# Use of an Extended SDL Environment for Specification and Design of On-Board Operations

Rainer Gerlich

Dornier Satellitensysteme GmbH

RST16

D-88039 Friedrichshafen
Phone:  +49/7545/8-2124
Fax  +49/7545/8-5626
e-mail: gerlich@space-elec.dofn.de

Thomas Stingl, Yankin Tanurhan

FZI
Forschungszentrum Informatik
ESM
Haid-und-Neu-Strasse 10-14
D-76131 Karlsruhe
Phone: +49/721/9654-152
Fax: +49/721/9654-159
e-mail: stingl@fzi.de

Christoph Schaffer
Institute for System Sciences (CRC)
Johannes-Kepler-University
Altenbergstrasse 69
A-4040 Linz, Austria
Phone: +43/732/2468-9200
Fax: +43/732/2468-893

Frederic Teston, Giovanni Martinelli
WSC
ESTEC
P.O.Box 299
NL-2200 AG Noordwijk, The Netherlands
Phone: +31/1719/8-5088
Fax: +31/1719/8-4295

# Use of an Extended SDL Environment for Specification and Design of On-Board Operations

Rainer Gerlich

RST 16

Dornier Satellitensysteme GmbH

D-88039 Friedrichshafen, Germany

Phone +49/7545/8-2124
Fax +49/7545/8-5626
e-mail: gerlich@space-elec.dofn.de


Thomas Stingl, Y. Tanurhan

FZI

Forschungszentrum Informatik (FZI)
Haid-und-Neu-Strasse 10-14
D-76131 Karlsruhe
Germany
Phone +49/721/9654-152
Fax +49/721/9654-159
e-mail: stingl@fzi.de


Christoph Schaffer
Institute for System Sciences (CRC)
Johannes-Kepler-University
Altenbergstrasse 69
A-4040 Linz, Austria
Phone: +43/732/2468-9200
Fax: +43/732/2468-893


Frederic Teston, Giovanni Martinelli
ESTEC, WSC

P.O.Box 299
NL-2200 AG Noordwijk, The Netherlands
Phone: +31/1719/8-5088
Fax: +31/1719/8-4295

**Abstract:** At the beginning of system development system engineering has to ensure that the goals given by the customer will be met at the end of development. All system aspects have to be considered in a coherent manner. Properties and interfaces can be expressed in a manner human beings can understand in case of e.g. mechanical structures or electrical circuits. However, system operations do not have a physical shape. Hence it is difficult to understand what a system will really do before components are integrated and a system is in operation.

Therefore the goal of the OMBSIM study [1] was to define an approach (called EaSyVaDe: Early System Validation of Design) for early validation of system operations, i.e. validation at the beginning of a development lifecycle. System operations are identified by specification, represented by a design, and provided by hardware and software. Their services must be available right in time and they must correctly interact over time. Also, they must ensure that a system is dependable and remains it in case of non-nominal conditions.

A system's feedback is the "shape" which we need to assess whether a system fulfills a user's needs or not. Such a feedback must be provided in an understandable manner by <u>executable</u> hardware and software components in order to be able to prove a system's correctness and completeness for all such aspects during specification and design.

In the OMBSIM project it was decided to use SDL for implementation of executable behavioural models during specification and design. This allows to prove correctness of operations either by execution of operational scenarios or by automated analysis of state transitions. To cover performance and resource aspects and to introduce a system's architecture performance models are used in addition. As SDL is a formal language w.r.t. interfaces and behaviour tools can support the verification process.

The coherent use of SDL for specification and design allows a consistent transition from specification to design and from design to specification when refining a system's hierarchy. Moreover, a model represents a system property, not hardware or software. The hardware-software trade-off can be postponed to a later time. System development can be done more coherently. There is no need for separate lifecycles for hardware and software right from top level.

This development approach has been applied to an excercise and first results are available now. Mapping of the methodology onto the tool environment had to be defined. Decisions had to be made how the correct behaviour can be specified and how the architecture and hardware properties have to be introduced when making the transition from specification to design. Subcontracting can be eased if it is accepted that such models serve as contractual specifications and entities instead of informal textual requirements.

## 1. INTRODUCTION

Via operations a spacecraft provides the capability to a user to define and control the success of a mission. Its operations are implemented by hardware and software. For correct implementation different aspects like desired functionality, behaviour under nominal and non-nominal conditions, and consumption of timing and sizing resources have to be considered.

Implementation of larger systems must be managed by decomposition of a system into smaller components, which may be subcontracted. The development lifecycle starts with specification of system capabilities from a user's point of view and continues with design which describes fully the actually provided capabilities.

Currently, the hardware-software trade-off is done rather early causing separate development of hardware and software with final integration at the end. Specification is given in terms of textual requirements which do not allow to provide a feed-back on what they represent really. Also, software design concentrates on the functional aspects, leaving the proof for correct performance up to the final lifecycle phases.

The current development approach bears a reasonable risk not to meet the requirements or to have given wrong or incomplete requirements due to late validation. In the OMBSIM study [1] it was investigated how the current development approach can be improved. Main aspects were coherent consideration of functionality, behaviour and performance, hardware and software, and a coherent transition between specification and design and traversing through system hierarchy levels.

In the course of the OMBSIM study the methodology EaSyVaDe was defined and the supporting tool environment EaSySim was implemented. As no tools were available on commercial market which covered sufficiently functional and performance aspects together with aspects of verification and validation two tools were coupled. For behavioural and functional aspects the SDL-tool GEODE [2] from Verilog is used extended by the performance simulation tool SES/workbench [3]. Add-on software had to be developed to complement the tools by additonal validation capabilities and to provide tool coupling [4]. By a pilot application the consistency of the methodology and its mapping onto the tool environment were investigated.

The methodology and the tool environment are introduced in chapter 2. In chapter 3 the pilot application is described and results are presented. Feedback from the pilot application is given in chapter 4. Chapter 5 summarizes the experience.

## 2. THE EASYVADE APPROACH

The approach deals with the methodology EaSyVaDe and the tool environment EaSySim supporting the methodology.

### 2.1 The Methodology EaSyVaDe

EaSyVaDe means "Early Validation of Design", i.e. the methodology aims to give an early proof that the system under development will meet the user needs later on. It guides an engineer how to do stepwise system validation during successive specification and design phases.

The cornerstones of EaSyVaDe are:
1. models are executed with scenarios right from the beginning
2. a feedback is provided on system operations by model execution
3. a system is hierarchically decomposed into sub-components
4. object-oriented principles are applied for definition of components
5. all system aspects are addressed:
    - functionality and behaviour
    - operations (in terms of user interaction)
    - interfaces between system components
    - performance and resources needs and consumption

- interference of functions over time (phasing)
- dependability
  in terms of
  -- formalisation of interfaces and behaviour over time and related verification
  -- consideration of nominal and non-nominal cases
     with powerful non-destructive analysis capabilities by simulation

6. formalisation of sub-contracting
   by means of formal models instead of informal textual requirements
7. unique consideration of hardware and software both expressed by executable models
8. automation of code generation and hardware manufacturing from design.

Steps 3 and 4 are compliant with the principal ideas of HOOD [5].

## 2.2 The Tool Environment EaSySim

The proof for correctness and consistency of specification and design shall be given by execution of models and operational scenarios, i.e. by means of simulation. Four principal aspects have to be considered:

1. functional and behavioural simulation

2. performance simulation

3. verification and validation

4. derivation of target code and VHDL code from the validated models.

No commerical tool is currently available which covers sufficiently all four aspects. Therefore candidate tools were evaluated and two tools were finally selected: GEODE/SDL for

- functional and behavioural verification by formal checks,

- validation of behaviour by execution of operational scenarios

and SES/workbench for

- validation of performance by execution of test scenarios and performance models.

The two commercial tools of EaSySim can still be operated in stand-alone mode. In this mode SDL/GEODE verifies interfaces and behaviour, SES/workbench allows to evaluate architectures and visualisation of system activities over time. The main starting point is modeling and verification with GEODE. After that complete system validation is performed in coupled mode.

| Verification Steps | |
|---|---|
| **SDL stand-alone** | as defined by engineer |
| **SDL extended** | as defined by engineer<br>+<br>preparation for coupling |
| **SDL coupled** | as defined by engineer<br>+<br>preparation for coupling<br>+<br>activation of coupling |

*Figure 2-1: EaSySim Operating Modes*

By simulation nominal and non-nominal cases can be executed and evaluated. Add-on software extends the functionality of the tools towards execution of operational scenarios, model reuse and implementation of tool coupling. Fig. 2-1 lists the operating modes of EaSySim as seen from SDL.

The SDL environment is the master environment from which the target system (target code or VHDL code) are derived. Specification and design models are expressed by SDL. Specification models are refined to Design Models by replacing an SDL block (process) by another SDL block. This procedure introduces Specification Models for the next lower level which are included in the block of the Design Model. The selected mapping process for the tool environment ensures stability of interfaces when refining blocks or changing their internals.

The architecture and timing and sizing budgets are represented by SES/workbench models. It is master of time. The SDL modelling enviroment can be modified or enhanced in the following manner when SES/workbench is attached:

- an SDL process may be replaced by a corresponding performance model,

- 4 -

- an SDL channel may be replaced by a performance model,

- an SDL task may be replaced by a performance model,

- an SDL timer may be mapped onto a delay in SES/workbench.

A user just has to specify the mapping between SDL entities and performance models. The SDL source code is prepared automatically for tool coupling by EaSySim. Routing of commands and data from SDL to the performance simulation environment and back to SDL is done automatically by the add-on software.

GEODE allows to generate code from SDL with interfaces to real-time operating systems. SES/workbench allows to generate behavioural VHDL. Recently, a step has been made by Dornier towards an automated transition from SDL to VHDL [6] from a system's point of view based on the experience from previous activities [7,8].

More information about EaSySim can be found in [4] and [9].

## 3.   THE PILOT APPLICATION



*Figure 4-1: CLUSTER DMS (High Level View)*
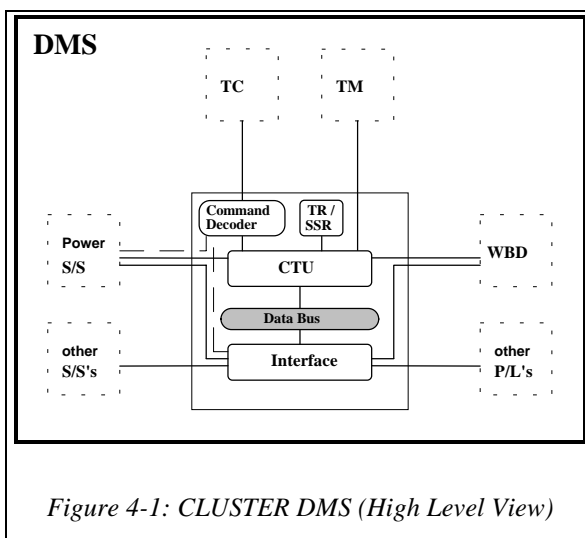
As pilot application the FDIR component of the CLUSTER Data Management System (DMS) was selected. Although this component represents only a small part of the CLUSTER on-board DMS, the complete DMS had to be modeled in a representative manner. This includes uplink and downlink with telecommand and telemetry units, power subsystem, all the other subsystems and payloads as complementing parts of the spacecraft, and the DMS itself consisting of command decoder, tape recorder, processor, OBDH bus and connected devices. Consequently, all principal on-board operations had to be considered.

Specification Models were estabished on top system level based on CLUSTER documentation. Then these models (SDL processes) were refined to next lower level level (subsystem level). For each SDL process a Finite State Machine (FSM) was implemented.



*Fig. 4-2: Generated MSC*
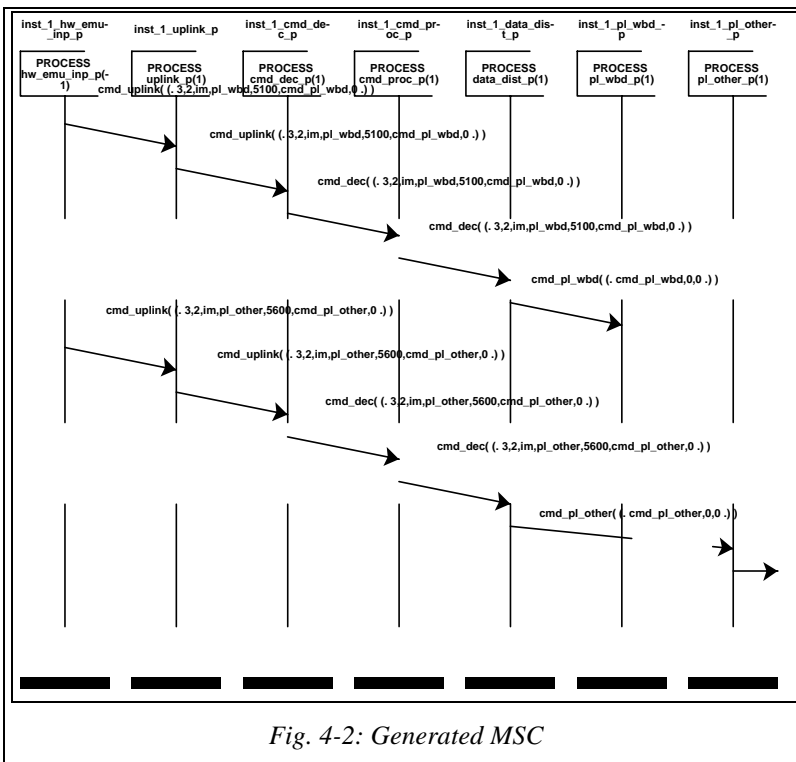
Message Sequence Charts (MSC) were defined for specification of the data and command flow.

By execution of the SDL models MSC's were generated. The specified MSC's were compared with the obtained MSC's to check for compliance of implementation with specification. Fig. 4-2 shows a sample MSC generated by execution of models. It describes the command flow from uplink via command decoder, command processing and distribution to payloads.

Several operational scenarios for initialisation, data acquisition and fault injection with reconfiguration were defined and fed into the SDL environment. After logical validation, i.e. validation of system behaviour w.r.t. functional, interface and operational

- 5 -

requirements, the performance models were attached by replacing the simple SDL channels by performance models of buses, processors and  devices.

This allowed to create timing diagrams of command and data flow through components like command decoder, data acquisition, data monitoring, buses and devices. Also, histograms and statistics were created on latencies of data acquisition, length of exchanged messages and queue lengths.
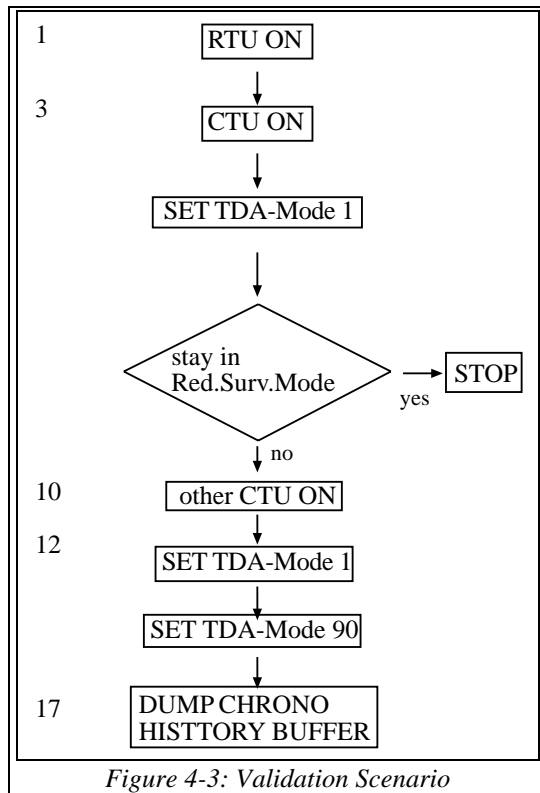
Correct behaviour of the models and their FSM's was verified by exhaustive simulation, a capability provided by GEODE. A feed-back on the actual implementation was possible by MSC's, timing diagrams, histograms and statistics. Faults were injected into the processor and the bus. Reconfiguration was initiated after a fault was detected and its correctness was verified by visualisation means.

Validation was done by execution of scenarios causing execution of system operations. They were derived from command flow charts like the one shown in Fig. 4-3. Such a flowchart gives a rough idea of what has to be done e.g. during DMS initialisation. For validation this sequence has to be more accurate because for modeling one needs exactly to know what to do. Therefore the command flow has been refined to a sequence shown in Fig. 4-4. This flow formed the input for the operational scenario fed into the system.

Visualisation of system operations is very important for validation. Timing diagrams are very powerful for visualisation. They inform on the state of the DMS components like power unit, command decoders (nominal and redundant one), RTU, the OBDH bus, CTU's, tape recorder and solid state recorder. Fig. 4-5  to 4-7 show what is happening in the system. For sake of good demonstration the time has been compressed from minutes to seconds for the non-interesting operational phases like "wait's". Therefore the timing of Fig. 4-4 is rescaled. The time units on the X-axis if figures 4-5 - 4-7 mean "milliseconds" Figures 4-5 - 4-7 have been generated by execution of SDL models and routing of the SDL signals to



Figure 4-3: Validation Scenario

1) Set RTU ON

    send PAPCMD: RTU_ON
       Priority 1 Mode 1/2 commands
2) wait 1 second (for RTU power on)
3) Set CTU ON

    send PAPCMD: CTU_ON
       Priority 1 Mode 1/2 commands
4) Wait 1 Minute
    (to complete CTU Bootstrap and Initialization run)

    send PAPCMD: TDA_MODE11
    send PAPCMD: FDIR_PAR
    send PAPCMD: ACQ_T_E
    send PAPCMD: MON_T_E
5) CTU now in Reduced Survival Mode, TDA-11

    no TM link available
6) Set TDA-Mode 1 to get TM contact

    send TDA_MODE1
7) wait 15 seconds to get TM (VC0 frames)
8) Verify VC0 data

    CDMU current
    RTU current
    CHB - number of entries = 2
    DH quality word

9) RSM (Reduced Survival Mode) is reached

    CTU Switch Over disabled
    CTU Reboot disabled
    RTU Switch Over disabled
10) If target mode is NM (Nominal Mode) then switch CTU, Priority 1, mode 1/2.

       If CTU A is ON: send command CTU B ON
       If CTU B is ON: send command CTU A ON
11) Wait 1 minute (to complete CTU Bootstrap and Initialization run)
12) CTU is now in Nominal Mode with TDA-11

    no TM link available
13) Set TDA-Mode 1 to get TM contact
14) wait 15 seconds to get TM (VC0 frames)
15) Verify VC0 data

    CDMU current
    RTU current
    CHB - number of entries = 4
    DH quality word
16) switch over and Reboot Condition settings

    CTU Switch Over disabled
    CTU Reboot disabled
    RTU Switch Over disabled
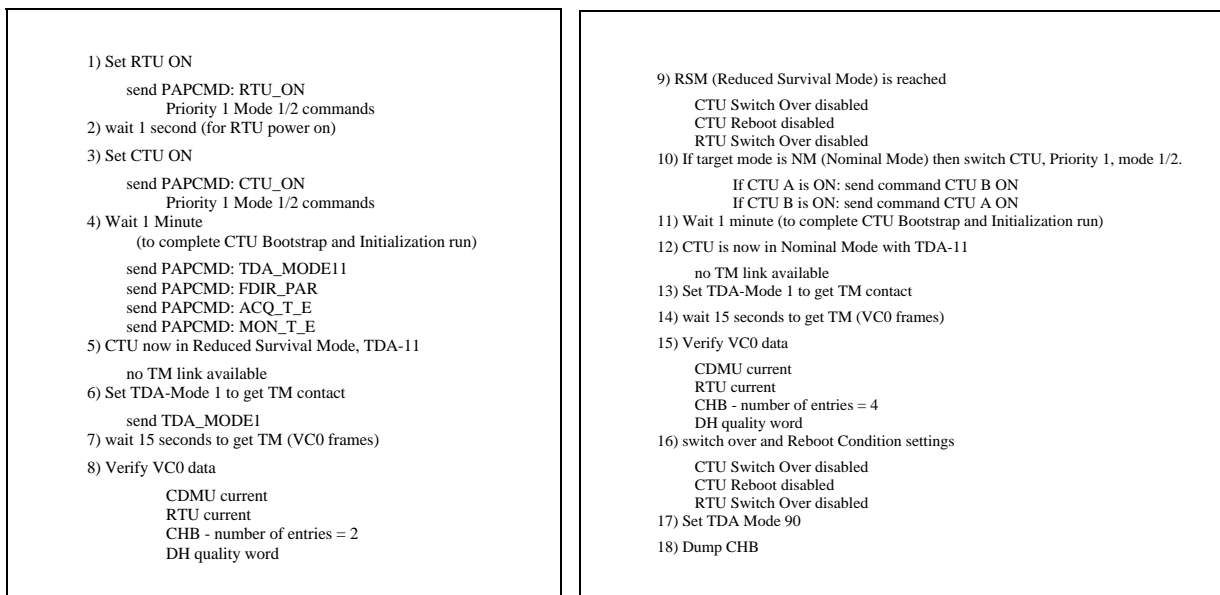17) Set TDA Mode 90
18) Dump CHB

Fig. 4-4: Refined Command Flow

performance models by which the activities are visualised.

Fig. 4-5 shows activation and deactivation of components and processing of the commands of step 4 of Fig. 4-4 on the active processor CTUA.

"Low level" of the status trace means that the component is deactivated, in case of "medium level" a component is powered, but not ready, "high level" means it is ready. Most of the components (except the tape recorder) switch automatically over into the ready state after a delay when powered. The occurence of commands on CTUA is visible by a peak caused by their processing time.



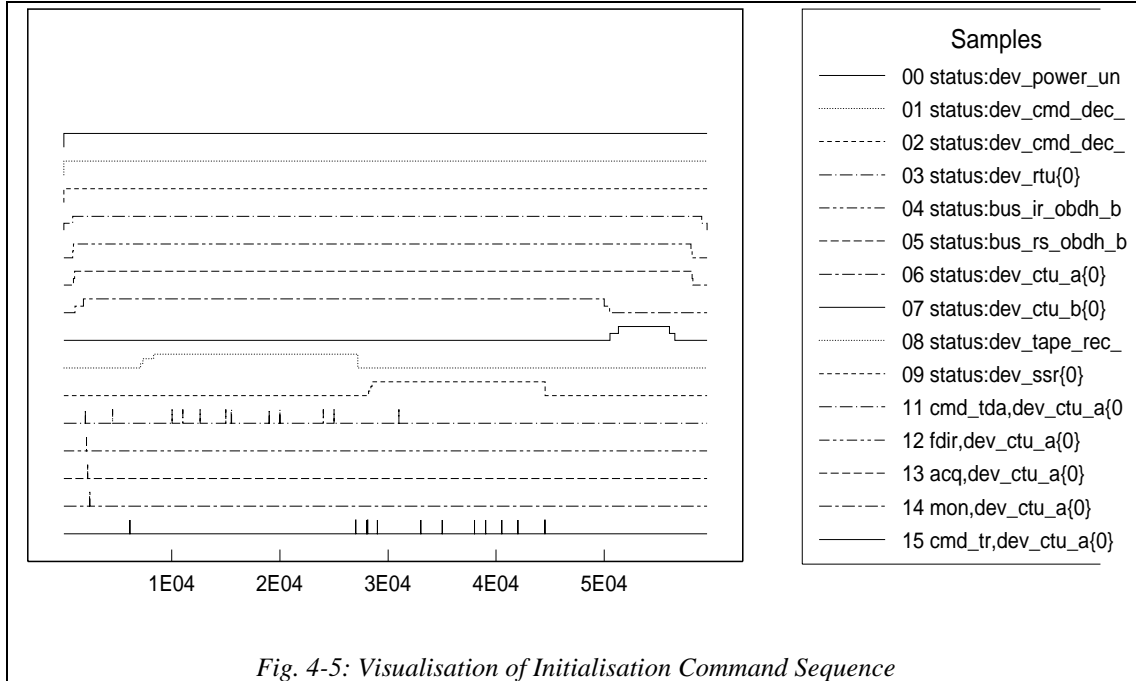*Fig. 4-5: Visualisation of Initialisation Command Sequence*

Fig. 4-6 shows details of tape recorder operations. The top trace (00) represents its status. Four states are indicated by the trace. "zero" means it is depowered. "1/3" level means it is powered. In powered mode it receives a "reassign" command which causes an immediate switch to "2/3" level. After some delay it automatically switches in stand-by mode indicated by "highest" level.



*Fig. 4-6: Tape Recorder Commanding*
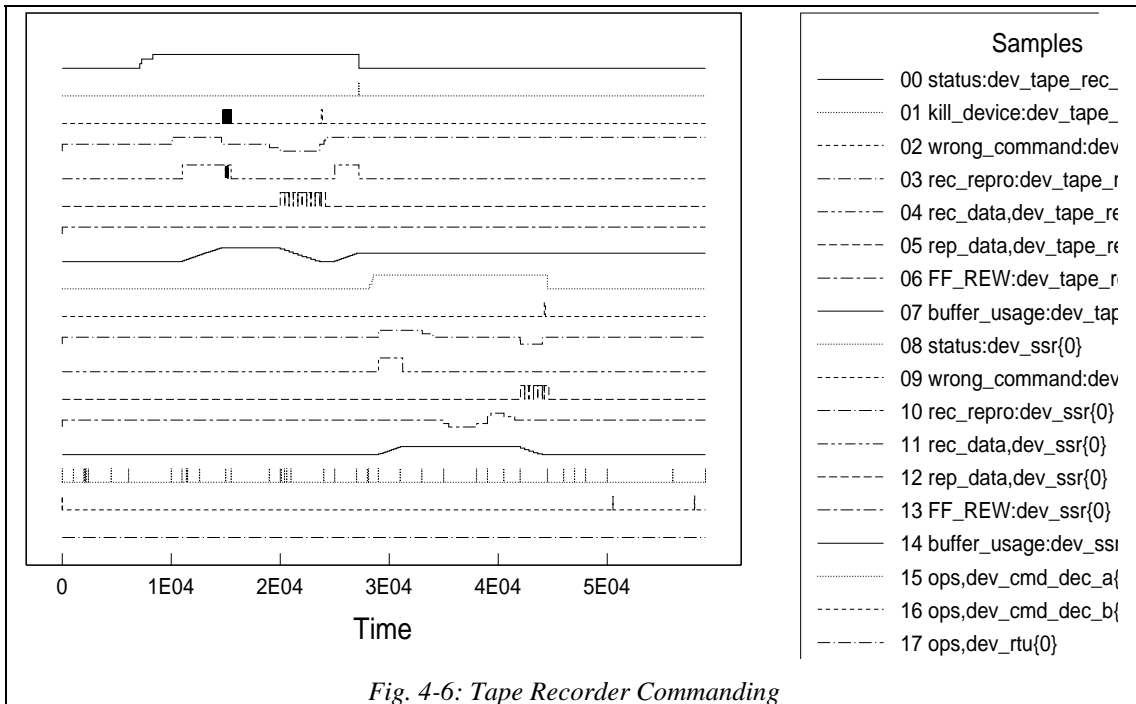
The next two traces (01, 02) show anomalies like (desired) fault injection or (unexpected) wrong commands, i.e. commands which are not compatible with the current state.

The fourth (03) trace shows the status of recording and retrieval (reproduction) of data. Medium level means stand-by mode, in high level data are recorded and in low level data are retrieved. When the tape recorder receives a "recording" or "reproduction" command it switches in an intermediate state before it automatically reaches the final state after some delay. Such parameters as delay can be specified by data file for each component.

The two next lower traces (04, 05) indicate data recording or reproduction. Trace 07 shows the amount of data ("buffer_usage") already stored. At the time for which wrong commands occur the first time (see trace 02) the storage capacity (see 07) is exhausted and the tape recorder switches back into stand-by mode (see 03). But in stand-by mode no data recording is allowed. Therefore the following recording commands are wrong commands(see 02).

Later on, the recorder is switched into reproduce mode and data are read: the amount of data decreases (see 07). Now, data are read until the storage is exhausted: the tape recorder switches automatically after some delay back to stand-by mode (see 03). Therefore the following reproduce command are illegal and marked as erroneous (see events in 02). Some time later again it is switched into recording mode and the amount of data increases again (see 07). At about t=29000 ms a fault is injected (see second trace 01) which causes the tape recorder to fail completely ("kill device") (see trace 00). The state switches down to "power off".

Therefore the redundant unit "SSR" (Solid State Recorder") is powered  and switched to stand-by mode (08). Again data are recorded (see 11) and the data storage gets data (see 14).

Trace 13 is an indicator for fast forward and rewind of the SSR. The medium level again indicates the stand-by mode. The recorder receives first a Rewind and then a Fast-Forward command (see 13) before data are retrieved (see 12). Again, a wrong command appears (see 09) because the storage is exhausted.

The lower traces show processing of commands in the command decoder (traces 15, 16 and 17).

An important feedback on recorder operations was that mostly it was forgotten to switch into stand-by mode before a new mode command was issued. Then the "wrong command" traces indicated that a command and the actual state were not compatible.

Each of such traces is also available in a more detailed manner, i.e. there are traces for which the name is expanded by additional information like command type, source and destination of the command. So a lot of information on propagation of data and commands can be obtained and visualised.



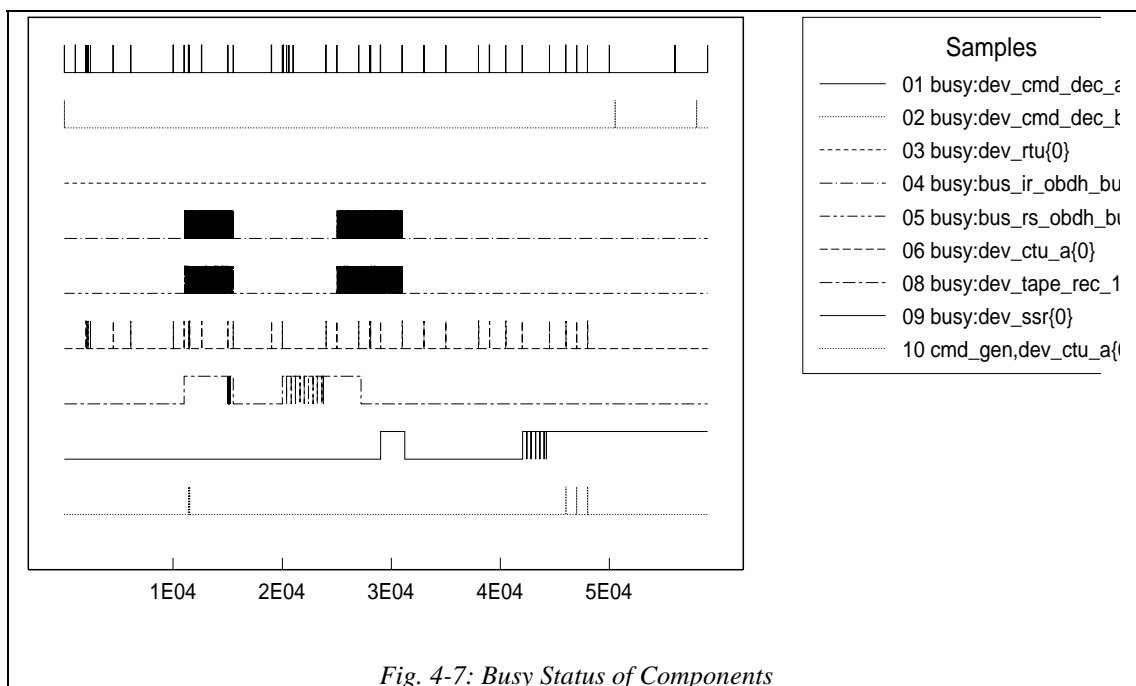*Fig. 4-7: Busy Status of Components*

- 8 -

Fig. 4-7 shows the busy state of components like command decoder, OBDH bus (interrogation and response bus), CTU's and tape recorder. The RTU is inactive altough data are flowing through the bus. Obviously this is a fault, probably a modelling bug. Hence, modelling faults are also visualised.

Above examples give an impression on the visualisation power of EaSySim. The traces can be correlated arbitrarily in a figure and can be inserted into several figures. This allows to analyse correlation of events or to detect missing or wrong correlation. By such patterns it is much easier to detect functional and timing inconsistencies in a system wrong commanding sequences (scenarios).

The simulation environment allows to analyse more parameters and system properties than would be possible in a real system.

## 4.    FEEDBACK

Implementation of the pilot application gave a first feedback on modelling issues, use of the tool environment and project management.

### 4.1    Modelling Issues

#### 4.1.1    Abstraction from Hardware and Software

Some properties of a system like "to be powered" are mostly considered as a matter of architecture and design. As a system or one of its components can only be powered when it is mapped to a physical component one could believe "powering" is a matter of design or of representation of hardware.

However, the property of a component to provide a service or not does not depend really on a physical property. It is an abstract system property which should already be considered during specification. For validation of a specification and its related scenario it is essential to consider the state of a component. If not specified that it must be activated it may mean it should be continusly be powered or the contractor may omit the activation command even he should not do it.

Hence, for the correctness of a specification consideration of activation or de-activation is essential. Also, this eases the transition from specification to design. If such a capability is already introduced during specification there is no potential conflict concerning the system's structure in design. That part representing the state can just be mapped onto an architectural item.

The difference between specification and design is the correlation with time. During specification each component is independent concerning its activation or de-activation. Having introduced an architecture a dependency is introduced when different system components are mapped e.g. onto the same chip. Then their activation and de-activation is depending on the power supply of that chip. If components are on the same chip they are powered or they fail simultaneously. If not, they are independent and may fail at different instants.

Similarly, it is with faults and fault injection. A fault analysis during specification considers only <u>that</u> a fault can occur. Multiple faults can also be considered, of course. So the complete error scenario can already be executed. After design the faults are correlated due to the architecture. Now the probability for combinations of faults may be weighted by probabilities depending on the architecture.

To summarize: to avoid restructuring of a system decomposition and to enforce consideration of states and faults already during specification, a system component shall be represented by the following three principal parts which shall be separated from each other:
1.    an operational component
       which provides the desired functionality and performance,
       e.g. the capability to transmit data in case of a bus
2.    a management component
       which represents the inherent physical state of a component not visible from an operational point of view and possibly separated by architecture from the operational component,
       e.g. power circuits which impact the operational component (activation and de-activation)
3.    an environment or simulation component
       which represents the environmental impacts or test management during validation by simulation,
       e.g. the ability for fault injection or restart of simulation.

### 4.1.2 Allocation of Management and Simulation Components

In case a model shall be translated to hardware, management of the state (active, not active etc.) of a component is no longer needed because it is directly driven by the hardware properties. When powered, the component is alive, if not it does not respond. So the management and environmental components can/must be dropped. If software is derived we would still need the part for management of a component.

Consequently, we have to care about some parts which may be removed later on together with their communication channels. In order to reduce the effo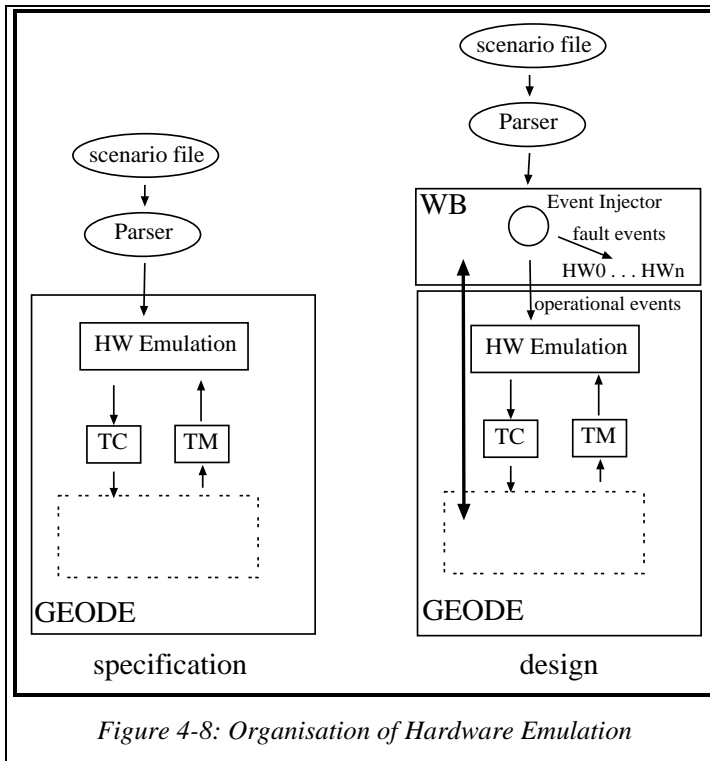rt needed to remove the additional parts and channels, the operational channels shall be used for communication with such parts. This means that all such components have to be placed at a central location from which the related operational components can be accesed.

This location is the interface with ground in case of an on-board DMS because all components will directly or indirectly receive commands via TC from ground or deliver data via TM to ground. Fig. 4-8 shows the solution as used for the pilot application for stand-alone and coupled mode.

In stand-alone mode the "HW Emulation" component manages itself the states. In coupled mode such states are managed by the performance models which are accessed from "HW Emulation".



*Figure 4-8: Organisation of Hardware Emulation*

| model type | instances |
|---|---|
| device | ctu_a, ctu_b, cmd_dec_a, cmd_dec_b |
| taperecorder | tape_rec1, ssr |
| obdh bus | ir_obdh_bus, rs_obdh_bus |

*Fig. 4-9: SES/WB models used in the PAP*

### 4.1.3 Reduction of Modelling Effort

A lot of effort was saved due to use of model types in SES/workbench. In fact, only three model types were needed to represent all the DMS components from a performance point of view. Fig. 4-9 shows the correlation between model types and derived instances. The advantage of using types is that only at one place - in the type - changes have to be made, if needed.

The operational part of type "device" causes a delay according to the length of incoming data. The bus type is similar but the delay depends on the bus data rate.

Furthermore, one needs to monitor other properties or to generate other timing diagrams and histograms compared to the device type. Therefore another model type was derived for the bus from the device type.

The tape recorder type needs more intelligence for proper and realistic operation. Therefore an own type had to be created based on the templates taken from the device type.

In SDL, process instances can be derived from a process type.

## 4.2 Lessons Learned

CLUSTER was selected as pilot application because it was believed that an already existing DMS design would allow a comparison of the new development approach with the conventional one. Specification should start with the CLUSTER DMS specifications documents only. However, it was recognized rather soon that only few detailed information on the FDIR component was included in CLUSTER specifications. Mainly, the FDIR requirements were generic. Therefore the already available design documentation had to be considered in addition.

However, the final design documentation was already too detailed and a lot of effort was needed to extract that information which is adequate to enhance the CLUSTER specification of the FDIR component on top level. This shows that in the conventional developement approach at least incomplete information is provided for a specification, which is enhanced during the following phases by the contractor by interpretation of the general (generic) requirements. Adding of more precise behavioural requirements is contractually based on the general requirements, but may be ambigious. It also means that tasks are shifted from earlier development phases to later ones. And the same happens for the risk because transformation of the possibly incomplete and/or inconsistent generic requirements into detailed ones may cause conflicts with already existing requirements.

It was also recognized that more add-on software than planned was needed for better support of feeding in operational scenarios. Therefore this functionality was added to EaSySim, when implementing the pilot application.

As explained above some more effort was needed for finding of the right textual requirements as starting point for formalisation. Also, additional effort was required to find the right mapping of EaSyVaDe to EaSySim. But this is quite normal in case of a pilot application. This caused that the procedure for refinement from system level to subsystem level and from specifcation to design could not be applied in the planned manner due to resource and time limitations in the project. Therefore after definition of specification models on system level immediately the subsystem level models were derived and the architecture was introduced. Logical validation of specification and complete validation of design were done on subsystem level.

It is now planned to apply the full procedure to the telecommand and telemetry component to investigate and to demonstrate the efficiency of the EaSyVaDe approach and of the EaSySim tool environment having now available the required guidelines.

## 4.3    Impacts on Project Management

The applied methodology uses executable, formal models for specification and design. They allow to express customer requests  and contractor response unambigously. Implementation of models requires more effort during specification and design phase, but it saves time in later phases. It implies definition and execution of acceptance tests. Therefore the later lifecycle phases will be shorter, especially when code is generated automatically from the validated design models. Effort is shifted from later phases to earlier phases with a decrease of the total effort.

However, project management has to accept that

1.    more effort and time has to be planned for the earlier phases,

2.    executable models instead of informal, ambigous textual requirements are a contractual matter.

## 5.    CONCLUSIONS

By the pilot application it has been shown that and how the methodology and the tool environment can be used. Some iterations were needed to find the best way to implement a hierarchical object-oriented approach for the system components and to combine both tools. The current GEODE version is based on SDL88 which provides some object-oriented features like Abstract Data Types. It is expected that the next version ObjectGEODE which implements SDL92 will better fit with the object-oriented ideas of the methodology.

An essential advantage of modeling and simulation is the visualisation of operational processes by MSC's, timing diagrams, histograms, statistics and debuggers and the analysis of behaviour based on state transitions. Experience is that a lot of indicators for faults exist which are provided by the visualisation menas. So it is easy to detect occurence of a fault. But for identification of the fault one needs more details in most cases which means that the internals of the performance models may have to be enhanced continously with increasing quality of models (less bugs) and increasing complexity of remaining bugs.

For efficient modelling appropriate techniques have to be identified and applied. Especially, design for reuse is essential, but not so easy to achieve. Therefore one has carefully to consider modelling progress and to introduce better techniques, when needed. By the activities performed for OMBSIM and the previous HRDMS project [10] sufficient experience on this subject is available now.

Moreover, it was identified that activation - deactivation of a component is not a matter of the architecure only, but also a matter of design. This has to be applied consequently.

## REFERENCES

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

[2] GEODE SDL-Tool, Verilog, *150 rue Vauquelin, F-31081 Toulouse Cedex, France*

[3] SES/workbench, *Scientific and Engineering Software Inc., Building A, 4301 Westbank Drive, Austin, Texas, 78746-6564, USA*

[4] OMBSIM (On-Board Mangement System Behavioural Simulation, EaSyAdd Software User Manual, *ESTEC contract no. 10430/93/NL/FM(SC), Noordwijk, The Netherlands*

[5] HOOD, Reference Manual, *Masson, Paris, Prentice Hall, London, 1993*

[6] M. Lehmann, master thesis, finalisation in September 1995, Dornier / FH Albstadt-Sigmaringen

[7] W. Glunz, G. Venzl, Using SDL for Hardware Design, *Fifth SDL Forum, Glasgow, 1991, Elsevier Publishers*

   B.Lutter, W.Glunz, F.J.Ramming, Using VHDL for Simulation of SDL Specifications, *Proceedings of the EURO-DAC/EURO-VHDL 92, Hamburg, Germany, Sept. 1992*

   W.Glunz, Th.Kruse, T.Roessel, D.Monjau, Integrating SDL and VHDL for System Level Hardware Design, *Proceedings of the CHDL, Ottawa, Canada, April 1993*

   W.Glunz,Hardware-Entwurf auf abstrakten Ebenen unter Verwendung von Methoden aus dem Software-Entwurf, Dissertation, *Paderborn/Muenchen, April 1994*

[8] K. Kronloeff, Method Integration - Concepts and Case Studies, *Wiley Professional Computing Series*

[9] 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*

[10] HRDMS (Highly Reliable DMS and Simulation), ESTEC contract no. 9882/92/NL/JG(SC), Final Report 1993, Noordwijk, The Netherlands