A Strategy for Development of High Quality Embedded Systems

Translation of the contribution to

CASE Anwendertag 1995
DLR, Göttingen
September 19, 1995

Rainer Gerlich

e-mail: gerlich@t-online.de

A STRATEGY FOR DEVELOPMENT OF HIGH QUALITY EMBEDDED SYSTEMS

Rainer Gerlich

e-mail: gerlich@t-online.de

1. INTRODUCTION

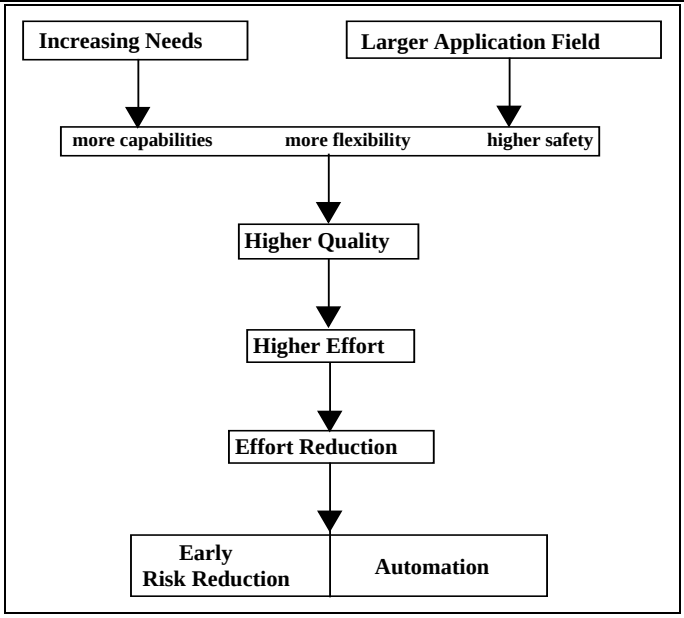

Fig. 1: Increasing Needs for System Development vs. Efficiency

Embedded systems are broadly used for control and data management tasks. A large number of such applications is safety critical and/or requires sufficient reliability. With increasing needs (Fig. 1), (e.g. for more functionality or better performance), embedded systems are becoming more complex. As the amount of flexibility and functionality required increases, software takes up an ever larger percentage of the system.

From the past we know all the problems related to software development, especially pertaining to real-time software. The higher the percentage of software in the system, the larger its potential impact, especially if faults occur. Consequently, we have to ensure that software will be of better quality in future. Surely, this is a **must** for safety-critical applications.

Quality implies reliability, dependability and safety together with the desired functionality and performance (Fig. 2). Higher quality also means higher effort. For industrial applications, effort in terms of costs and time is also a critical parameter for system development. If effort becomes too high, the economical benefit may be not achieved and development of the system-of-interest is dropped.

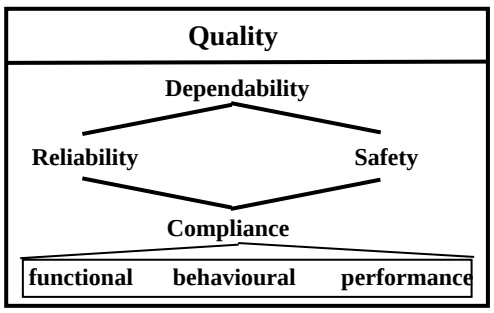

Fig. 2: Quality Measures

One cannot conclude from the "shape" of the software, (i.e. its source code), whether it is correct or not. Correctness of software can only be demonstrated when the results produced by software are visualised. This usually occurs at a rather late phase of the development lifecycle; for example, in case of the V-model, during test and integration (Fig. 3) at the right branch of the "V".

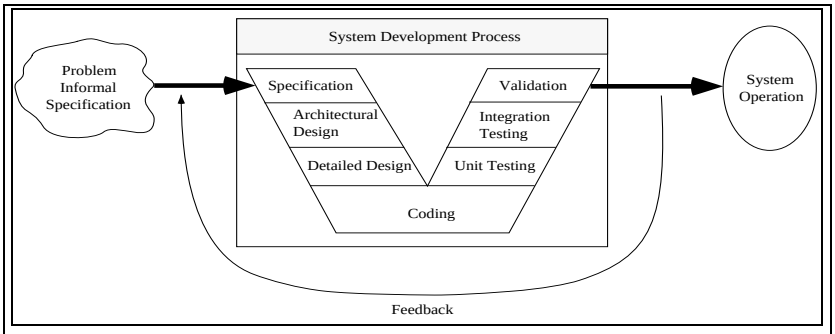

Fig 3: Development Risk in case of the V-Modell

Hardware faults are mainly due to material degradation which tend to occur randomly; only in some very rare cases is hardware improperly specified or designed. Software cannot degrade. All software faults are related to wrong

specification, design or coding.

Hardware faults caused by degradation can be corrected with redundant components. In the case of faults related to wrong specification, design or coding, redundancy cannot help. To decrease the probability of fault occurence, fault avoidance techniques have to be applied. These include formal verification of the system and its validation by execution and visualisation of the results.

To be prepared for the future, first, identification of the weaknesses of the current system and software development methods are needed concerning remaining risks and effort. Secondly, one has to provide an alternative. It is essential to consider optimizing the complete system lifecycle and not just concentrate on its individual phases. Also, task sharing between an engineer and tools must be reconsidered (Fig. 4).

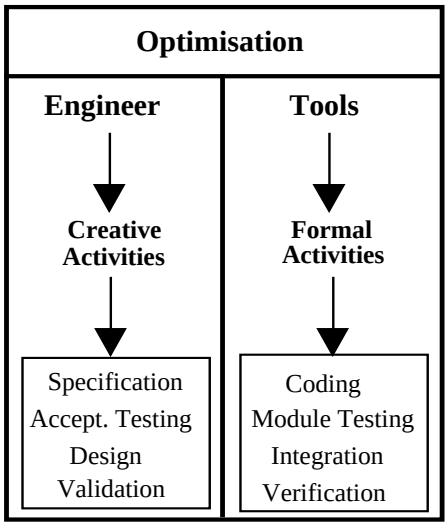

Fig. 4: Optimisation of Task Sharing

In the field of space applications, system complexity is also increasing. Reliability and safety demands in this application field are extremely high. But costs and development risks also have to be considered for future spacecraft, and must be kept at an acceptable level. Therefore ESA initiated activities by which new system and software development methods were defined to help to master the development of future systems.

During two pilot projects HRDMS [1] and OMBSIM [2], new approaches were applied and tested. In OMBSIM the development methodology "EaSyVaDe" was defined which introduces a systematical approach concerning early validation, incremental development and embedded systems. In the follow-up project, DDV [3], EaSyVaDe is applied to the specification and design of the fault management part of a future comet mission. EaSyVaDe (Early System Validation of Design) [4] is the result of investigations and feedback from projects since 1986.

EaSyVaDe can also be applied to other application fields. Even smaller projects can take advantage of it. The EaSyVaDe methodology is supported by the development environment, EaSySim.

2. EARLY AND INCREMENTAL VALIDATION

Risk Reduction	Automation
executable models (simulation)	formal methods
refinement: stepwise, hierarchical	automated checks
simulation: nominal conditions, non-nominal conditions	code generation from simulation
homogeneous system approach: hardware-software co-design	tool integration
late hardware-software trade-off	tool improvement (add-on's)
Increased Reuse	

Fig 5: Strategy to Increase Quality and Efficiency

To achieve higher efficiency, optimisation of task sharing between engineers and tools is a must (Fig. 4). An engineer shall concentrate on the creative part of system development, whilst tools shall take that part which can be formalised and automated. Therefore an engineer will be more involved in the early development phases and less is the later ones. This leads to early validation and an increased degree of automation (Fig. 5). Currently, it is just the other way around. An engineer spends a lot of time coding, module testing and integrating during rather late development phases.

Consequence, quality is increased for the following reasons:

- With incremental and early validation the expected system properties are known and can be continuously be refined. The system is validated for functionality and for performance. With executable models nominal and non-nominal scenarios can be simulated and evaluated. This strategy decreases the danger of not achieving the desired goal.

- By using formal methods, tools are able to automatically detect discrepancies in a system such as incompatible interfaces or wrong behaviour. Automated (target) code generation also becomes possible, and development efficiency is increased by this higher degree of automation.

The higher the degree of formalisation, the higher the percentage of system develoment which can be covered by tools.

Quality is increased because more development faults are detected earlier and automatically and to a higher percentage by optimizing tool usage. Automated (target) code generation from pre-validated models ensures that no faults are introduced as might occur in the case of manual coding.

The percentage of reuse increases, because by standardisation and formalisation, similarities can be readily identified early-on. If needed, properties, behaviour and interfaces of the new components can be harmonised with those of already existing components.

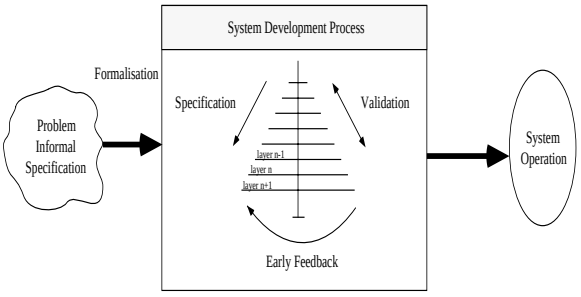

Fig. 6: Early and Incremental Validation

EaSyVaDe helps to identify development risks rather early because **executable** models are used for specification and design. Only by executing the software can one identify what is specified and designed and if this is compliant with what is expected. EaSyVaDe ensures a smooth and consistent transition between specification and design.

After each step of refinement, a functional and performance check is possible by executing the models; a preliminary, "early" validation of the specification and design can be achieved in an early development phase. Obviously iterations have to be performed. When the system is sufficiently understood and when it is compliant with the user needs at a certain level of refinement, it is then expanded to the next lower level of the system hierarchy (Fig. 6). Again, the system is validated under consideration of the extended components.

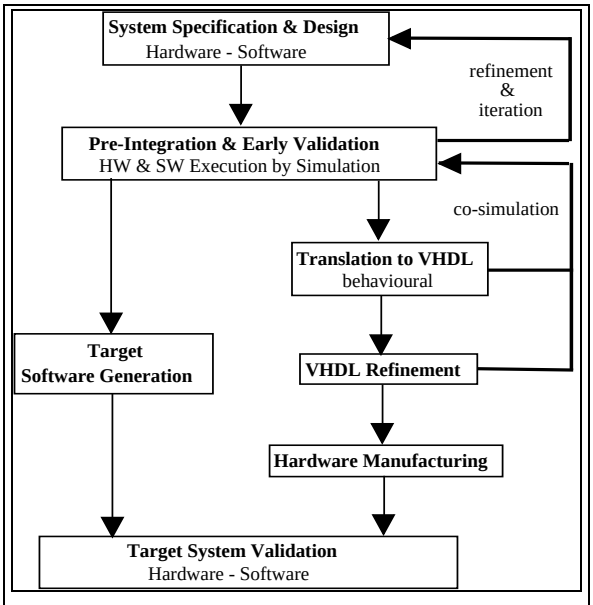

Fig 7: Homogeneous Approach for System Development

Refinement of a system is performed incrementally in a hierarchical approach with associated early validation at each level. Early Validation is based on the assumption that the properties requested for the lower level components can be fulfilled. Of course, there is still a risk that the lower levels may not provide what is imposed on them. Only, if the lowest level has been successfully reached, one can be sure that all what is requested can be provided. However, as the needs are well known, the probability is significantly higher to identify what is not possible compared to the case when the needs are uncertain or completely unknown. This is of special importance for performance properties which are usually not considered.

The models represent system components, neither hardware nor software (Fig. 7). Therefore the hardware-software trade-off can be postponed until the needs are accurately known. Currently, this decision has to be done rather early. It can hardly be revised later on if needed, and then only with rather high effort.

All system components can be executed and validated within a "homogenous" environment (i.e. a unique consideration of hardware and software.) If one decides that a certain component shall be represented by software the target code is generated automatically. In case one opts for hardware, the model will be translated into "Behavioural VHDL". This translation is currently under development [5,6], specifically the transition from SDL [7] to VHDL [8]. After the translation, the component can further be refined in VHDL. As tool vendors are improving tool interfaces towards co-simulation, it may become possible in near future to perform SDL-VHDL co-simulation with VHDL models refined down to gate-level.

System validation implies verification and consists of two primary activities (Fig. 8):

- verification of static and dynamic properties done by tools, and
- validation of the operational behaviour under nominal and non-nominal conditions done by the engineer.

Due to formalisation, the verification steps can automatically be performed by tools, whilst the validation is an engineering task supported by tools for execution and result analysis.

Principal System Validation Steps	
verification (tools)	checks by formalisation static properties (interfaces, types, ...) dynamic properties (behaviour, state transitions, ...)
validation (engineer)	system execution nominal conditions non-nominal conditions

Fig. 8: Verification and Validation

Consequently, the possibilites to drive development in early phases are better. Tools are used for such tasks for which their benefit is most appreciable. This allows increased quality with a comparable or greater development efficiency.

3. CONCLUSIONS

EaSyVaDe makes the development of embedded systems more efficient at high quality levels by introducting early validation, formalisation and automation. This is of high interest for safety-critical applications and applications for which a high degree of reliability and dependability is required, but also for applications outside the area of embedded systems because development time and costs shall always be kept low. The task of an engineer is concentrated on the creative and critical activities. Tools are applied to activities like consistency checks and code generation. This requires use of formal methods, even for description of real-time behaviour.

EaSyVaDe extends an approach already applied in control applications [9,10] using formally defined standard components for specification and design. In a general manner it introduces this approach into system and software development for a much broader range of applications.

Also, EaSyVaDe supports project management. It is especially useful for system development performed at different sites. The executable models can be provided as specification and the refined models delivered in response by contractors for re-integration.

The related development environment EaSySim consists of the commercial tools GEODE [11] and SES/workbench [12] and additional software for coupling both tools together and improving their functionality.

It is planned to apply EaSyVaDe for further projects, also outside the space area, and to integrate EaSySim in an already existing development environment which also covers the other tasks of the system or software lifecycle such as configuration management.

This paper only gives a brief introduction to the principal EaSyVaDe approach. Important aspects like performance and reliability of automatically generated code, the transition between specification and design, consideration of functionality and performance could not be addressed sufficiently or at all. Further information is included in papers [13,14,15,16], which are available on request.

REFERENCES

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

[2] OMBSIM (On-Board Mangement System Behavioural Simulation, ESTEC contract no. 10430/93/NL/FM(SC), Report, OMBSIM/FR/001/DOR/, September 1995

[3] DDV (DMS Design Validation), ESTEC contract no. 9558/91/NL/JG(SC), Formal Methods and Tools, Selection & Justification Report, TR/171/PhH/95, 30.06.95

[4] 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

[5] M.Lehmann: Automatischer Übergang von SDL nach VHDL (in German) (Automated Transition from SDL to VHDL), master thesis, Fachhochschule Albstadt-Sigmaringen, September 1995

[6] T.B.Ismail, A.Jerray: "Synthesis Steps and Design Models for Codesign", IEEE Computer, February 1995, pp. 44-52

[7] ITU, Recommendation Z.100, Specification and Description Language, SDL, 1993. Blue Book, Vol. X.1, and appendices A, B, C, D, F1, F2, F3

[8] The Institute of Electrical and Electronics Engineers: "IEEE Standard VHDL Language Reference Manual (IEEE-1076-1992/B)", New York, 1993

[9] Matrix-X, Integrated Systems Inc., 3260 Jay Street, Santa Clara, CA 95054-3309, USA

[10] K.J.Vogel, D.H.Johnson: An Integrated COTS Prototyping and Test Environment for Spacecraft Control Systems, EUROSPACE Symposium on "Techology and Applications for Space Data Management Systems", January 25-27, 1994, Rome, Italy

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

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

[13] R.Gerlich, N.Schäfer, A.Schäferhoff: Early Validation of DSM Design by a Reusable Environment, EUROSPACE On-Board Data Management Symposium on "Technology and Applications for Space Data Management Systems", January 25-27, 1994, Rome, Italy

[14] 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

[15] R.Gerlich, Th.Stingl, Ch.Schaffer, F.Teston, G.Martinelli: Use of an Extended SDL Environment for Specification and Design of On-Board Operations, Systems Engineering Workshop, ESTEC, Noordwijk, The Netherlands, November 28-30, 1995 (paper available by end of September 1995)

[16] R.Gerlich, C.Joergensen: An Alternative Lifecycle Based on Problem-Oriented Methods and Strategies, International Symposium on On-Board Real-Time Software, ESTEC, Noordwijk, The Netherlands, November 13-15, 1995 (paper available by end of September 1995)