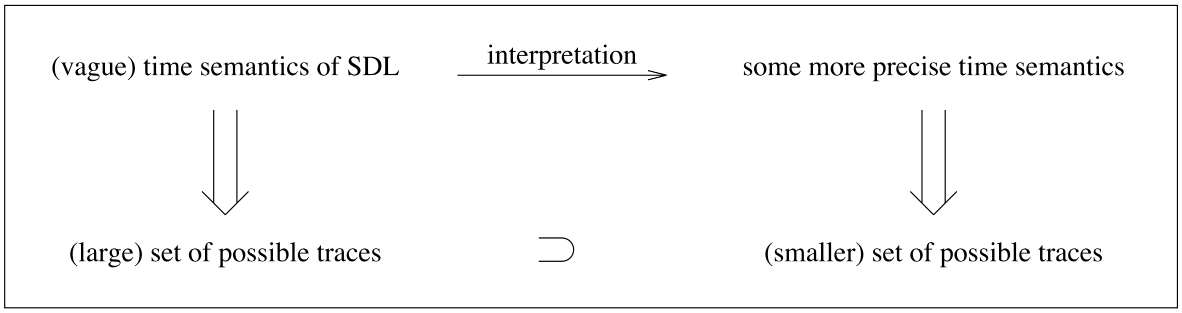
(vague) time semantics of SDL
interpretation
some more precise time semantics
(large) set of possible traces
⊇
(smaller) set of possible traces

Figure 3: Influence of the time semantics on the set of possible traces

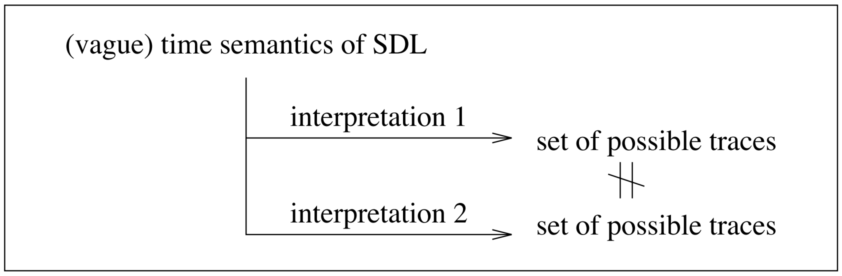

Figure 4: Difference of the system traces due to different interpretations of the time semantics in SDL

also the inheritance of defaults of performance information, which eases the user from associating performance data with each time-consuming action in SDL.

2.2.8 Tool Support versus Language Extensions and Standardization

The question to what extent the performance evaluation is a tool issue rather than a standardization issue has been raised. The important question in this respect is whether the additional information needed to support a performance evaluation are (1) directly associated with the SDL specification by introducing new constructs, (2) directly associated by using extensions to current SDL constructs, or (3) to keep the information separately and link them to the SDL specification by the respective tools.

In the first and second case an important question is to what extent the specification of the application (with resource demands) is interleaved with the specification of the available resources. For example, is it allowed to place resource specifications within regular SDL blocks? Note that this would have serious implications on the semantics of SDL. For example, it would allow to inherit and instantiate resources with the instantiation of blocks.

In any of the discussed cases, a standardization of the specification of the additional information needed for a performance evaluation is required in order to allow different performance tools to work with the same language notation. Thus, standardization is important for tool vendors to get a return on their investments and also for users to be able to apply different performance tools to the same SDL specification.

2.3 Integration of Missing Performance Information in SDL and MSC

In section 2.1, we have concentrated on the kind of information that is needed to support a model-based performance evaluation. In the following, we focus on design issues concerning the integration or association of the extra information with the SDL (or MSC) specification. This includes questions as where to put the additional information and the exact syntax as well as semantics of the extra information.

2.3.1 System Stimuli

Two basic concepts exist to specify the system stimuli. QSDL [Die95] uses SDL processes (extended to model time more precisely) to describe arrival processes. PMSC [Fal97] employs a special notation to describe system stimuli. Another approach is to have a separate workload generator linked to the SDL specification, implemented in an arbitrary (non-standardized) language or notation (e.g. [Ger97, Rou98]). For a standardized approach only the first two approaches seem to be appropriate. Due to its similarities with the current SDL standard, the use of SDL processes to specify the system stimuli seems to be most appropriate. In addition, the similarities may ease the extension of the formal semantics.

2.3.2 Available Resources

Level of detail of the resource/platform description The level of detail of the resource description is an important issue. For an early evaluation, a coarse model may be sufficient. The known approaches range from the simple specification of delays to a precise model of the underlying hardware. In [Bue96, Rou98], delays are specified from which the response times are derived considering concurrency within a set of processes. In [Die95, Ger97], queuing models with various service strategies and priorities are employed. A detailed emulation of the underlying processor hardware is used in [Ste97].

Multiple use of the resource/platform description The description of the resources or implementation platform may serve two very different purposes. It is needed for a performance evaluation as well as to specify the platform for the implementation of the SDL specification. However, these two purposes require rather different information of the machines which may be the motivation to keep two descriptions. For a performance evaluation, the capacity of the resources along with their service strategy is typically sufficient, while much more information is needed to support an implementation.

Specification of resources in standard SDL An alternative approach for the specification of the available resources is the use of standard SDL constructs, e.g. SDL processes, amended with special data types. This has the advantage that no changes to SDL, especially to its semantics are needed. On the other hand, this divides the SDL specification in parts from which code is derived and parts that are not used for the code generation. This comprises SDL processes that implement resources as well as channels and signalroutes that transfer resource requests rather than messages.

2.3.3 Resource Demands

The resource demands may be given in a generalized form, i.e. as parameters from which the specific resource requirements can be derived. Thus, the resource demand depends on the decisions concerning the mapping and the specific resources that are available. For example, the resource demands can be given either absolutely (e.g. the delay time of a data transmission) or relatively (e.g. the volume of the transmitted data) from which the absolute time can be derived. In addition, the granularity of the parts for which the resource demands are specified may differ. It may vary from the coarse-grain specification of the resource demands of a set of modules down to the fine-grain specification of the execution cost per single construct.

Level of detail of the resource demands The level of detail of the resource requests is an important issue. Alternatives are the association of a service request to actions or complete transitions in SDL [Rou98], or to add special service requests to arbitrary points in the SDL specification [Die95, Ger98]. It has been argued that service requests should be associated to SDL constructs, rather than introducing a new construct. In this context also the question of flexibility versus practicability and ease of use of the approach has been raised.

Blocking versus nonblocking resource demands Typical approaches support blocking resource requests similar to the semantics of procedure calls known from programming languages, i.e. the executing process is blocked until the given resource is acquired and the required service time has passed. In [Ger98] also nonblocking requests can be issued. This may be helpful to directly specify nonblocking resource demands, e.g. a signal output. On the other hand, nonblocking service requests are somehow less intuitive than blocking requests.

Resource demands for communication The specification of resource demands for communication (delay) should be described in a straight-forward manner. This is currently not the case with most of the available performance tools, which focus on the resource demands of the transitions rather than on communications. Thus, they require a refinement of a delayed channel by an SDL process which issues service requests to the model of the communication resource.

Overhead modeling Focusing on the association of resource demands to the SDL specification itself, a major cause of resource demands has been ignored, namely the cost involved with the runtime system, typically comprising the SDL runtime support system and the operating system. Measurements have shown that the overhead in executing an SDL specification (i.e. the SDL transitions) may be much larger than the cost of the SDL transitions itself [Hen97]. Thus, the overhead of the underlying runtime system is an important issue and should be modeled appropriately.

In performance evaluation, there are (at least) two basic approaches to include overhead. Overhead is directly added to the resource demands of the actions itself, or alternatively overhead is modeled by an abstract overhead factor that reduces the capacity of the resource which is available to the application. A problem may be the fact that the overhead typically depends on the load of the runtime system itself, i.e. the number of processes it has to handle.

Hierarchy and defaults The association of information on the resource demands to the time consuming actions of the SDL system may be very tedious. Thus, a mechanism to specify and use defaults for the resource demands is desirable. Most appropriate seems a hierarchical scheme that allows to inherit the defaults in the SDL block hierarchy. This releases the user from the task of associating a resource demand to each action. On the other hand, this approach may not be appropriate in early design phases, when the focus is on the system structure rather than its behavior and where only a small number of actions is defined.

2.3.4 Mapping of Resource Demands on Available Resources

The different SDL units have to be assigned to the resources in order to study the impact of resource consumption and contention. Several alternatives exist. The mapping can be specified implicitly by directly naming the resource within the resource requests. However, this considerably limits flexibility. Flexibility is supported by a separate description of the mapping of the resource requests on the available resources (e.g. [Die95, Ger98]).

2.3.5 Performance Metrics and Sensors

Performance sensors are used to specify the performance metrics monitored and measured during the performance evaluation. QSDL provides sophisticated aggregated performance sensors [Die98, Que98] while others rely on rather simple probes. Simple probes used in SPEET [Ste97] or PMSC [Fal97] focus on providing an interface to output performance data.

With QSDL, aggregated performance sensors are special data types maintained by the underlying system during the system simulation. The sensors can be used also to dynamically influence the functional behavior of the SDL specification based on the load figures. This allows to dynamically adapt the system to meet QoS requirements or to control overload. A different approach is known from the implementation of telecommunication systems. There, load sensors are implemented by the underlying runtime system depending on the special needs of the application. The detection of specific load situations triggers SDL signals sent to the SDL processes in charge of dealing with the problems (e.g. see [Wir98]).

2.3.6 Performance Requirements

This is an important issue for the automatic verification of performance requirements. However, it is not central for a typical performance evaluation.

Basic approaches to specify performance requirements are

- extended MSCs [Fal97, Sch98],
- temporal logics used in conjunction with performance sensors [Die98, Que98], and
- the direct use of SDL to check performance requirements within the SDL specification during the system simulation.

MSCs are especially appropriate to specify response time requirements.

The direct use of SDL to specify performance requirements seems not to be a sensible approach since the specification of performance requirements within SDL highly interferes with the functional specification. This is especially true if performance requirements represent metrics that require the monitoring of several SDL processes. In this case, a description within SDL requires to communicate the measured values using standard SDL communication mechanisms.

Currently, MSCs are less expressive than temporal logics. On the other hand, temporal logics are not acceptable for end users. An idea is to enhance the expressiveness of MSCs and map these enhanced MSCs internally on a temporal logic to support the formal verification of functional and non-functional aspects by model checking techniques [Die98].

3 Final Remarks

Support for performance evaluation and performance engineering in general is an important issue in the software engineering process. This is increasingly acknowledged by the telecommunication industry and reflected by a growing demand in SDL tools that support performance engineering activities. However, the integration of performance aspects into standardized description techniques raises a lot of questions. This is due to the fact that information relevant for a performance evaluation is not orthogonal to information needed for other activities in the engineering process. Thus, a sensitive approach for the integration of performance aspects into design methods requires to deal with various other aspects of the system engineering process, too. With formal description techniques as SDL and MSC, additional questions concerning the formal semantics arise. Examples are the semantics of time and questions of how to actively deal with limited resources.

Acknowledgements

We would like to thank all the people involved in the process of integrating performance and time aspects into SDL and MSC. We especially thank the participants at the recent Workshop on Performance and Time in SDL and MSC in Erlangen for their contributions and the lively discussion. We also thank the colleagues from the ITU-T study group 10 for many discussions on the topic.

References

[Bra93] R. Braek, O. Haugen. Engineering Real Time Systems – An object-oriented methodology using SDL. BCS Practitioner Series, Prentice Hall, 1993.

[Bue96] M. Bütow, M. Mestern, C. Schapiro, P.S. Kritzinger. Performance Modelling with the Formal Specification Language SDL. Joint Int. Conf. on Formal Description Techniques for Distributed Systems and Communication Protocols (IX) and Protocol Specification, Testing and Verification (XVI) (FORTE/PSTV'96), R. Gotzhein, J. Bredereke (eds.), Chapman & Hall, 1996.

[Die95] M. Diefenbruch, E. Heck, J. Hintelmann, B. Müller-Clostermann. Performance Evaluation of SDL Systems Adjunct by Queuing Models. SDL '95 with MSC in CASE (Proc. Seventh SDL Forum), R. Braek, A. Sarma (Ed.), Elsevier, 1995.

[Die98] M. Diefenbruch. Functional and quantitative Verification of time- and resource-enhanced SDL Systems with Model Checking. In [Mit98].

[ElS98] H. El-Sayed, D. Cameron, M. Woodside. Automated performance modelling from scenarios and SDL designs of telecom systems. Proc. of Intl. Symp. on Software Engineering for Parallel and Distributed Systems (PDSE'98), IEEE Press, 1998.

[Fal97] N. Faltin, L. Lambert, A. Mitschele-Thiel, F. Slomka. An Annotational Extension of Message Sequence Charts to Support Performance Engineering. Proc. of the Eighth SDL Forum, Evry, France, Elsevier, 1997.

[Fer86] D. Ferrari. Considerations on the Insularity of Performance Evaluation. IEEE Trans. on Softw. Eng., 12(6), 1986.

[Ger97] R. Gerlich. Tuning Development of Distributed Real-Time Systems with SDL and MSC – Current Experience and Future Issues. Proc. of Eighth SDL Forum, Evry, France, Elsevier, 1997.

[Ger98] R. Gerlich. EaSySim II SDL Extensions for Performance Simulation. In [Mit98].

[Hec91] E. Heck, D. Hogrefe, B. Müller-Clostermann. Hierarchical Performance Evaluation Based on Formally Specified Communication Protocols. IEEE Trans. on Computers, 40(4), 1991.

[Hec96] E. Heck. Performance Evaluation of Formally Specified Systems – The Integration of SDL with HIT. Doctoral Thesis, Informatik IV, Universität Dortmund, Krehl Verlag, 1996.

[Hen97] R. Henke, H. König, A. Mitschele-Thiel. Derivation of Efficient Implementations from SDL Specifications Employing Data Referencing, Integrated Packet Framing and Activity Threads. Proc. of the Eighth SDL Forum, Evry, France, Elsevier, 1997.

[ITU93] ITU-T. Z.100, Specification and Description Language (SDL). ITU, 1993.

[ITU93a] ITU-T. Z.100, Appendix I. ITU, SDL Methodology Guidelines. ITU, 1993.

[ITU96] ITU-T. Z.120, Message Sequence Charts (MSC). ITU, 1996.

[ITU97] ITU-T. SDL+ Methodology: Manual for the use of MSC and SDL (with ASN.1). Supplement 1 to Z.100, 1997.

[Kab97] H.M. Kabutz. Analytical performance evaluation of concurrent communicating systems using SDL and stochastic Petri nets. Doctoral Thesis, Department of Computer Science, University of Cape Town, Republic of South Africa, 1997.

[Mit96] A. Mitschele-Thiel, P. Langendörfer, R. Henke. Design and Optimization of High-Performance Protocols with the DO-IT Toolbox. Joint Int. Conf. on Formal Description Techniques for Distributed Systems and Communication Protocols (IX) and Protocol Specification, Testing and Verification (XVI) (FORTE/PSTV'96), R. Gotzhein, J. Bredereke (eds.), p. 45-60, Chapman & Hall, 1996.

[Mit97] A. Mitschele-Thiel, B. Müller-Clostermann. Performance Engineering of SDL/MSC Systems. SDL '97 – Time for Testing, Tutorial Notes, (Eighth SDL Forum), A. Cavalli, D. Vincent (Eds.), Institut National des Telecommunications, Evry, France, 1997, (to appear in Computer Networks and ISDN Systems, Elsevier, 1998).

[Mit97a] A. Mitschele-Thiel, F. Slomka. A Methodology for Hardware/Software Codesign of Real-Time Systems with SDL/MSC. Intl. Workshop on Conjoint Systems Engineering (CONSYSE 97), Bad Tölz, Sept. 1997, (to appear by IT Press).

[Mit98] A. Mitschele-Thiel, B. Müller-Clostermann, R. Reed (Eds.). Proc. of Workshop on Performance and Time in SDL and MSC. Report IMMD VII-1/98, University of Erlangen, 1998.

[Ols94] A. Olsen, O. Faergemand, B. Moller-Pedersen, R. Reed, J.R.W. Smith. Systems Engineering Using SDL-92. North Holland, 1994.

[Que98] QUEST and QSDL Homepage. http://www.cs.uni-essen.de/Fachgebiete/SysMod/Forschung/QUEST/. 1998.

[Ree96] R. Reed. Methodology for real time systems. Computer Networks and ISDN Systems, 28, pp 1685-1701, 1996.

[Rou98] J.L. Roux. SDL Performance Analysis with ObjectGEODE. In [Mit98].

[Rud96] E. Rudolph, P. Graubmann, J Grabowski. Tutorial on Message Sequence Charts. Computer Networks and ISDN Systems, 28, 1629-1641, 1996.

[Sch98] I. Schieferdecker, A. Rennoch. Usage of Timed MSCs for Test Purpose Definition. In [Mit98].

[Smi90] C.U. Smith. Performance Engineering of Software Systems. SEI Series in Software Engineering, Addison-Wesley, 1990.

[Smi97] C.U. Smith, L.G. Williams. Performance Engineering Evaluation of Object-Oriented Systems with SPEED. Intl. Conf. on Computer Performance Evaluation: Modelling Techniques and Tools, Lecture Notes in Computer Science 1245, Springer Verlag, 1997.

[Ste97] M. Steppler, M. Lott. SPEET - SDL Performance Evaluation Tool. Proc. of Eighth SDL Forum, Evry, France, Elsevier, 1997.

[Tel96] Telelogic Malmö AB: SDT 3.1 User's Guide, SDT 3.1 Reference Manual. 1996.

[Ver96] Verilog. ObjectGEODE – Technical Documentation, 1996.

[Wir98] K. Wirth. Overload Control in GSM – Handling the Problem in the Context of SDL. In [Mit98].