Mapping an Embedded Hard Real–Time Systems SDL Specification to an Analyzable Task Network — A Case Study*

Thomas Kolloch Georg Färber

Laboratory for Process Control and Real–Time Systems, Prof. Dr.–Ing. G. Färber
Technische Universität München, Germany
{Thomas.Kolloch,Georg.Färber}@lpr.e-technik.tu-muenchen.de

Abstract. It is undoubtedly true, that the usage of a formal specification methodology in software design will reduce the development effort, particularly as embedded hard real–time systems show increasing functional complexity. We suggest the use of the language SDL even for the design of real–time systems with hard timing constraints. Emerging problems, caused by the non–deterministic semantics of SDL, can be solved by adding EDF process activation to the SDL system model. This paper describes the different steps necessary to map a SDL system specification to an analyzable task network. Considering a SDL process as a typical server process, the mapping rules are resolving the resulting interdependencies and delays, caused by possible priority inversion and blocking. Finally the study of an application example, the "Mine Control System" proofs the usabilty of the introduced methods.
Keywords: hard real–time, schedulability analysis, design methodology, Specification and Description Language SDL, EDF

1 Introduction

Rapid prototyping of embedded hard real–time systems requires a specification language as the basis for an automated design process. The "Specification and Description Language" (SDL), originally developed for the design of telecommunication systems, suits for this purpose. SDL [1] is very similar to ROOM [2], but is a standard of the ITU and has a larger user family. In contrast to the telecommunication domain, hard real–time systems require the proof that all timing constraints will be met even in a worst case scenario, because a deadline miss may result in loss of money or even in loss of lives. Unfortunately the semantics of SDL includes non–determinisms like unpredictable ordering of messages or unpredictable process activation. The addition of earliest deadline first scheduling (EDF) to the execution scheme of SDL can resolve this drawback. A survey of the rapid prototyping design methodology is given in

* The work presented in this paper is supported by the *Deutsche Forschungsgemeinschaft* as part of a research programme on "Rapid Prototyping for Embedded Hard Real–Time Systems" under Grant Fa 109/11-1.

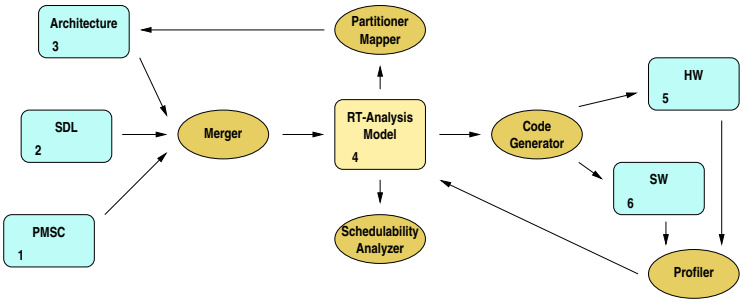

Fig. 1. Design Methodology

Fig. 1: An extended MSC [3] specification provides a description of the embedding process behaviour, i. e. describes the deadlines and the worst case scenario of the triggering external events. The architectural and detailed design is built with SDL blocks and processes. These two single models, complemented with information about the target architecture, specify all aspects of the complete system and have to be merged into the *"RT–Analysis Model"* (RTAM). In a partitioning and mapping step [4] the RTAM is linked to the target architecture [5]. Proceeding with HW/SW code generation [6], profiling should calculate the worst case execution times (WCET) of the single SDL state transitions. Now, all information, necessary for the schedulability analysis is available.

This paper is organized as follows: The next section surveys related work in the research area of timing analysis in combination with formal specification languages. Section 3 explains the schedulability proof based on EDF, followed by a description of the SDL to RTAM mapping rules. The "Mine Control System" case study (Sec. 5) evaluates the usability of the introduced methods. The last section gives a short conclusion and shows our future work.

2 Related Work

The design of embedded hard real–time systems requires a complete design methodology, which allows both the expression of functional and non–functional requirements and supports the verification of these system demands. HRT–HOOD [7] includes the explicit definition of application timing constraints and integrates appropriate scheduling paradigms with the design process, but lacks of the capability of specifying the behaviour of HOOD objects in an abstract and formal manner, e. g. with hierarchical statemachines.

Although ROOM [2] was developed for the design of real–time systems, the validation of timing properties is not included. [8] provide a heuristic, which leads to an analyzable implementation of ROOM models.

Supplementing SDL with a load and machine model, QSDL [9] uses queueing theory to calculate job and message queueing times and processor peak and average workloads.

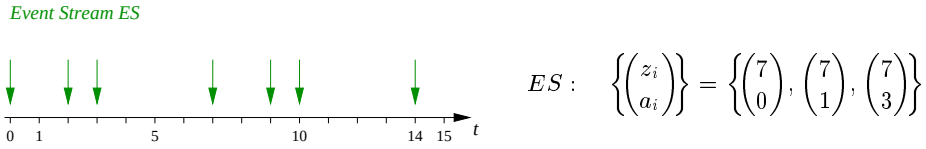

Fig. 2. Event Stream Example

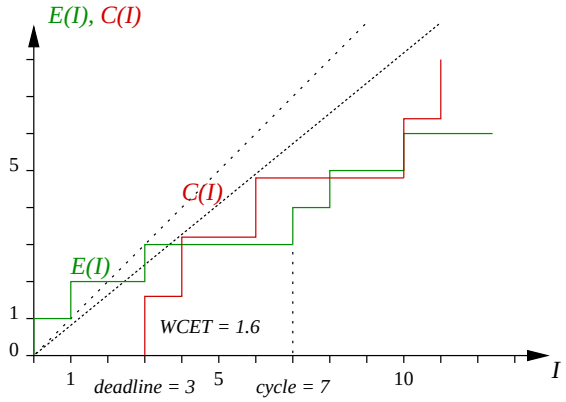

Fig. 3. Event Function $E(I)$ and Requested Computation Time $C(I)$

3 Schedulability Analysis

This section introduces the EDF based schedulability analysis for event driven hard real–time systems [10]. First, the characteristics of the embedding process have to be described, i. e. the timely behaviour of stimulating events using *Event Streams* (ES). ES describe the maximum possible number of events i of a certain type within an interval a_i [11]. Considering the occurrence of interrupts on the left side of Fig. 2, the resulting ES[1] is shown on the right. This leads to an *Event Function* $E(I)$, expressing the number of events per interval I (Fig. 3).

Single tasks are characterized by their WCET $c_{max,j}$ and a deadline d_j for the triggering event. The internal structure of an analysis task consists of atomic *receive* and *semaphore obtain* operations at the beginning of each task, a preemptive task body without calls to operating system services and finally atomic *send* (non–blocking), *semaphore release*, *timer* and *in, out* operations.

The $C(I)$ Function is defined as maximum computation time requested and due within interval I. For a single task $C_j(I)$ can be calculated easily from $E_j(I)$ by shifting by the deadline d_j and multiplication with the WCET $c_{max,j}$ (Fig. 3). While the resulting $C(I)$ for a number of *independent* tasks on a computing node is simply the sum of all $C_j(I)$ functions, Gresser developed an algorithm

[1] there are no 2 simultaneous events, i. e. only 1 event occurs in interval 0, a maximum of 2 events in interval 1 and a maximum of 3 events in interval 3, repetition with cycle 7

to determine $C(I)$ for a network of communicating tasks, taking into account dependencies[2] of the triggering events, precedence constraints, inter node communication and mutual exclusion. For EDF he proved, that all tasks on one node meet their deadlines, if the resulting $C(I)$ always runs under the bisector which specifies the available computing time in each interval.

$$C(I) \leq I \qquad \forall \quad I \geq 0 \tag{1}$$

One crucial point in schedulabilty analysis is the avoidance of priority inversion in critical regions. For the analysis of complex structures of mutual exclusion, i. e. several tasks in different and overlapping critical regions, [10] explains two strategies to resolve the edges of a *"priority inversion graph"*: Either by shifting the deadline of the task to a new shorter, but during runtime fixed deadline or by taking into account a dynamic deadline inheritance protocol.

4 Mapping SDL to the RT–Analysis Model

There are prerequisites, which have to be satisfied for the translation of a SDL system to the RTAM: A static allocation of processes to processing units is mandatory and dynamic process instantiation can not be considered in the moment, i. e. most of the object oriented language extensions of SDL–92 are forbidden in a SDL model. Further language restrictions are: no usage of services and priority inputs, continuous signal trigger conditions and signal *SAVE* statements. A general requirement — valid for all real–time systems — is saveness. Saveness means, that in spite of any possible stimulation the system will regain a save (rest) state, especially there are no receive/send – receive/send loops. For a save system the algorithm, described in Sec. 4.4 will terminate.

4.1 SDL Process — Server Behaviour

The similarity of the analysis model (Sec. 3) and the statemachine structure of the SDL process allows an automatical RTAM generation. For this purpose the hierarchical SDL block composition is transformed into a network of communicating (leaf) SDL processes. If there are no dependencies between these processes, the simple addition of the WCET $c_{max,j}$ will be allowed for the calculation of the overall $C(I)$. Therefore the interferences within this process structure have to be resolved.

Unfortunately, a SDL process (statemachine) has usually server process behaviour, with different sources of incoming messages and different destinations of outgoing messages. The RTAM, corresponding to a server process, has the process duplicated and protected in an area of mutual exclusion (Fig. 4(a)). Duplication takes into account a possible worst case delay, caused by an earlier message, mutual exclusion protects the ordering of execution. Preemption of

[2] "*event dependency matrix*" (EDM)

Fig. 4. (a) Server Process and (b) Process Precedences

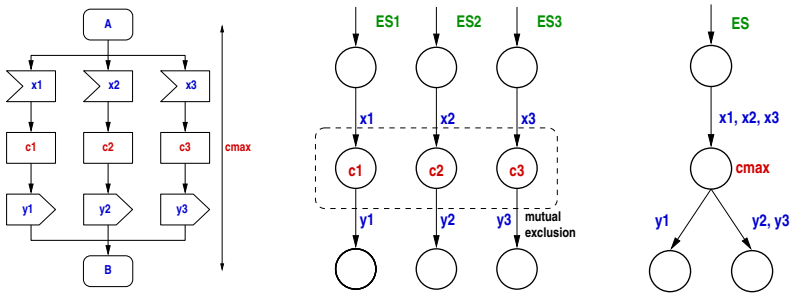

Fig. 5. Analysis Model — (a) different and (b) same message source

a state transition is not allowed in SDL semantics, but can occur in the implementation. Depending on the type of implemented task system (Sec. 6), a transaction may be even preempted by a state transition of the same SDL process with shorter deadline, therefore the access to common SDL state information has to be synchronized by mutual exclusion. Using "tight integration" for code generation, which maps to one task per SDL process and one message queue per task, the order of computation is managed by the queue and therefore the mutual exclusion is not necessary in the implementation. But regarding the possible delay, caused by the computation of a reaction to an earlier received message, the analysis must take into account a mutual exclusion too.

4.2 Mapping SDL Statemachines

Depending on the source of the incoming messages (one single ES or serveral different ES), a statemachine has to be mapped to different RTAM trees, either with individual execution times c_i and mutual exclusion (Fig. 5 (a)) or to one single task (Fig. 5 (b)) with $c = c_{max} = Max(c_i)$.

If the same message triggers a transition in different states (Fig. 6, states A, B, C), the RTAM only takes into account the maximum computation time $c_{max} = Max(c_i)$. Depending on the destinations of the outgoing messages, the target analysis node can be derived. Assigning an asterix symbol to states and message receive statements, SDL syntax provides a kind of behavioural hierarchy like ROOMCharts [2]. This conforms to a state or a message enumeration and can be translated using the former explained mapping rules.

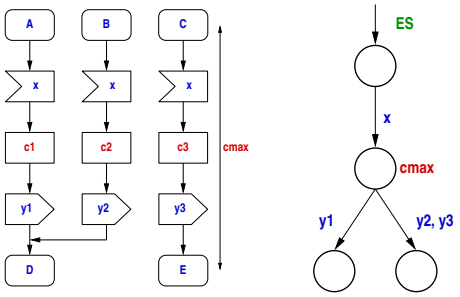

Fig. 6. Analysis Model — same message in different states

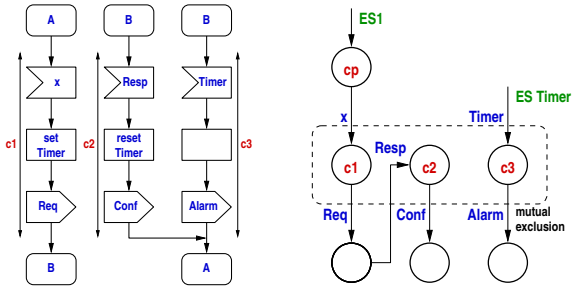

Fig. 7. Analysis Model — Dependent Timer

4.3 Mapping SDL Timers

SDL Timers are internal events, which can occur independently or dependently on external events. An example of a independent timer event is the cyclic activation of polling tasks in the case study (Sec. 5). A dependent timer event is e. g. the surveillance of a timely response to a server request. The first type has to be mapped to an analysis node with its own, the timer behaviour describing ES (2). The dependent timers mapping rule is identical to the former one, but supplemented with additional event dependency information ([10], Fig. 7). The minimum distance between the stimulating external event and the internal timer event results from the sum of minimum computation times of the tasks in the same precedence system plus the timer interval (3).

$$ES_{Timer} = ES_1 = \begin{pmatrix} z_1 \\ 0 \end{pmatrix} \qquad (2)$$

$$EDM = \begin{pmatrix} z_1 & c_p + T \\ 0 & z_1 \end{pmatrix} \qquad (3)$$

4.4 Mapping Algorithm

The mapping algorithm starts the transformation at each triggering external event or each independent SDL Timer. Then, for each environment handling

Table 1. Mine Control System — Timing Constraints and Analysis Parameters

	P/S	Cycle z_i	Interv. a_i	Deadl. d_i	WCET c_i	Deadl. d_i	WCET c_i	Deadl. $d_{i,mutex}$	WCET $c_{i,mutex}$
CH_4 Sensor	P	$1\,s$	$0\,s$	$1\,s$	$0.350\,s$	$1\,s$	$0.350\,s$		
CO Sensor	P	$5\,s$	$0\,s$	$5\,s$	$0.125\,s$	$5\,s$	$0.075\,s$	$1\,s$	$0.050\,s$
Air Flow	P	$5\,s$	$0\,s$	$5\,s$	$0.125\,s$	$5\,s$	$0.075\,s$	$1\,s$	$0.050\,s$
H_2O Flow	P	$3\,s$	$0\,s$	$3\,s$	$0.075\,s$			$1\,s$	$0.075\,s$
H_2O Level	S	$100\,s$	$0\,s$	$20\,s$	$0.150\,s$	$20\,s$	$0.025\,s$	$1\,s$	$0.125\,s$
Operator	S	$10\,s$	$0\,s$	$1\,s$	$0.175\,s$	$1\,s$	$0.175\,s$		

SDL process, the consecutive SDL processes are identified by means of *SEND* statements. In a next step dependent SDL Timers have to be detected and the assignment of their own ES have to be prepared. Subsequently all state transitions, triggered by the same message, are eliminated, and a replacement with maximum computation time has to be defined. Finally, the analysis node, appropriate to the resulting dependent state transitions, has to be multiplied in an area of mutual exclusion. The algorithm continues with the next SDL process in the precedence system, until no more *SEND* statements can be found.

Applying this transformation to a SDL system, the resulting RTAM will consist of serveral independent analysis task precedence systems (Fig. 4 (b)), whereby each network is triggered by a different type of ES. The single branches of the RTAM tree are linked by regions with mutual exclusion. All analysis nodes in one precedence system have the same deadline, i. e. are fixed to the messages, sent through a precedence system. This leads to the fact, that different SDL state transitions in one SDL process may have different timing constraints.

5 The Mine Control System

The "Mine Control System" case study is originally described in [7, pp. 145–224] as an example of modelling a real–time system with HRT–HOOD. The purpose of the pump is to manage the water level in a mining environment.

5.1 Functional and Non–Functional Requirements

The pump monitors the water level in a sump. According to a high level detector or to operator interaction, the pump is turned on and the sump is drained, until a low level detector responses or the pump is turned off by the operator. The pump should only be allowed to operate, if the CH_4 concentration is below a critical level. The operator console and the level detectors communicate via interrupts with the pump control station. Additional sensors for monitoring the environment are polled in different cycles. Critical levels of CH_4, CO or an unsufficient air flow must be signalled to the operator as an alarm. In case of an operating pump, the water flow in the pump can be measured. A critical

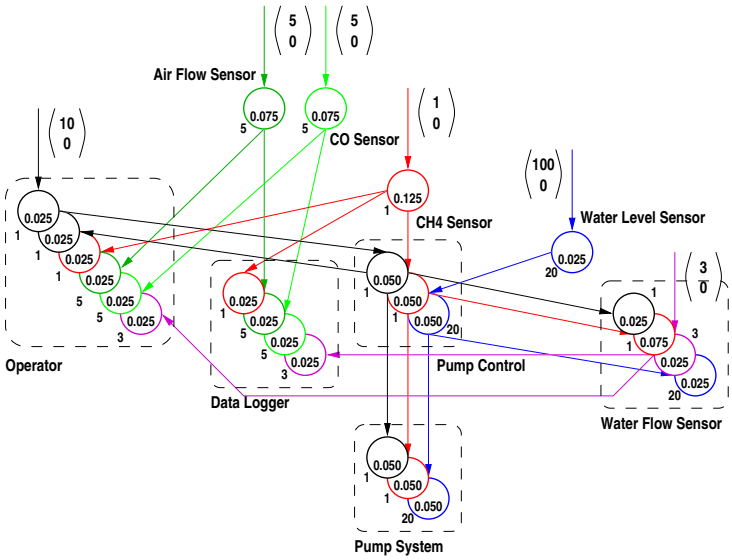

Fig. 8. Complete Analysis Model

CH_4 level must lead to an undelayed shut off of the pump. A summary of the tasks characteristics, their cycle times, respectively the minimum distance of the stimulating external events and the appropriate deadline is listed in Tab. 1.

5.2 Analysis Model and Analysis Results

As a representative, the analysis model of the pump control process (Fig. 9 (a)) will be explained. Although six different messages are consumed, the number of different message sources evaluates to three, therefore the analysis node must be tripled in an area of mutual exclusion. The upper left node in the pump control region of Fig. 8, triggered by an operators message, has three message outputs, whereby one is the response to the operators request. Since there are no further send statements in the consecutive processes, the forks of the analysis precedence tree (mapping algorithm) end in the next nodes.

The execution times are derived from the HRT–HOOD example [7, pp. 145–224]. To demonstrate the influence of priority inversion avoidance, the values of the WCETs are multiple oversized, compared to the complexity of the processes. The WCETs of the individual SDL state transitions can be seen in the appropriate analysis nodes in Fig. 8. Summing up the WCETs c_i of the nodes of one precedence system results in analysis parameters, summarized in Tab. 1.

In a next step a possible priority inversion in critical regions is considered by shifting the task deadlines to the shortest deadline in the region. The stimulating event stream remains untouched by this manipulation. The new deadlines calculate to $d_{i,mutex} = 1\,s$. The sums of the $c_{i,mutex}$ in the critical regions of one precedence system are shown in the last column of Tab. 1.

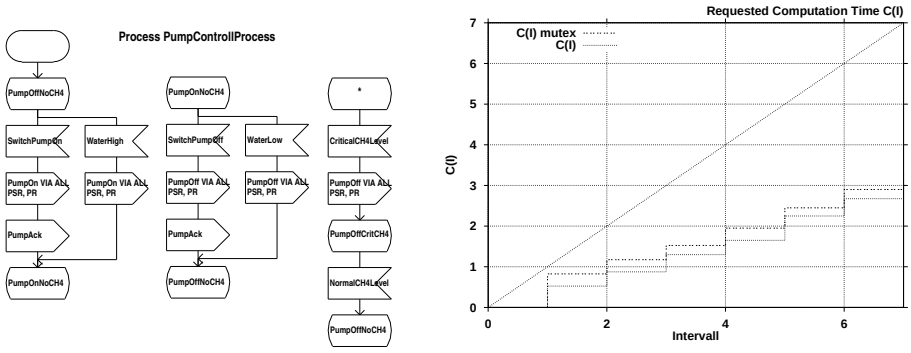

Fig. 9. (a) Pump Control SDL Statemachine and (b) Analysis Results

The result of the schedulability analysis can be seen in Fig. 9 (b). The distance between the $C(I)$ and the $C(I)_{mutex}$ functions shows the influence of the manipulated deadlines. By shortening the deadlines, the analysis has to schedule more computation time as possibly needed to meet all deadlines, i. e. the laxity, available for further tasks, may be lost. This leads to design rules to evade this effect: Keep computation times in critical region as small as possible; Minimize the number of nodes in critical region, i. e. avoid SDL process server behaviour, and minimize the number of critical regions by combining similar processes, e. g. combine the cyclic polling processes.

6 Conclusion

In this paper, we focused on the integration of a schedulabilty proof in the design flow for embedded hard real–time systems, based on the language SDL. This integration is done, by adding EDF semantics to SDL process activation to resolve non–predictable system behaviour and by mapping the SDL system to an analyzable task network.

The introduced SDL to RTAM mapping rules and algorithm allow the automation of this transformation and therefore the integration in a rapid prototyping design environment. To cover the complete syntax and semantic of SDL, further mapping rules are necessary. Forced by server behaviour of SDL processes, the mapping creates many areas of mutual exclusion. For this reason the system designer should get support by design rules, which help to develop an efficient and analyzable software architecture.

A trade–off appears, regarding the way of code generation,[3]done by the SDT CASE tool. Considering the fine granularity of a SDL process, — the transitions of a SDL statemachine are normally short — the generated task system is fine granular too. This leads to the phenomenon, that the resulting system will mainly do task switching, instead of processing real data. A solution for this

[3] tight integration to the “Real–Time Executive for Multiprocessor Systems RTEMS”

effect could be the implementation of a complete analysis precedence system in a single task (similar to [12]). The minimization of the number of context switches and the number of *receive* and *send* calls, should result in a more efficient implementation.

References

[1] ITU–T. *ITU–T Recommendation Z.100: CCITT Specification and Description Language (SDL)*, June 1994.

[2] Bran Selic, Garth Gullekson, and Paul T. Ward. *Real–Time Object–Oriented Modeling*. John Wiley & Sons, Inc., 605 Third Avenue, New York, 1994.

[3] ITU–T. *ITU–T Recommendation Z.120: Message Sequence Chart (MSC)*, September 1994.

[4] Georg Färber, Franz Fischer, Thomas Kolloch, and Annette Muth. Improving processor utilization with a task classification model based application specific hard real–time architecture. In *Proceedings of the 1997 International Workshop on Real–Time Computing Systems and Applications (RTCSA'97)*, Academia Sinica, Taipei, Taiwan, ROC, October 27–29 1997.

[5] Franz Fischer, Thomas Kolloch, Annette Muth, and Georg Färber. A configurable target architecture for rapid prototyping high performance control systems. In Hamid R. Arabnia et al., editors, *Proceedings of the International Conference on Parallel and Distributed Processing Techniques and Applications (PDPTA'97)*, volume 3, pages 1382–1390, Las Vegas, Nevada, USA, June 30 – July 3 1997.

[6] Franz Fischer, Annette Muth, and Georg Färber. Towards interprocess communication and interface synthesis for a heterogeneous real–time rapid prototyping environment. In *Proceedings of the 6th International Workshop on Hardware/Software Co–Design — Codes/CASHE '98*, pages 35–39, Seattle, Washington, USA, 15–18 March 1998. IEEE, IEEE Computer Society Press.

[7] Alan Burns and Andy Wellings. *HRT–HOOD: A Structured Design Method for Hard Real–Time Ada Systems*. Elsevier Science B. V., Amsterdam, The Netherlands, 1995.

[8] M. Saksena, P. Freedman, and P. Rodziewicz. Guidelines for automated implementation of executable object oriented models for real-time embedded control systems. In *Proceedings of the IEEE Real–Time Systems Symposium (RTSS'97)*, San Francisco, California, December 2–5 1997. IEEE Computer Society Press.

[9] Marc Diefenbruch, Elke Heck, Jörg Hintelmann, and Bruno Müller–Clostermann. Performance evaluation of sdl systems adjunct by queueing models. In *SDL'95 With MSC in CASE, Proceedings of the Seventh SDL Forum*, pages 231–242, Oslo, Norway, September 1995.

[10] Klaus Gresser. *Echtzeitnachweis ereignisgesteuerter Realzeitsysteme*. Number 268 in Fortschrittsberichte VDI, Reihe 10. VDI–Verlag, Düsseldorf, 1993. Dissertation am Lehrstuhl für Prozessrechner, Technische Universität München.

[11] Klaus Gresser. An event model for deadline verification of hard real–time systems. In *Proc. Fifth Euromicro Workshop on Real Time Systems*, pages 118–123, Oulu, Finland, June 1993. IEEE.

[12] Ralf Henke, Hartmut König, and Andreas Mitschele-Thiel. Derivation of efficient implementations from SDL specifications employing data referencing, integrated packet framing and activity threads. In *Proceeding of the Eighth SDL Forum, SDL'97 Time for Testing SDL, MSC and Trends*, pages 397–414, Evry, France, September 1997. Elsevier Science Publishers B.V.