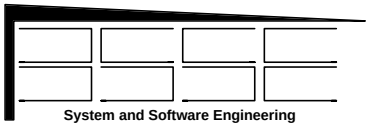
Experience with Validation by Simulation, Automated Code Generation and Integration

'DASIA 97'
- Data Systems in Aerospace -
Sevilla, Spain
May 26 - 29, 1997

Rainer Gerlich
BSSE
(Bodan) System and Software Engineering

Auf dem Ruhbuehl 181
D-88090 Immenstaad

Phone: +49/7545/91.12.58
Mobile: +49/171/80.20.659
Fax: +49/7545/91.12.40
e-mail: gerlich@t-online.de



Experience with Validation by Simulation, Automated Code Generation and Integration

Rainer Gerlich

BSSE
(Bodan) System and Software Engineering

Auf dem Ruhbuehl 181
D-88090 Immenstaad, Germany

Phone +49/7545/91.12.58 Mobile: +49/171/80.20.659
Fax +49/7545/91.12.40
e-mail: gerlich@t-online.de

Abstract: During DASIA'96 the CIVE approach [1] was presented which is based on a computer-integrated life-cycle starting with system validation right from the beginning. Such early validation is performed by simulation considering functionality, behaviour and performance. The integration of the life cycle phases is achieved by continuous expansion of the simulation models from specification to design, refinement and automated code generation from the models.

To get an early and immediate feedback from the system-under-development such models are executable during simulation on host and on target (see also [2]) under conditions which are representative for the final target system.

During previous ESA/ESTEC projects [3-5] first steps to an integrated life cycle approach [6] were performed and the tool environment EaSySim (I) was established.

During the last year a new tool environment EaSySim II[1] [7] has been established which overcomes the weakness of the first version. The existing integration capabilities have been exploited and more consequently applied. This helped to succeed with validation by simulation for more complex systems, to extend the supported application type from embedded to more general areas including MMI's and databases, and to integrate Ada software. Also, the procedure for code generation has been better organised. Now, it is possible to move from early validation by simulation to the target system within 15 minutes[2].

So an engineer can analyse a system by the powerful means which are available during simulation, such as capabilities for analysis of data flow and verification of behaviour by exhaustive simulation, and he can already execute the system on the target during each development step without loosing too much time for handling of the transition.

Keywords: Verification and validation, automated code generation, simulation, tool and method integration, SDL, Ada, ObjectGEODE, VxWorks, formal methods, databases, MMI

1. INTRODUCTION

The CIVE (Computer Integrated Validation Environment) approach [1] as presented during DASIA'96 uses a coherent transition between the life cycle phases and applies validation right from the beginning of development. This has to be supported by tools to be sufficiently efficient.

SDL [8] is used for formal definition of a system's behaviour. The ObjectGEODE [9] simulator and its extension by EaSySim II verify and validate behaviour and performance. The ObjectGEODE code generator automatically produces C code from the SDL representation to interface with a number of (real-time operating systems) such as UNIX or VxWorks [10] / 80x86 target

In the EaSySim (I) environment the code generation procedure was rather complex and a lot of time was needed to configure the tool environment for the simulation and all the code generation branches (Fig. 1).

EaSySim II now provides a procedure which allows to install the complete EaSySim II and application environment and to run through simulation, target code generation and execution within fifteen minutes for a sample application.

The application defined in SDL/C will be executed after simulation (1) on the host (Sparc/UNIX platform) and (2) on a PC bare machine on top of the real-time operating VxWorks. The executable code for UNIX and VxWorks is generated on-line during the presentation.

The fast transition between simulation and target makes computer-integrated validation reasonable and efficient. During simulation the data flow, exceptions, formal correctness of behaviour and performance are analysed. On the target system the performance results of simulation can be calibrated.

This way both environments complement each other and minimise the risk due to the detailed feedback they are providing.

---

[1] EaSySim II is a tool environment developed by BSSE based on the experience made during execution of the OMBSIM pilot application [4] and the follow-on ESTEC project DDV [5].

[2] This will be confirmed by an on-line demo during the conference

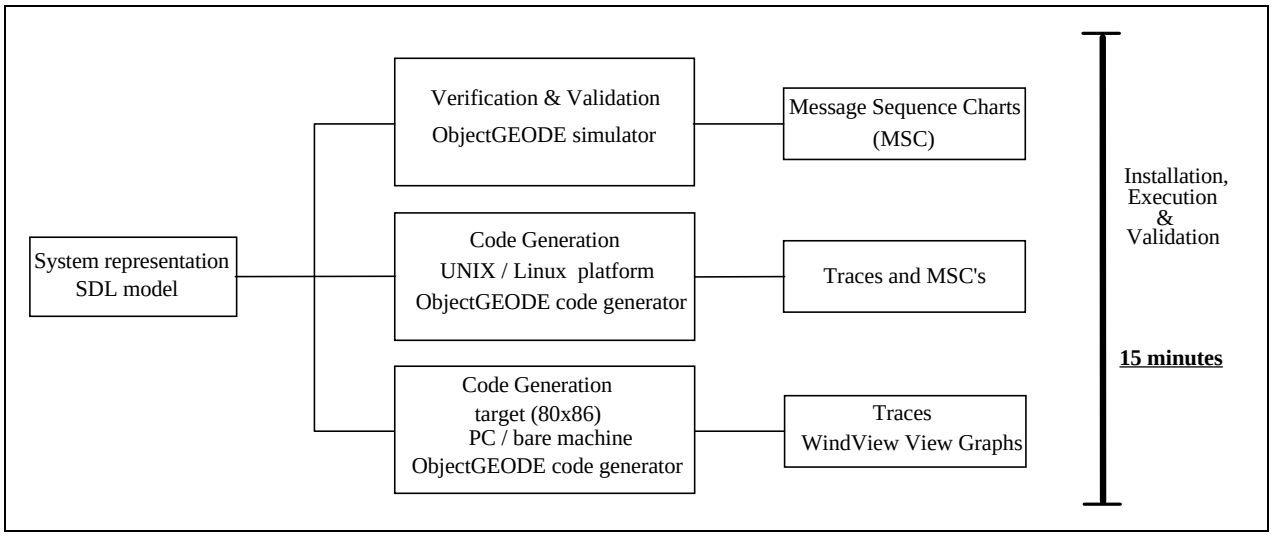


Fig. 1: From Verification &Validation to Code Execution

2. THE 2-DIMENSIONAL LIFE CYCLE

By performing the transition from simulation to the target system during each life cycle phase rather than from the beginning to the end of the life cycle, the current "one-dimensional" life cycle is extended to a "two-dimensional" life cycle (Fig. 2). It allows during each phase simulation, code generation and execution on the target system.

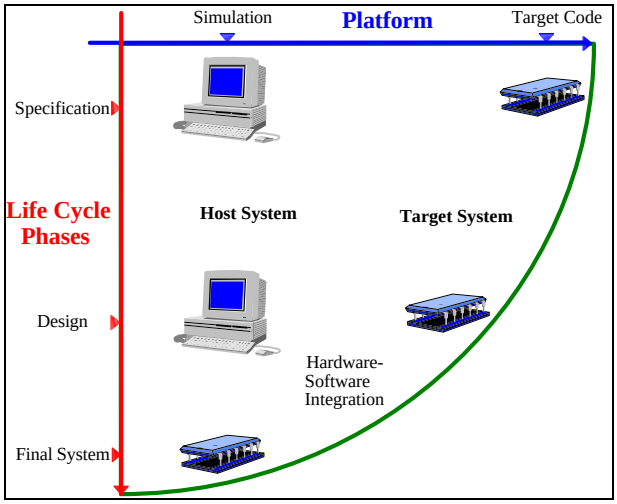


Fig. 2: The 2-dimensional Life Cycle

Hence, the second dimension allows to evaluate already at the beginning what is available in the conventional (1-dimensional) life cycle only at the end and to continuously check during development if one is still on the right way.

The generated code converges from an early, representative version towards the final version at the end of the project. Specification and design are always executable, they are validated by simulation and by execution on the target system at each step of refinement.

System refinement and extension go along with continuous hardware-software integration on the target starting with emulation of hardware and ending up with drivers which connect the final software with the real hardware.

3. VALIDATION BY SIMULATION AND CODE EXECUTION ON TARGET

Parts of the validation procedure are discussed below for an embedded system which consists of five components (Fig. 3): the Data Management System processor (dms), a sensor and actuator and the bus connecting them all. EaSySim II injects external commands and stimulates the system. Data are requested from sensors, processed by the DMS processor, which in turn sends commands to the actuators for control of the spacecraft.

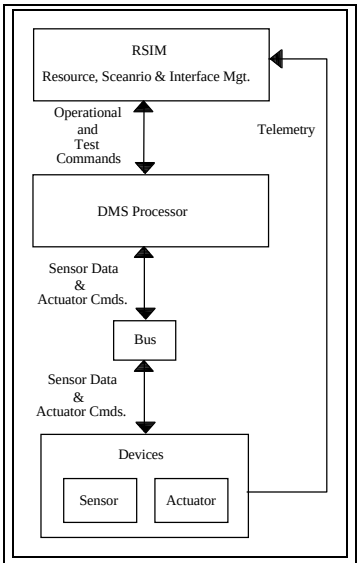


Fig. 3: DMS Example

The ObjectGEODE simulator produces Message Sequence Charts by which the data exchange between system components is visualised graphically (Fig. 4). This feature allows a human being to check the logical flow of data. A (simplified) MSC can be generated from the target code, too (UNIX or real-time operating system). In case of VxWorks a graphical view of task execution can be obtained as shown by Fig. 5. By these means the system under development can be well monitored during the life cycle.

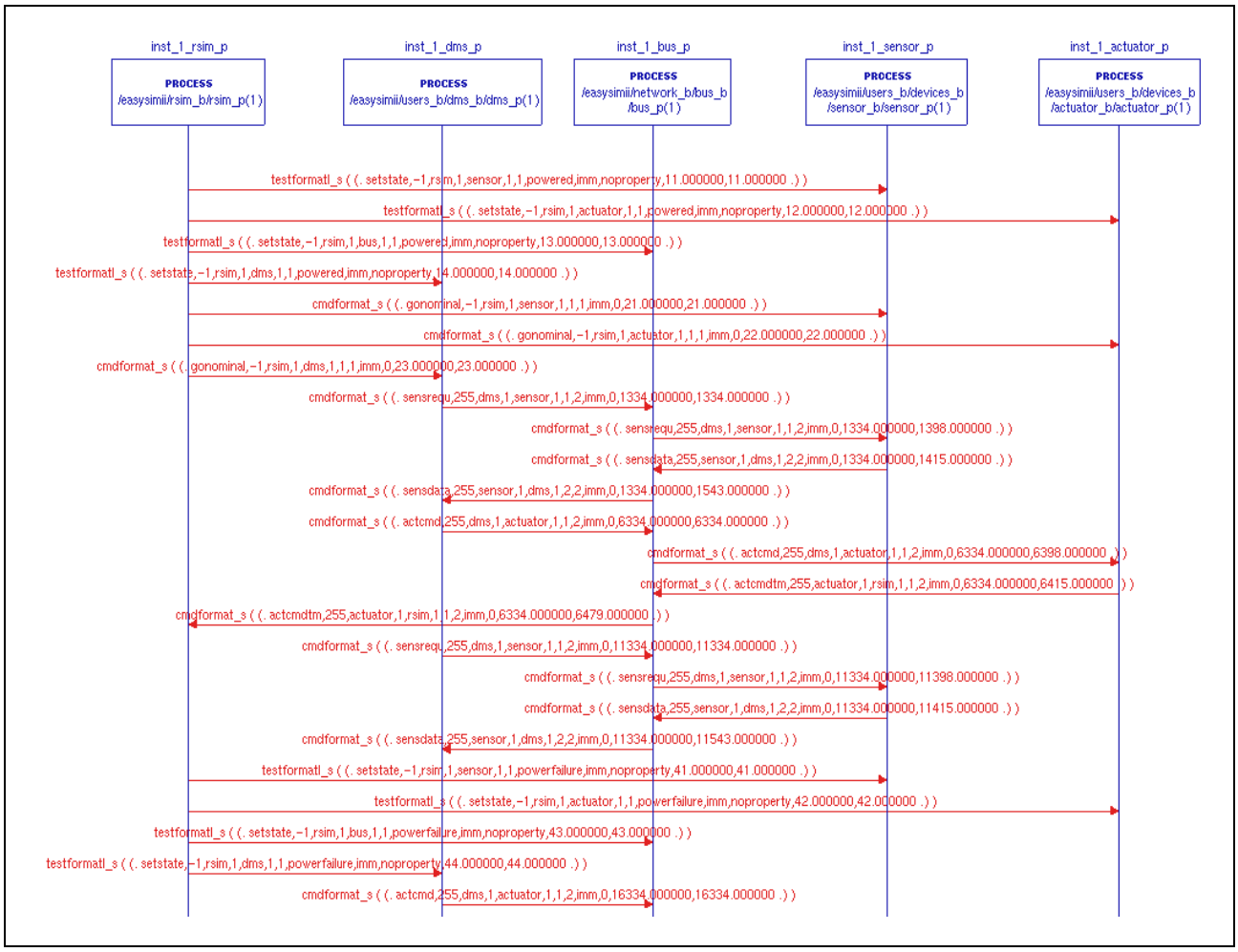


Fig. 4: Generated MSC (Simulation)

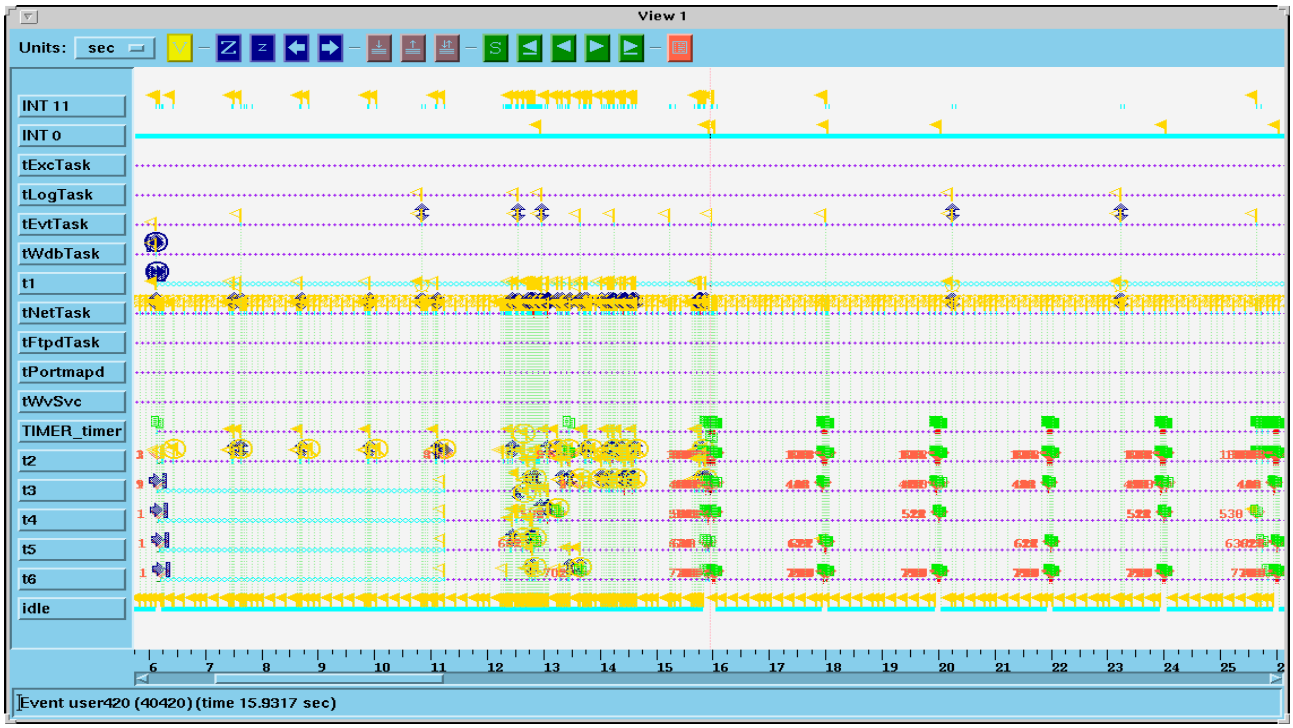


Fig.5: Recording of Target System Execution

The performance analysis capabilities of EaSySim II are made visible by the MSC of Fig. 4: Each message exchanged between the components, e.g. a sensor data request (sensrequ, line 8 from top) includes two time stamps at the end of the data record: a time stamp indicating when an activity like a sensor data request was initiated, and another time stamp which gives the actual time when the message leaves a component.

A sensor data request is initiated at t=1334 in the DMS component (message line 8, Fig. 4). It enters the bus and leaves it at t=1398 after a delay of 64 time units. It enters the sensor component and leaves it at t=1415 after a processing delay of 17 time units. Finally, the data arrive at t=1543 in the DMS component (line 11) yielding a data acquisition time of $\Delta t=145$ time units.

The experienced reader may recognise that there is a bug in the implementation of the bus: this bug is visualised by the time information included in the MSC traces.

By line 20 a power failure is injected into the sensor component from EaSySim II driven by an external command. As such a message does not use the physical network it is not time-stamped, but includes a stamp as defined by the engineer. After injection of this fault the system still behaves normal because the requirement is that after the first power failure the system shall still be fully operational. So this MSC expresses this need.

An MSC can also be used as a specification for the data flow of the developed code and the ObjectGEODE simulator searches for such patterns during simulation. It reports whether the pattern is found or not.

By Fig. 5 the periodic processing of sensor data and generation of actuator commands is visualised. The data flow from the DMS processor (t3) via bus (t6) to the sensor (t4) and to the actuator (t5). The period is 4 time units. Sensor data are acquired at the beginning of the cycle, actuator commands are issued in the middle of the cycle after 2 time units.

4. EXPERIENCE WITH METHODS AND TOOLS

The work executed during the ESA/ESTEC projects [3-5] showed that the EaSyVaDe approach [6] (which is an outcome of the OMBSIM project [4]) can coherently be applied over the life cycle. However, one strongly needs to consider that SDL (on which the EaSySim environment is based) is not a general purpose language. Its strong point is that it formalises specification of behaviour. It has been targeted to support protocol verification. When applying it to more general application areas certain rules must be followed to succeed with system verification.

SDL tools do provide the capability for exhaustive simulation which is very powerful for analysis of a system's behaviour. But already at low complexity the physical boundaries of computer resources may be reached[3]. In such a case this capability is lost because the proof of correctness is not possible due to exhausted computer resources.

Hence, the goal of an engineer must be to monitor his SDL system carefully and to introduce the right means - as soon as needed - so that the computer resources will not be exceeded.

Other formal methods like B [11] or Z [12] are targeted for another type of applications: sequential programming and algorithms. Such tools may also not give the performance and provide results as expected when they are applied outside their dedicated application domain, e.g. to describe behaviour and to implement distributed systems.

To summarise: each tool has strong and weak points. If one does not carefully consider such constraints one may fail in total.

As a system's properties should not be compromised by a method or tool, appropriate solutions need to be considered. A reasonable solution is to partition the problem into smaller pieces corresponding to independent parts of the system and different application domains. Then for each such piece the most appropriate method and tool can be used.

This leads to the following approach: hide algorithms from SDL and apply more appropriate formal methods and tools. Vice versa, behavioural aspects should be treated by SDL because it provides the better support.

Consequently, we get a heterogeneous method and tool approach and integration of all such dedicated components for verification and validation of the complete system is getting a challenging issue. Such an approach is supported by EaSySim II due to its tailored communication and integration capabilities.

---

3 Filtering techniques which may be applied to master this problem are not considered as an appropriate means by which representative validation results can be achieved. The justification of this position cannot be given here in detail.

5. TOWARDS AN INTEGRATED, HETEROGENOUS APPROACH

Now, the idea is to combine the strongness of SDL and its related tools with benefits of methods and tools from the other application domains as shown by Fig. 6. Such areas may be: algorithms, human interfaces, databases.

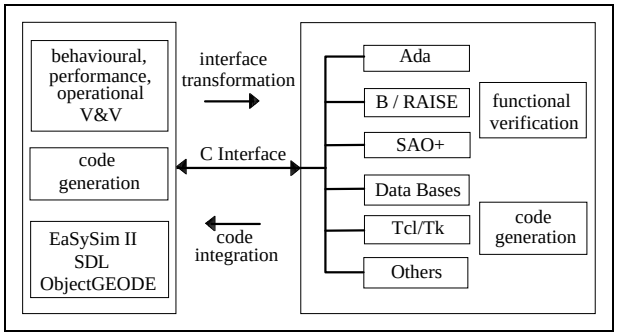


Fig. 6: Complementary Verification and Validation

In such a heterogeneous tool approach EaSySim II serves as an integration platform allowing to define a formal behavioural skeleton by SDL into which other software just can be plugged-in. This concept has already been applied for simulation and code generation.

It is reasonable to implement algorithms by other languages like Ada, C or C++ which provide a richer type concept. If formal verification and validation are needed, methods and tools like B [11], Z [12], RAISE [13] and SAO+ [14] may be applied. Also, links to database software and GUI builders like Tcl/Tk [15] or Java [16] may be established.

During the presentation it is demonstrated that external processes (a Tcl/Tk window) can be started and accessed from SDL by socket communication.

Via the C interface, which SDL tools like ObjectGEODE are supporting, the different system parts can be integrated on code level: either existing code or code simultaneously developed with other tools.

It has already been demonstrated that SDL operators may be implemented in Ada and attached to SDL. This is possible in general for every Ada83 compiler which supports a C interface or for Ada95 compilers.

Hence, SDL takes over the control (behavioural) part and reacts to incoming demands in accordance with the system state (described by Finite State Machines). This allows to be on the safe side for the decision logic, but leaves the functional part to the appropriate verification and validation means.

Currently an approach is considered for which a transformation of the interfaces (as defined in SDL) from left part to right part of Fig. 6 can be automated. In the opposite direction just the generated code would be integrated into the SDL environment.

So far, the main application field of SDL was limited. This will change in future. One of the next BSSE projects will make use of databases and man-machine interfaces based on the integration capabilities as provided by EaSySim II and SDL/ObjectGEODE.

This possibility for integration brings the existing validation capabilities to new application domains.

6. RECOMMENDATIONS

As was explained a synthesis of existing tool environments is the best way to get optimum support for development, verification and validation of systems. To be successful the tools must provide open interfaces for integration of system parts which are developed in dedicated environments.

A system's behaviour is the result of decision-making which coordinates the functional branches. Therefore such branches need to be integrated into a behavioural tool environment. SDL as language and ObjectGEODE / EaSySim II are an example for such an integration platform.

To gain advantage from early system validation in top-down manner the transition from simulation to target system needs to be efficient. This requires good organisation and a tool environment which allows for an easy and fast transition.

7. CONCLUSIONS

The capability of SDL and ObjectGEODE / EaSySim II to immediately move from the verification and validation environment to the target system (or its early representation) provides enhanced means to check if and how close the current development stage is to the user's expectations. This 2-dimensional life cycle allows to smoothly approach the final system in a controlled manner from the very beginning.

The code generation capabilities, especially the automated generation of all the code needed for interprocess communication, simplify significantly the development of distributed systems and help to save effort.

The openness of SDL allows to integrate code generated by other tools and verification environments and to apply it to a larger application domain. The integrated code can be used for verification and validation on the host and on the target. So each of the different methods and tools can take advantage of the capabilities of the other ones and there is no need to implement in one environment capabilities which are already available.

In such a heterogeneous environment SDL will take the part of control logic and decision-making while the more functional parts will be provided by the other, complementary tool environments.

Currently, work is going on to exploit an automated transformation to another method and environment from an interface expressed in SDL.

8. REFERENCES

[1] R. Gerlich: "From CASE to CIVE: A Future Challenge!", DASIA'96, Data Management Systems in Aerospace organised by EUROSPACE, Paris, May 20-24, 1996, Rome, Italy

[2] J.L. Terraillon: "The benefits of formal description techniques for space on-board systems and their integration in an on-board architecture", DASIA'97 (this conference), Data Systems in Aerospace, organised by EUROSPACE, Paris, May 26-29, 1997, Sevilla, Spain

[3] HRDMS (Highly Reliable DMS and Simulation), ESTEC contract no. 9882/92/NL/JG(SC), Final Report, Oct. 1994, Noordwijk, The Netherlands

[4] OMBSIM (On-Board Mangement System Behavioural Simulation, ESTEC contract no. 10430/93/NL/FM(SC), Final Report Nov. 1995, Noordwijk, The Netherlands

[5] DDV (DMS Design Validation), ESTEC contract no. 9558/91/NL/JG(SC), Final Report Dec. 1996, Noordwijk, The Netherlands

[6] R.Gerlich, V.Debus, Ch.Schaffer, Y.Tanurhan: EaSyVaDe: Early Validation of System Design by Behavioural Simulation, ESTEC 3rd Workshop on "Simulators for European Space Programmes" Noordwijk, November 15-17, 1994

[7] EaSySim II environment, Rainer Gerlich BSSE, Auf dem Ruhbuehl 181, D-88090 Immenstaad, Germany

[8] ITU, Recommendation Z.100, Specification and Description Language, SDL, 1989, Geneva. Blue Book, Vol. X.1, and appendices A, B, C, D, F1, F2, F3

[9] ObjectGEODE SDL-Tool, Verilog, 150 rue Vauquelin, F-31081 Toulouse Cedex, France

[10] TORNADO / WindView / VxWorks, WindRiver Systems, Inc. 1010 Atlantic Avenue, Alameda, CA 94501-1153, USA

[11a] Abrial, J.-R.: The B Book - Assigning Programs to Meanings. Cambridge University Press, August 1996.

[11b] Atelier B, version 2.0, STERIA DIGILOG, BP 16000, F-13791 Aix-en-Provence Cedex 3, France

[11c] B-Core [UK]: B-Tolkit User's Manual, version 3.2, Magdalen Centre, The Oxford Science Park (1996)

[12] J. Spivey: The Z Notation - A Reference Manual, Prentice Hall, 1989

[13] The RAISE Specification Language, The RAISE Language Group, Prentice Hall, 1992

[14] SAO+, Verilog, 150 rue Vauquelin, F-31081 Toulouse Cedex, France

[15a] J.K.Ousterhout, Tcl and the Tk Toolkit, New York 1994

[15b] Tcl/Tk Archive: ftp://wuarchive.wustl.edu/languages/tcl

[16] www-reference: java.sun.com