EaSyVaDe

Early Validation of System Design by Behavioural Simulation

ESTEC 3rd Workshop on

"Simulators for European Space Programmes"

Noordwijk

November 15-17, 1994

Rainer Gerlich, Volker Debus

Ch. Schaffer
Institute for System Sciences (CRC)
Johannes-Kepler-University
Altenbergstrasse 69
A-4040 Linz, Austria

Y. Tanurhan
Forschungszentrum Informatik (FZI)
University of Karlsruhe
Haid-und-Neu-Strasse 10-14
D-76131 Karlsruhe, Germany

EaSyVaDe: Validation of System Design by Behavioural Simulation

Rainer Gerlich, Christoph Schaffer[1], Volker Debus, Yankin Tanurhan[2]

[1] Institute for System Sciences (CRC)
Johannes-Kepler-University
Altenbergstrasse 69
A-4040 Linz, Austria

[2] Forschungszentrum Informatik (FZI)
University of Karlsruhe
Haid-und-Neu-Strasse 10-14
D-76131 Karlsruhe, Germany

Abstract: The EaSyVaDe methodology aims to improve the development process for embedded systems like a Data Management System on-board of a spacecraft. Currently a rather long time is needed until a system is ready for operation. Future embedded systems will have to provide even more capabilities. This will make system development more complex and will increase effort and time. So we need to improve the development process to have systems sooner available at lower costs. The EaSyVaDe approach allows to validate system specifications and designs continously during the development process, i.e. already from top level specification until final integration. This gives the engineers the opportunity to prove for each development phase if the goals will be met. The transition between development phases and information exchange between development teams will be simplified. The amount of paper will be reduced. Formal specifications and representations of design instead of informal text are used from which the implementation can be derived. EaSyVaDe uses models and simulation for early validation. The feedback provided by simulation makes visible whether the modeled system behaves as expected or not and if all its components have been defined correctly and consistently. EaSySim, the corresponding tool environment, provides what is needed to apply the methodology. We think that by the presented methodology the system development process can be significantly improved. To demonstrate this we are going to apply EaSyVaDe to an example. The EASyVaDe approach was defined by Dornier, University of Linz and FZI under the still running ESTEC study OMBSIM.

Keywords: Validation, simulation, behaviour, real-time, lifecycle, performance, tool coupling.

1 INTRODUCTION

Embedded systems - on-board space applications like a Data Management System (DMS) belong to this type - have to react in real-time to events coming either from the environment or from the system itself. This type of system tends to be very complex and it will become more complex in future. The possibility to forget an essential system parameter in the specification is very high. Mostly specification and design faults are detected very late in the design process and in the worst case we will detect them when already operating the real system. Also, system validation can currently only be done at the end of system development, as before one is not able to completely evaluate what a system really does.

Software lifecycles like the spiral model [1] try to minimize that risk by using prototypes in the early phases of the life cycle. Prototypes are used to "learn" more about the system and its environment. They reduce the risk to establish wrong system specification and design.

EaSyVaDe adopts this approach for the system lifecycle but tries to systematize the design of prototypes which is necessarey because prototypes are very often a synonym for quick and dirty solutions and therefore again include many risks. In EaSyVaDe prototyping is done by so-called Specification Models. These models reflect the specification and are used to make the specification alive. It is a well known problem that informal textual representations of a specification are hard to comprehend, especially for complex systems.

The objectives of EaSyVaDe and EaSySim are:

1. to use an executable specification
   which is achieved by formalisation of the textual specification and building of models,

2. to achieve a consistent and complete specification
by checking the interfaces and the real-time behaviour
e.g. by use of an SDL tool like GEODE [2],

3. to introduce an architecture and to check whether the related design fulfills the performance and resource requirements
e.g. by a performance simulation tool like SES/workbench [3],

4. to achieve a design which fulfills real-time behaviour and performance / resource requirements
e.g. by coupling of both tools,

5. to use models as compact specification
which can be exchanged between customer and subcontractor instead of ambigious requirements on paper,

6. to automate the transition between the development phases,

7. to avoid a separate implementation manually derived from the models.
Automated generation of software will be supported by EaSySim. Derivation of hardware is a future issue.

At system's level we have to define so-called Specification Models which allow us to verify and validate a system specification. These Specification Models are executable and can be animated. Such a "living" system is easier to comprehend than a static, "dead" one. Animation allows the designer to make several different experiments which will speed up the learning process significantly. Having a better picture of what the system has to do, will also increase the quality of the design.

To achieve a complete and consistent specification and design a formal language has to be used. EaSySim uses the GEODE tool which is based on SDL (CCITT'88 standard). Therefore lexical and syntactical faults as well as dead locks etc. can be identified.

With introduction of an architecure non-functional requirements like performance or resource requirements have to be considered. Performance simulation is used to verify that these requirements can be met with the selected design. For this purpose EaSySim provides SES/workbench.

The next goal is to verify that the design fulfills <u>all</u> requirements. Therefore both tools have to run together to cover all aspects. Beside the definition of performance models only very small user intervention is necessary to run an SDL model in the coupled simulation mode of EaSySim. Both tools still can be used in stand-alone mode.

If a component will be subcontracted a Specification Model will be passed to a subcontractor. He can use a Specification Model as a prototype. Hence possible misinterpretations, frequently occuring when passing textual requirements, are minimized. Moreover, a subcontractor does not have to build his own new prototype.

Specification Models are used as a starting point for systems design. Design Models are derived from Specification Models. This speeds up the design process. The interface of the Design Model is identical with the interface of the Specification Model. Refinement of the Design Model will lead to Specification Models again on next lower level[1]. Therefore it is necessary to model the Specification Models in the same environment as doing systems design.

Having finished the design it is possible to generate target code (C, ADA) automatically. This will avoid erroneous and time consuming coding. Automated manufacturing of target harware is a future issue.

System specification and design are following each other layer by layer. This is different to what is done mostly up to now, e.g. in the "V"-approach [4,5]. There one has to perform requirements analysis completely from top to bottom before one starts the top-down design. EaSyVaDe considers that a specification (below

[1]We distinguish between "level" and "layer". A "level" is related to a hierarchy of decomposition into e.g. objects, operations, components. For an item refering to another item the following definition is made: when an item on level n refers to another item, this item is on level n+1. A "layer" also means a vertical decomposition of a system. But a layer may contain more than one "level". So a layer groups levels in each branch of a system hierarchy to a selfstanding larger unit of decomposition.

top layer) depends on the previous design. Therefore requirements analysis on a lower layer shall not be done before the design on next higher layer is available.

In chapter 2 we will describe our validation strategy and will present which means we will use for evaluation of specification and design. Chapter 3 explains what EaSyVaDe is. The toolset EaSySim and its relationship to EaSyVaDe are described in chapter 4. In chapter 5 we discuss future issues concerning the methodology. By an example we show in chapter 6 how EaSyVaDe can be applied. Chapter 7 concludes on what has been achieved up to now.

2. VALIDATION STRATEGY

Validation means to evaluate that the right system was built according to user requirements [6]. Early validation means to evaluate that the right system will be built. Verification means to confirm that the system was built right according to its specification [6]. Consequently, in context of Early Validation, firstly requirements have to be validated by simulation, as they specify a system and therefore will influence if we will build the right system. Secondly, we have to verify the design to prove that we build the system right and that it is in accordance with the specification. Finally, we have to (preliminary) validate the design to prove that it represents the right system.

Verification can be done for each design component of a system. However, validation can only be done at the end of the development process, because only then the real behaviour and performance can be observed (see Fig. 1). If we did not build the right system we have a big problem. So we have to minimise this risk by "Early Validation".

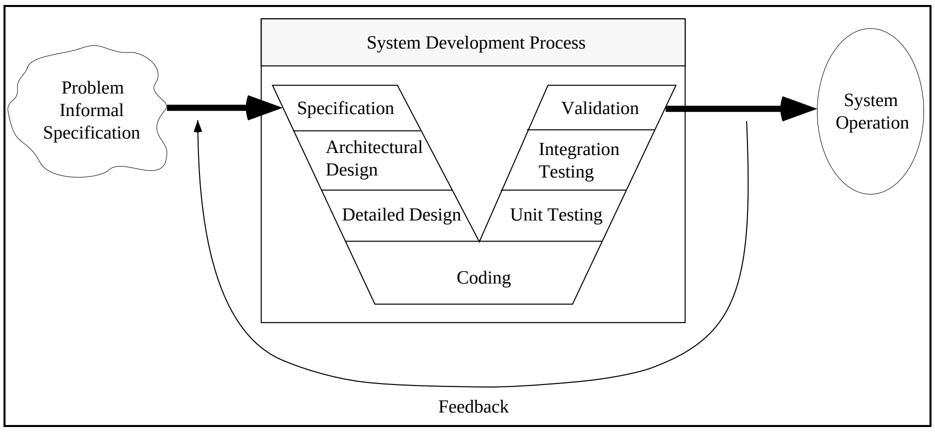


Figure 1: System Development with Late Validation

Early Validation means that validation is done successively for each layer of a hierarchically decomposed system (see Fig. 2). The term "early" indicates that validation can only be done preliminary. When a specification is derived from a design, the design will be validated based on the assumption that the lower layers will fulfill the requirements imposed on them.

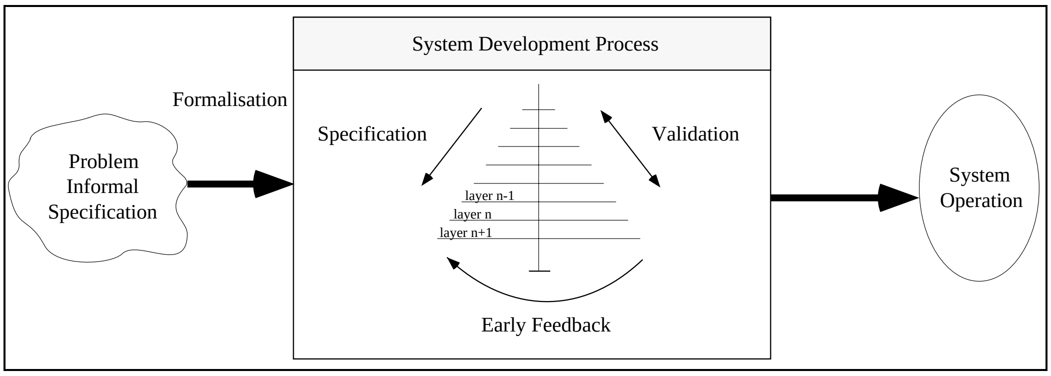
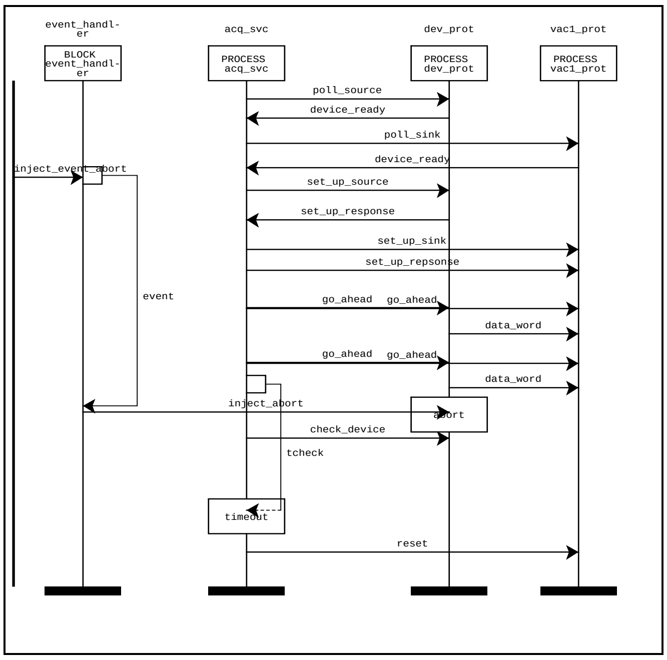


Figure 2: System Development with Early Validation

We are applying behavioural and performance simulation. Therefore the risk is significantly lower not to meet the goals, because by animation and visualisation we better comprehend a system and can earlier detect specification and design faults. E.g. on a higher layer we establish and check simulation performance. We get knowledge of what is requested and we can assess if this is feasible or not.

The most crucial point is not to know what is requested rather than not to be able to provide it. This means when we know what to provide, mostly we can provide it. Similarly, it is for behaviour. When we see by animation or visualisation what we have specified then we can immediately decide if the system we are going to build is the right one or not.



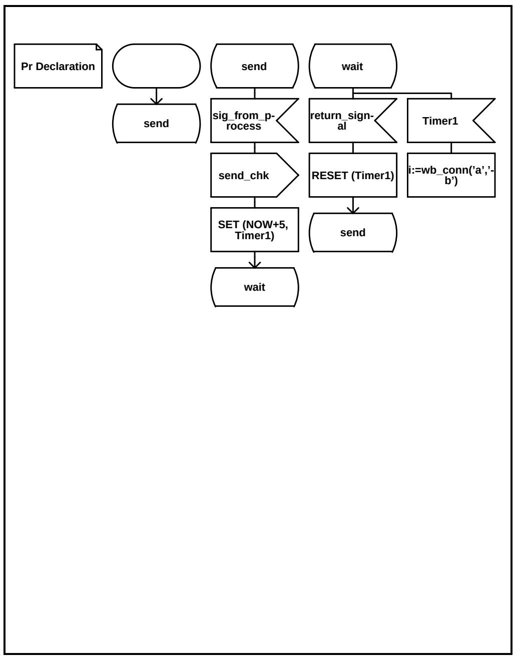


Figure 3: MSC for a Reconfiguration Scenario

Figure 4: Sample FSM with Timeout

We apply four major steps by which Early Validation of a system's behaviour and performance is achieved:

1. specification of behaviour
   by describing the data flow and message exchange between system components, e.g. by Message Sequence Charts (MSC) in SDL. Informal specification of performance and resource requirements are still expressed by text.
2. implementation of behaviour
   by SDL models and associated Finite State Machines (FSM),
3. consideration of performance and resources when introducing architecture and design,
4. analysis of behaviour by visualisation of a system's behaviour and performance
   e.g. by post-processing of results of model execution with tools.

2.1 Specification of Behaviour

Validation of a system can be done only in context with its environment. Message Sequence Charts (MSC) of SDL are used to show up the interaction between system components and its environment. MSC's are specifying the expected output of a component as reaction on a certain event. Fig. 3 shows a sample MSC for reconfiguration of a device as reaction of an identified fault. In fact, MSC's are used to define test scenarios for components' behaviour.

To ensure that the "system" and its "environment" can be simulated on lower layers we have to pass those parts of the actual system and its environment, which are needed to validate the system. Also, related test scenarios have to be transferred.

2.2 Implementation of System Behaviour

With EaSySim behaviour of a system is defined by extended Finite State Machines (FSM). States and state transitions can be defined either in graphical or in textual representation. SDL supports FSM's and EaSySim/GEODE will check consistency and completeness of the FSM's. Animation of system's behaviour allows to step through the sequences of states and state transitions and to see what happens under which conditions. The scenarios can be recorded and played back. The observed message sequences can be compared with the expected ones.

Fig. 4 shows a sample FSM for data acquisition with specification of a timeout (the FSM does not correspond to the MSC of Fig. 3).

Each component of a system will include a FSM, so that the overall behaviour will be composed out of small FSM's. The granularity of the components will control the complexity of the FSM's. The smaller the components, the less complex the FSM's and the better the behaviour can be defined and evaluated.

2.3 Consideration of Performance and Resources

When we introduce an architecture we have to care about performance and resources which are considered with SES/workbench. We can do performance analysis independently from behavioural analysis or both together.

In the combined mode either the specific SES functions, so-called "nodes", needed for performance modelling can be used by SDL-models, or an SDL model can be replaced by an SES performance model or several ones. Also, test scenarios are needed when we do performance analysis without using SDL models.

In stand-alone mode we can evaluate performance and usage of resources independently from the SDL models. Then we also have to model the architecture and to define performance test scenarios.

The system enginieer has to decide if the combined mode is sufficient or if the stand-alone performance mode is needed in addition.

2.4 Analysis of System Behaviour and Performance

The GEODE tool can compare the expected MSC's with the observed ones. This allows to detect deviation from expected (real-time) behaviour. By EaSySim we also provide timing diagrams, histograms and statistics to analyse behaviour and performance of a system and usage of its resources.

By timing diagrams we can observe propagation of events or data in a system. The example of Fig. 5 shows propagation of data in a data network. In Fig. 6 the size over time of a data buffer is displayed. Histograms visualise distributions, e.g. usage of resources, latencies of data acquisition. In Fig. 7 a histogram gives information on usage of bus slots.

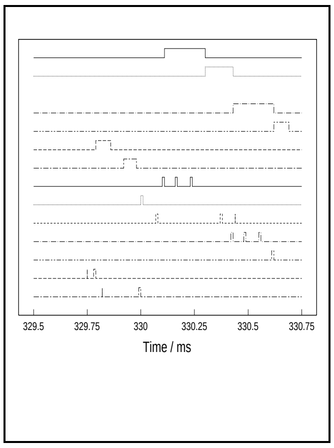


Fig. 5: Data Flow Timing Diagram

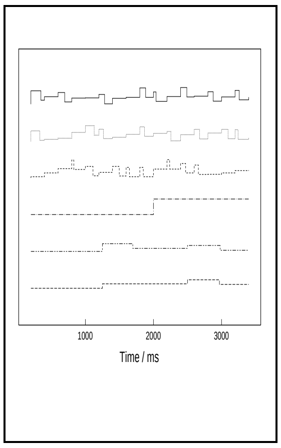


Fig. 6: Data Buffer Monitoring

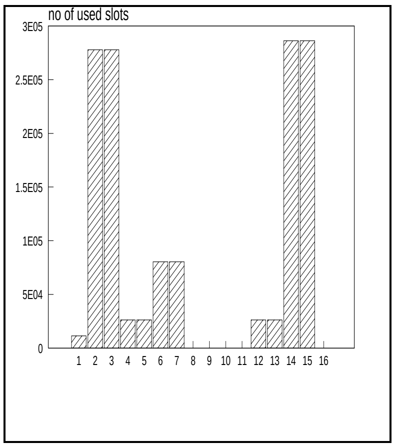


Fig. 7: Histogram on Resource Usage

State transitions can also be visualised by timing diagrams. This allows evaluation of logic flow in the system. By use of several traces correlation between state transitions and events is possible.

These examples show how by simulation and appropriate visualisation tools weakness or faults in specification or design can be early detected before end of system development.

Above means are used to verify and validate specification and design for a system's behaviour and performance.

3. THE EASYVADE METHODOLOGY

The next sections explain the details of EaSyVaDe:

- hierarchical validation
  describes how Early Validation is applied within a system's hierarchy,
- model refinement
  explains the transition between specification and design using models
- simulation and validation
  relates types of requirements as defined by [6] to simulation and validation types.

3.1 Hierarchical Validation

The EaSyVaDe methodology decomposes a system hierarchically into layers. The overall system lifecycle is decomposed into smaller lifecycles for each layer. Early Validation is done for each such layer. After succesful design validation one continues on next layer.

EaSyVaDe starts with top level informal, textual requirements, indicated by the "i"[2] in Fig. 8[3]. From the textual requirements functions are identified. Similar functions, e.g. all functions performing data handling, will be collected into objects together with the data they need.[4] The functions just provide what is needed on top level. We do not refine the functions, as we do not want to go into depth. E.g. for a monitoring function we do not implement the exact mechanism for data acquisition, but we may just provide random data on this top layer. Our point of view is similar to that on an onion: we just see the top shell and limit the scope to this layer.

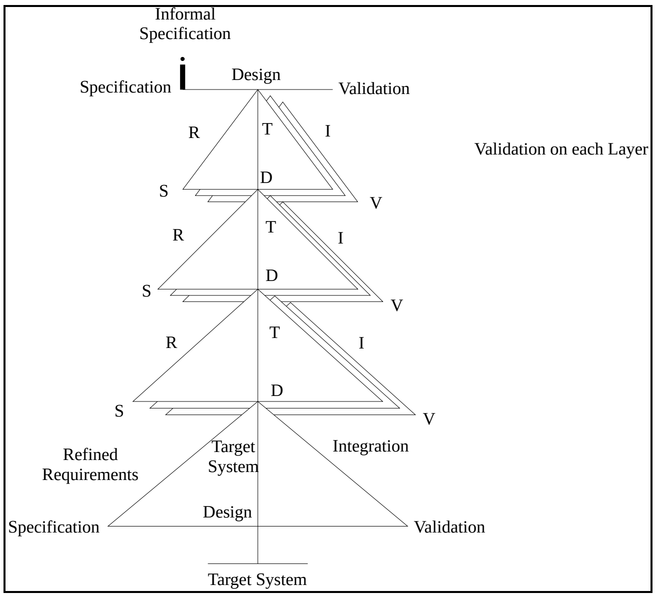


Figure 8: Hierarchical Validation

Having identified the objects we formalise the textual requirements and define executable models (Specification Models) including the identified functions. We also define test scenarios, e.g. by MSC's, which we need for Early Validation. The models represent the requested real-time behaviour and performance and

[2]like a candle

[3]As the figure looks like a tree, the hierarchical system decomposition with iteration between specification and design is called the "T"-procedure.

[4]Also, one may start with data, if desired. Then data will be collected and put into objects together with the functions needed to process the data. So the procedure for object identification is symmetrical concerning data and functions.

shall also consider resource requirements and constraints. After this step we arrive at the bottom of the "i". We will have an executable formal specification for the top layer of a system (on lower layers the formal specification is indicated by an "S" in Fig. 8).

Fig. 9 shows the steps which have to be executed when one proceeds from specification design phase. The related procedure is called the "SDV"-procedure as it describes how to come from "S" (specification) via "D" (design) to "V" (Early Validation).

Firstly, we execute the Specification Models using the test scenarios- Then we check if the models show the expected behaviour and performance. If not, we have to change the models and to repeat the tests. After succesful simulation we have achieved validation of the formal specification for behaviour and we have specified the performance of the Specification Models.

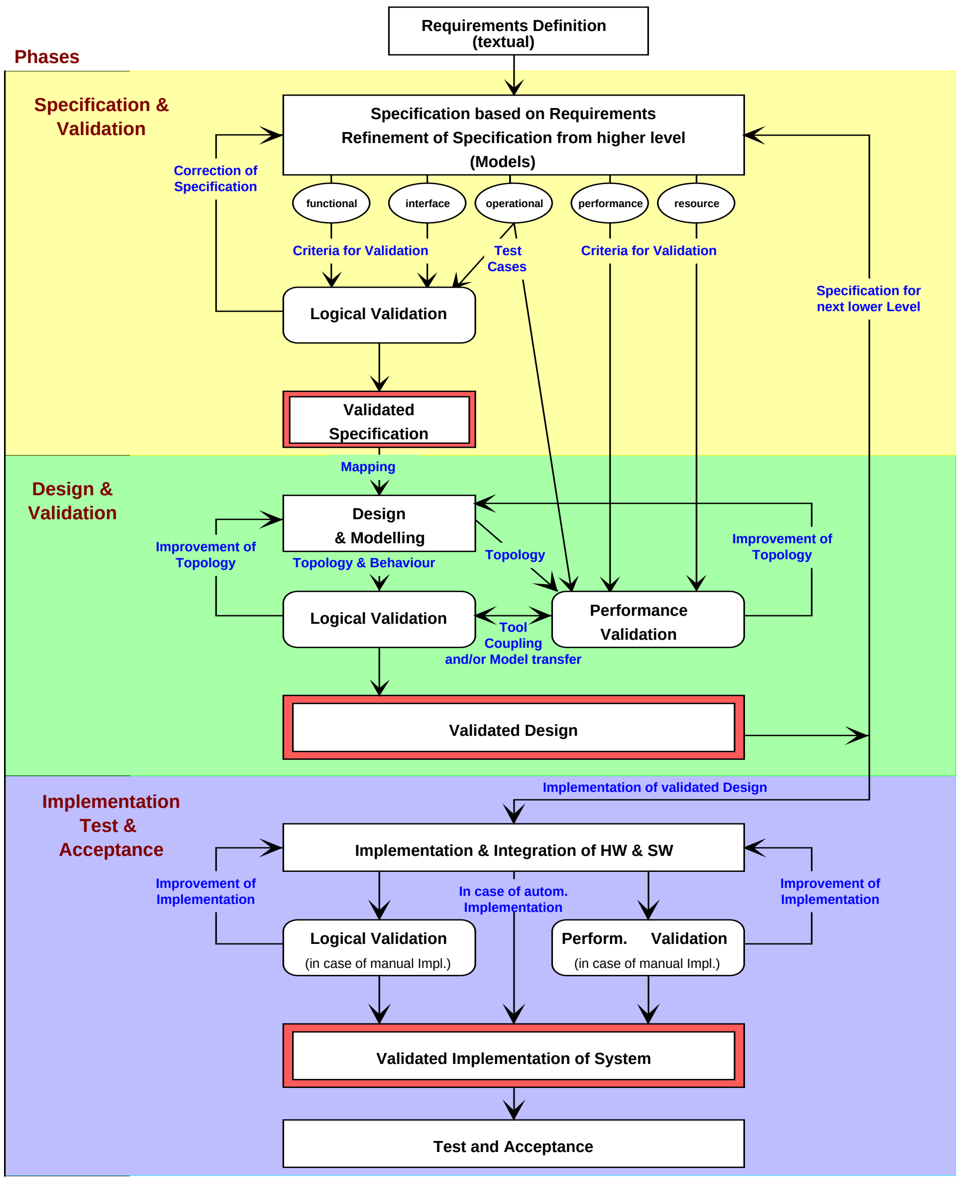

Requirements Definition (textual)
Phases
Specification & Validation
Specification based on Requirements
Refinement of Specification from higher level
(Models)
Correction of Specification
functional
interface
operational
performance
resource
Criteria for Validation
Test Cases
Criteria for Validation
Specification for next lower Level
Logical Validation
Validated Specification
Mapping
Design & Validation
Design & Modelling
Improvement of Topology
Topology & Behaviour
Topology
Improvement of Topology
Logical Validation
Performance Validation
Tool Coupling and/or Model transfer
Validated Design
Implementation of validated Design
Implementation Test & Acceptance
Implementation & Integration of HW & SW
Improvement of Implementation
In case of autom. Implementation
Improvement of Implementation
Logical Validation
(in case of manual Impl.)
Perform. Validation
(in case of manual Impl.)
Validated Implementation of System
Test and Acceptance

Figure 9: EaSyVaDe Phases of Early Validation

Now we establish a top-level architecture and a design indicated by the "D" in Fig. 8. From the Specification Models we derive Design Models[5] which shall provide the same behaviour behaviour and performance. Inside the Design Models we will identify again functions providing more of the functionality we need. We

[5] The transition from models used for validation of specification (Specification Models) to models used for design (Design Models) is described in detail in next section.

define functions and collect them into new Specification Models as we did before. We refine all top layer Design Models in this manner until we end up with a sufficient representation. Finally, we map the models to the architecture: then the top layer design is finished.

We have to extend the test scenarios, because we have to define the behaviour of the new Specification Models. To prove that our design meets the real-time, performance and resource requirements we have to execute the Design and Specification Models. Again we may have to do iterations if we are not successful immediately. After successful completion we have a validated top layer design and arrive at the "V" at the right side in Fig. 8.

After complete system validation we arrive at the bottom of our tree in Fig. 9. Then target code and hardware or VHDL code can be derived from the Design Models. This can be done manually or automatically layer by layer and for the multiple branches of system hierarchy (indicated by the "T" in the middle of Fig. 8). So we end up with the complete target system, shown by the rotated "T", which represents the stem and the bottom of our tree.

By iterating from specification to design and from design to specification we proceed from top to bottom. Vice versa, on each layer the validated Design Models from lower layers are integrated, indicated by the "I"-transition on the right side of Fig. 8.

After completion of design validation we take the Specification Models of next lower layer and validate them. On each layer we refine our architecture starting with top level architecture and going down step by step towards the final precise architecture.

On the layers below top layer we do not need separate validation of specification as the Specification Models are derived from the design (Design Models) on higher layer. These Specification Models are executed together with Design Models. The execution of all these models will result in Early Validation of design on the current layer. Design validation also implies validation of the specification of next lower layer or level. On top layer this is different as we start with an informal, textual representation. Therefore we have to do validation of formal before we validate a design the first time.

The validation on the current layer is based on the requested properties of the objects on next lower layer. For Early Validation we assume that the models on lower layers will provide what we have specified. If it turns out later, that the models on lower layers cannot provide what is requested we have to go up until we reach the highest layer affected. On this layer (and possibly on the layers below) we have to change the architecture, related design and derived specifications. However, as mentioned in chapter 2, the probability is not too high that such a dramatic change of architecture is needed, as we provide means for early detection of potential bottlenecks.

3.2 Model Refinement

When we proceed from layer to layer we refine the models. Specification Models are used to represent a model's specification. Similarly, Design Models represent architecture and design. For validation of the specification we concentrate on the "What", the functionality and performance an object has to provide. We limit the scope to the actual layer, we do not care what is below. What we need from the lower layers we try to provide by the Specification Model itself, e.g. dummy data, whilst Design Models would acquire data possibly from a "Data Handling" object..

When we introduce an architecture we care about "How" to provide the functionality, behaviour, performance and resources. Also, we extend our scope to the next lower layer and define another model, e.g. "Data Handling" which provides us with data and functions we need.

A Specification Model defines functions and data which form the so-called "interface" of an model. We have to validate this interface in the context of our design.

In fact, we have to do two steps:

1. we have to replace our (simple) Specification Model by a more sophisticated Design Model,
2. inside the Design Model we have to identify new objects for the next lower layer as we did before for top level when coming from textual requirements.

Fig. 10 shows these two steps. On the top of Fig. 10, left side, we are refering from a Design Model on layer n to Specification Model "comp3" on layer n+1. The reference is made by the interface, more precisely by the functions (and their parameters) and the data, and possibly data types, provided by the Specification Model. The Specifcation Model has a (simple) body representing the "What" of its interface, but ignoring the architecture and related topology. The interface of a Specification Model is kept when we establish a Design Model.

Possibly we will pass the Specification Model to a subcontractor using the model as unambigious, compact specification. The subcontractor will refine this model and will care about "How" to provide what is requrested by its interface. This is shown in Fig. 10 in the middle. In fact, the body of the Specification Model is replaced by a more sophisticated body, the body of the Design Model, indicated by the pyramid.

This Design Model identifies another Specification Model on next lower layer. Again, its functionality is described by its interface. It becomes executable by provision of a simple body, the body of a Specification Model.

In this manner we iterate from Specification Models to Design Models and from Design Models to Specification Models, when we proceed with Early Validation top-down.

It is essential for stability of the refinement process and the needed development effort, that refinement of models on a certain layer does not affect the higher layers. Otherwise the specification of objects on higher layers would have to be changed and we have to repeat validation on higher layers.

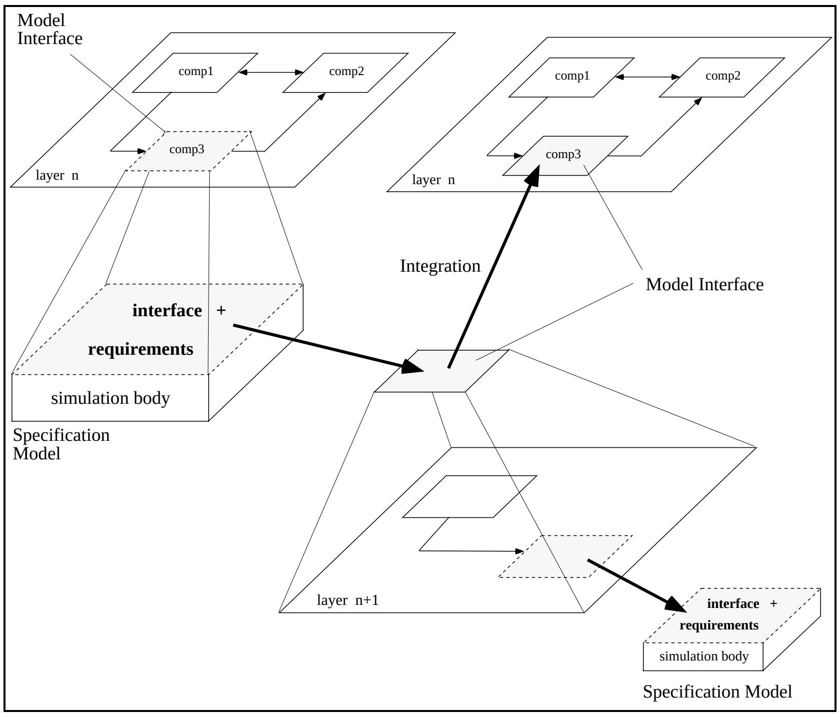


Figure 10: Model Refinement

However, it could happen when introducing Design Models, that we may identify other models on the current layer which were not defined before. If added on the current layer validation of specification is not affected, because the models are added during transition to design. The new models are not visible by the next higher layer, but is only internally needed on the current layer.

3.3 Simulation and Validation

Finally, we explain what we do during simulation of our system, i.e. when executing the models, and how we consider validation.

In principle, we distinguish between two different types of simulation, which result in two types of validation:

- behavioural simulation leading to behavioural validation,
  covered by the GEODE tool,
- performance simulation leading to performance validation
  covered by SES/workbench tool.

Behavioural validation comprises the evaluation, whether the following aspects of a system are consistent with requirements (terms are used in accordance with [6]):

- Functional requirements
  specifying "what" a system has to do. They define the purpose of the system.
- Operational requirements
  specifying "how" the system will run and how it will communicate with its environment. Operational requirements cover all user interface requirements as well as logistical and organisational requirements.
- Interface requirements
  specifying hardware or software elements, which the system or its components use to interact or communicate with its environment or other system components.

Performance validation means to confirm that the results of the performance simulation fulfill all of the following types of requirements:

- General performance requirements
  specifying numerical values for requested performance as timing characteristics like rates, frequencies, speed of the complete system, or the time which is allowable for reconfiguration activities.
- Constraining resource requirements
  giving lower and upper limits on size of physical resources like timeouts, processing power, main memory, disc space etc.

4 THE EASYSIM TOOL ENVIRONMENT

The EaSySim (EaSyVaDe Simulation Environment) allows to perform behavioural and performance simulation. We coupled two tools, namely GEODE which is based on the SDL CCITT'88 standard and SES/workbench (Fig. 11). The parts "Specification & Validation" and "Logical Design Validation" of Fig. 9 above are covered by the GEODE/SDL tool, whilst "Performance Design Validation" (as part of "Design & Validation") is achieved by coupling of GEODE and SES/workbench. For dedicated performance considerations SES/workbench can also be used in stand-alone mode.

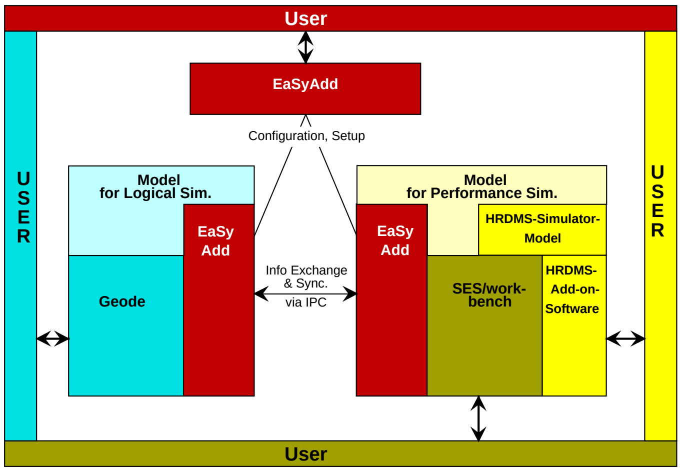


Figure 11: EaSySim Components

Fig. 11 shows the main components, GEODE and SES/workbench and additional software (Add-On) needed for tool coupling or improvement of the user interface. Also, the HRDMS [7] simulation environment or parts of it may be reused.

SDL will be used to specify or design the behavioural part of a system. This is because of the formal semantics which allows lexical and syntactic checks. One powerful feature of SDL is the possibility to specify Message Sequence Charts (MSC) which represent behavioural patterns of a system. Using the GEODE-simulator, differences between the expected behaviour, described by the MSC's, and the behaviour of the (executable) SDL system can be determinned.

Additionally, the designer can run a number of other checks, like

- deadlocks
- live locks
- SDL dynamic errors such as "arrays out of bounds" or "signal-sent to a non-existent process instance"
- transitions which are never executed.

In general the designer will start his system specification or design by using the GEODE/SDL-environment. Then he will specify the components and the architecture of the system. Afterwards he will assign behaviour to these components by the usage of FSM's and so-called process diagrams of SDL.

Because of its powerful statistical and animation facilities SES/workbench will be used to model the non-functional parameters (performance, resource allocation etc.) of the system. It should be stressed that it is also possible to model behaviour of a system in SES/workbench, but within SDL this can be done much better. This is because SES/workbench does not support validation of behaviour by provision of supporting functions. Therefore the designer cannot check his system model for deadlocks etc. Also, the automatic code generation feature of GEODE/SDL is not available in SES/workbench.

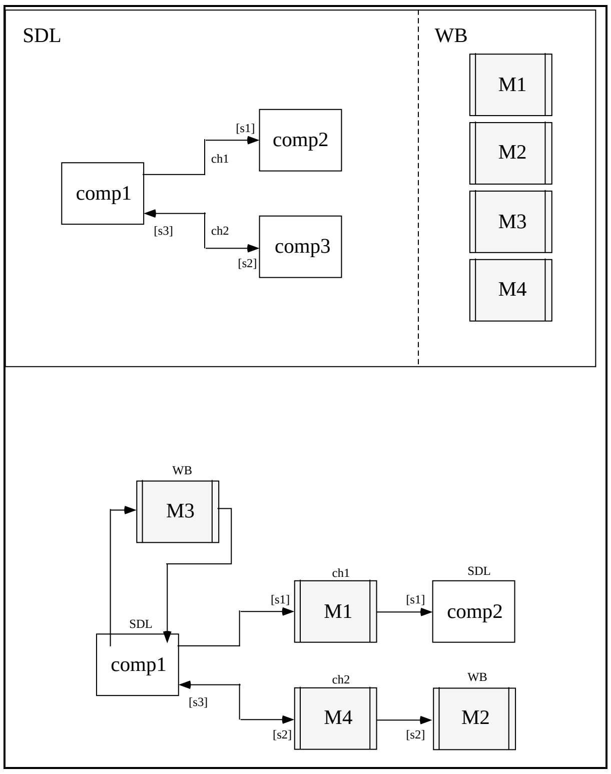


Figure 12: Integrating SDL and Performance

Figure 12 shows a typical application of EaSySim. In its upper part the SDL application is operated stand-alone. It consists of three coupled SDL components. Four SES/workbench submodels (e.g. from HRDMS) are available to be integrated for performance analysis. In the lower part of Figure 12 we see the coupled tool version which allows to exploit system performance. Three different integration types are shown:

1. Use of additional performance models

   Internals of SDL model comp1 are now driven by workbench submodel M3 which delivers performance inputs. In the simplest case model M3 may be just used to collect performance figures. Then only a few SES functions have to be included.

2. replacement of an SDL component

   performance model M2 of SES/workbench replaces SDL model comp3.

3. replacement of an SDL data channel by a performance model

SES/workbench model M1 is used to analyse the communication performance between SDL components comp1 and comp2. SES/workbench model M4 is used to investigate the communication performance between SDL components comp1 and comp3.

Running a standard SDL model in the coupled simulation mode will result in a system where all the logical behaviour is defined in SDL and the temporal behaviour is realized in SES/workbench[6]. So it is possible to use the powerful graphical and statistical features of SES/workbench to investigate the standard SDL timing behavior of a system. It should be stressed that the SDL model has not to be changed for this kind of investigation. All what has to be done is to call two scripts, which will prepare the SDL model for coupled simulation.

For combined simulation the designer has to define which performance models shall be used or shall replace SDL models. In the SDL code he has to insert directives containing this information. Then again the two scripts will automatically make the SDL model ready for execution in the coupled environment. These add-on's will not influence the behaviour of the SDL model when it is executed together within the GEODE environment. A switch from coupled simulation to stand-alone simulation and vice versa is possible at every time without having to make any changes in the SDL model.

The GEODE tool provides the capability to automatically generate target code (Ada, C) from the validated design and to debug the real-time code on the target system on SDL level.

The transition to VHDL models has to be done manually, currently. However, there are activities [8,9,10] for an automated transition from SDL to VHDL. We recognised, that a manual transition from high-level VHDL to RT-level VHDL is needed any way. Therefore our recommendation is to consider a transition from SDL to VHDL at the border to RT-VHDL, whenever possible.

Aspects of performance, quality of human interface and user acceptance, degree of automation, interfaces to VHDL tools or tools for analog simulation have been considered.

5 ISSUES FOR FUTURE WORK

We have now defined the EaSyVaDe methodology and have EaSySim available. We are going to apply EaSyVaDe the first time to an application, and we will see if we can realise what we have described before. Although we think that our approach is sufficiently mature, we have to prove it by a real example.

The following points are impacting the performance of the EaSyVaDe approach and we will carefully consider them:

1. the transition from textual, informal requirements to a formal specification, the identification of Specification Models,
2. the transition from Specification to Design Models and introduction of new Specification Models on same layer
3. the impacts from lower layers onto higher layers,
4. the mapping of models onto the architecture,
5. the trade-off between GEODE and SES/workbench, especially to which granularity the coupling between SDL and performance models shall be done,
6. the automated transition from GEODE to SES/workbench,
7. the time needed to execute simulation vs. representativity of modelling.

6 PILOT APPLICATION

Now the principal steps of EaSyVaDe are explained by an example. The selected example is the FDIR component (Failure Detection, Identification/Isolation and Recovery) of an on-board Data Management

[6] SDL is master of combined simulation, whilst SES/workbench is master of time.

System (DMS). The example shall clarify what the functionality of Specifications and Design Models is and how we proceed from specification to design.

6.1 Formalisation of Specification

According to the T-procedure system development starts with informal textual requirements (the "i"-candle on top left of Fig. 8). We do not want to list all the requirements related to FDIR. So we assume that we have behavioural and performance requirements regarding the FDIR component of a DMS and its environment and that we have already identified the top level Specification Models. Therefore our starting point shall be the formalised specification as shown by Fig. 13.

The related Specification Models are: Telecommand Assembly (TC), Telemetry Assembly, the DMS with the FDIR including Monitoring, Fault Identification and Recovery, Command Handling and Data Recording & Replay.

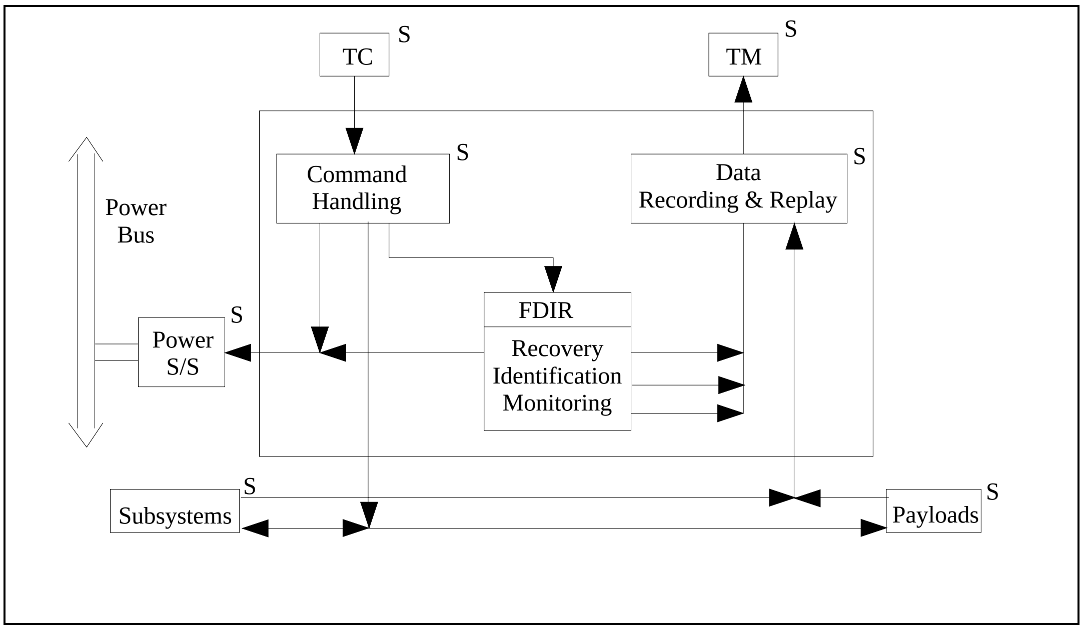


Figure 13: Top Level Formal Specification of DMS/FDIR and Its On-Board Environment

The TC Specification Model shall issue all commands which are needed to operate the DMS concerning the FDIR. The TM Specification Model shall be able to receive all relevant data and reports. Command Handling will process the commands from ground, which includes decoding, validation and distribution. Data Recording & Replay will handle all operations needed to store data and to transfer them later to ground.

FDIR of the DMS itself consists of three main groups. Monitoring will provide operations like StartMonitoring, DefineLimitSet, SelectLimitSet. Identification will deliver reports according to a failure symptom. Recovery will initiate actions, e.g. a reconfiguration, to keep the system alive and will send commands to the power subsystem. All FDIR components will report to ground via Data Recording & Replay.

The power subsystem shall be able to receive power commands either from FDIR/Recovery or from ground via Command Handling and to activate the hardware components. During simulation the power or initialisation status of components shall be considered. TC has direct access to subsystems and payloads via Command Handling.

All these Specification Models shall provide the operations needed to handle faults, to communicate with ground and to recover from a fault. However, they include very simple functionality, just what it is needed to validate system operations for FDIR on top layer. So it could be sufficient that Monitoring just flags randomly a violation of nominal conditions and reports the faults. No real data acquisition shall be foreseen Similarly, Identification may just issue an arbitrary, but predefined report. Recovery also may send a simple message together with a power reconfiguration command to power subsystem.

It is essential that not only the logical behaviour shall be specified, but also the performance and resource constraints. So command rates, down-link rates, monitoring performance figures shall already be defined for Specification Models and used for simulation as far as possible. E.g. timeouts can be implemented by FSM's as shown by Fig. 4 and already used by Specification Models.

For Early Validation of specification only SDL models are established. MSC's, FSM's, SDL are used to implement the Specification Models. If SDL signals or data inputs are used, timeouts shall be specified. Data rates shall be assigned to SDL data channels. Other performance or resource data shall be documented in the models. Then behavioural simulation is performed with the GEODE tool followed by combined simulation for analysis of performance.

In this manner the consisteny and completeness of logical behaviour shall be validated on top layer, and performance/resource figures shall be sufficiently considered.

6.2 Design Validation

Having achieved a validated Formal Specification by Specification Models we will establish the design (the "D" in Fig. 8 on top middle). Fig. 14 shows a first iteration of the design after introduction of an architecture. The used models are either Specification or Design Models indicated by "S" or "D".

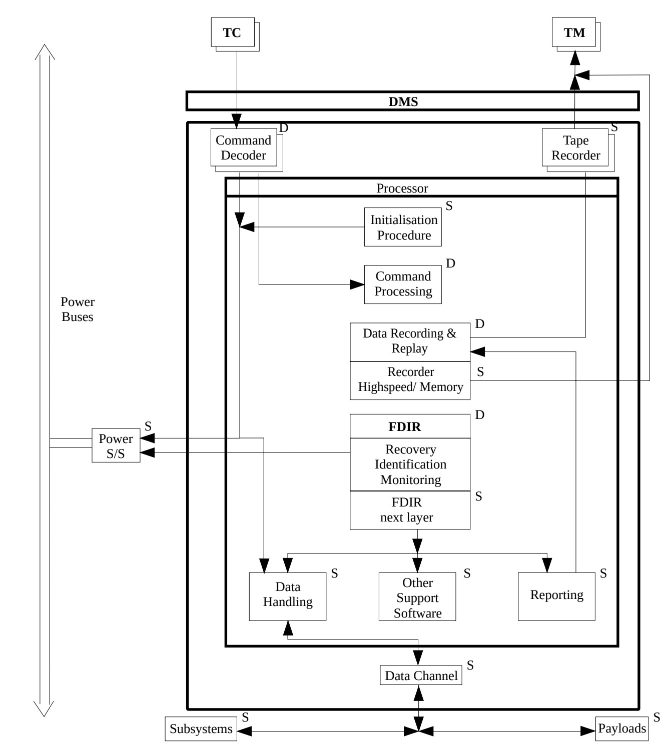


Figure 14: Top Level Design of DMS/FDIR and Its Environment

By the architecture the system engineer introduces hardware and maps objects to hardware components. Inside the DMS we identify as hardware elements a Command Decoder, a Tape Recorder, a Processor and a Data Channel, which connects the Processor with subsystems and payloads and their equipment. For simplicity the communication channels between Command Decoder, Processor and Tape Recorder are not shown.

The command rates, down-link rates, monitoring performance figures already defined for Specification Models are now forwarded to Design Validation.

Consequently, for the top layer the consisteny and completeness of logical behaviour and performance/resource figures shall be analysed and validated.

The transition from the Formal Specification to the corresponding design is described in more detail by the following figures (Figs. 15 and 16). We take as representation the system hierarchy as this makes it easier to follow the decomposition steps. Fig. 15 shows the system hierarchy for the top level Formal Specification corresponding to Fig. 13 above.

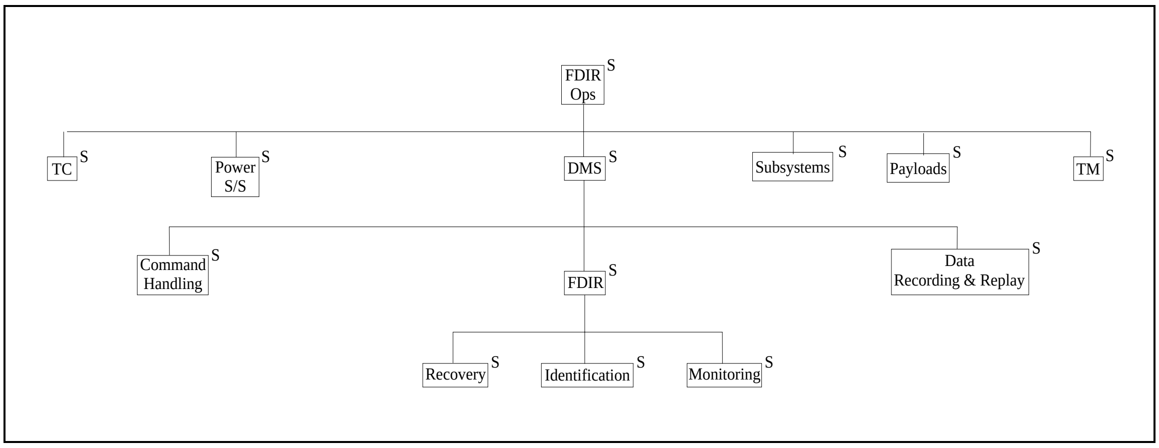


Figure 15: System Hierarchy of Top Level Formal Specification

The Specification Models for "Command Handling" and "Data Recording & Replay" are refined into Design Models and the need for new lower level Specification Models is identified. Fig. 16 shows these Design Models with their internal new objects.

The Design Model of "Command Handling" and its Specification Models are allocated onto different hardware items: the Processor and the Command Decoder. Performance may be impacted by this specific architecture separating the two internal objects of "Command Handling" from each other.

Similarly, it is for Data Recording & Replay. By its Design Model we introduce two new internal Specification Models: "High Speed Recorder" and "Tape Recorder". Both subcomponents provide the complete functionality and performance of the only, previous Specification Model.

The "Command Decoder" (an object on next lower layer) may now already include all the relevant commands to be processed and may take the requested decisions and actions based on that knowledge, whilst before the (simple) Specification Model of "Command Handling" just implemented the functionality visible at its interface and provided its decisions only on e.g. random base. Similarly it is for "Data Recording & Replay".

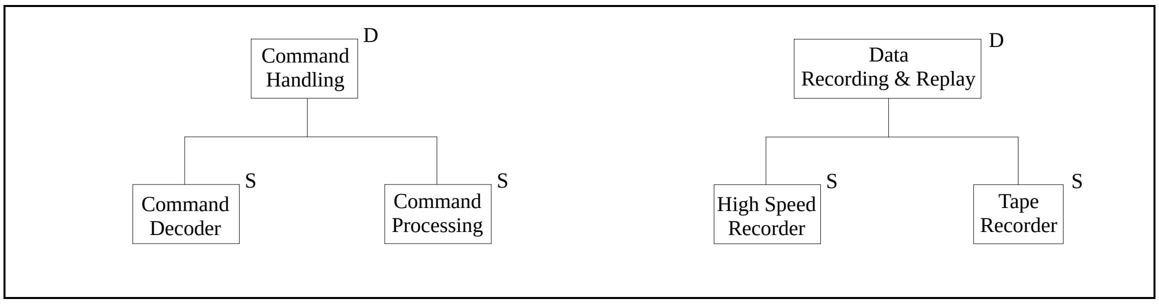


Figure 16: Decomposition of Design Models

For the FDIR object also Design Models are established which reference Specification Models of the next lower level. All FDIR models are mapped onto the Processor.

Fig. 17 shows the corresponding system hierarchy. One essential point is that the Design Model of "Monitoring" now really shall acquire data. Therefore its performance relies on the Specification Model of the "Data Handling" component. Similarly, the performance of "Identification" and "Recovery" depends also on "Data Handling". In this manner performance requirements together with functional requirements are passed top-down.

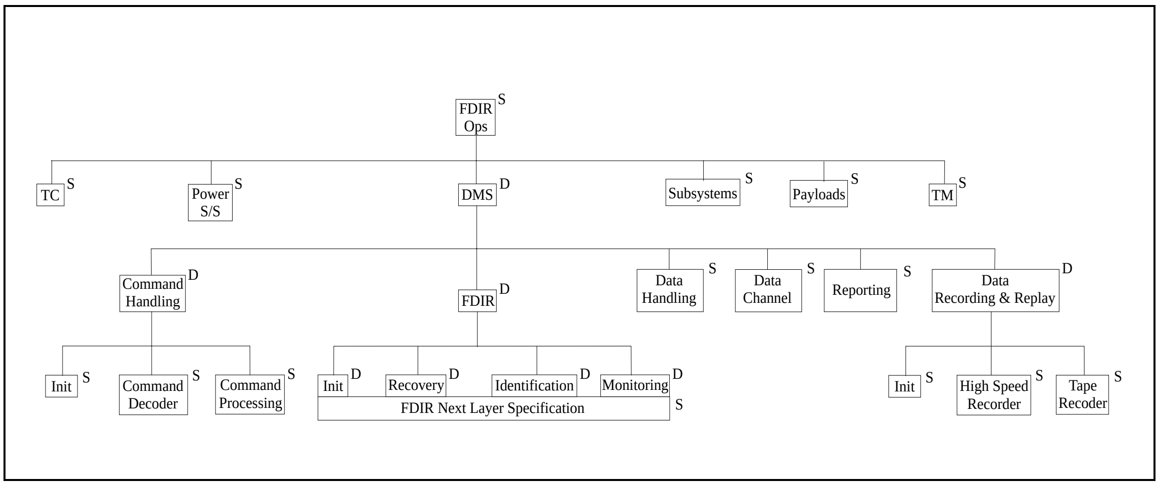


Figure 17: System Hierachy of Top Level Design

"Data Channel" is introduced as medium to acquire data from "Subsystems" and "Payloads". Now the "Data Handling" component (and all the components above, of course) are validated relying on Specification Models of "Data Channel", "Subsystems" and "Payloads" and their equipment.

The essential point for successive validation is that we are concentrating for each iteration on validation of certain components provided that the objects (on next lower layer) on which they depend will really behave as requested.

To summarise:

1. SDL Design Models are established taking the interface from the related Specification Models.
2. Inside the the Design Models Specification Models are built.
3. In parallel, performance models may be established together with test scenarios to evaluate some performance characteristics or to develop the needed performance models in stand-alone mode.
4. After verification of behaviour GEODE is coupled with SES/workench. Performance models are added or they replace SDL Specification and Design Models. In coupled mode a representative system model will be executed and results will be analysed.
5. If all the requirements are met we will have successful verification of design.
6. In addition, we check if the system model provides what we expected. If it does, we have achieved Early Validation of design on current layer.

7 CONCLUSIONS

The EaSyVaDe methodology allows to continously validate a system when refining it hierarchically layer by layer from top to bottom. Specification and design are made executable by use of models and simulation. The development process iterates between specification and design top-down. This procedure is significantly different from most known approaches and is considered as essential for succesful and efficient system development. We validate behaviour and performance together, because only then we can really achieve "Early Validation" and we can reduce the development risk.

The term "Early Validation" is explained as "preliminary validation" based on assumptions on properties of lower layers. This preliminary validation is considered as rather close to final validation. We hope that EaSyVaDe will significantly improve the system development process and will lower the risk not to achieve what was expected.

The toolset EaSySim is provided to support the EaSyVaDe methodology. With SDL and performance models we will be able to apply what was described above for a pilot application. This exercise shall confirm that the current concept is feasible or shall deliver a feedback on potential improvements.

ACKNOWLEDGEMENT

The authors thank for their contribution to EaSyVaDe Mr. H. Raschhofer and Mr. A. Simma from University of Linz, for his contribution to EaSyVaDe Mr. J. Bortolazzi, and for his support of our work Mr. Th. Stingl, both from FZI.

REFERENCES

[1] Barry W. Boehm, A Spiral Model of Software Development and Enhancement, *IEEE-Computer*, pp 61-72, May 1988

[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] Systems Engineering: Principles and Practice of Computer-based Systems, Engineering, ed. Bernhard Thome, *Wiley*, 1993

[5] Gerhard Chroust, Modelle der Software-Entwicklung, *Oldenbourg Verlag*, 1992

[6] ESA Software Engineering Standards, *ESA PSS-05-0, issue 2, February 1991*

[7] HRDMS: "Highly Reliable DMS and Simulation", Final Report, September 1994, *ESTEC contract no. 9982/92/NL//JG(SC) , Noordwijk, The Netherlands*

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

[9] 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*

[10] 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*