MISSION LEVEL DESIGN OF AVIONICS

Horst Salzwedel, MLDesign Technologies, Inc., Palo Alto, California

Abstract

Aerospace systems are characterized by architectural complexity, dynamic interaction between subsystems, and complex functionality, understood only by teams from different disciplines. 20 years ago the major challenge was the multidisciplinary design of avionics. Over the past 20 years, design methods and tools have been developed to cope with these challenges. Today the complexity of networked electronics in aircraft and the interaction of hardware and software impose similar complexity and design challenges. According to Moore's Law, closely followed by industry, the complexity of electronics increases by a factor 100 every 10 years, requiring to increase abstraction in the design methodology, in order to cope with this increase of complexity. This paper shows the move towards performance and mission level design and its advantages over functional level design approaches.

The challenge of complexity

Aerospace systems are characterized by architectural complexity, dynamic interaction between subsystems, and complex interdisciplinary functionality. The design challenges presented by the avionics of these system with more than 1000 electronic control units (ECUs), include not only the architectural and functional complexity of the avionics systems themselves, but also the complexity of the organizational structure of the design teams from mission specification, design, validation and verification, implementation, test, training, and operation.

The introduction of stability augmentation systems in the 1960th and 1970th coupled different areas of engineering developments and caused interdisciplinary problems in the designs. Every aircraft prototype tested exhibited aero-servo-elasticity problems. The analysis of these problems showed that the main cause for these problems was flawed specifications resulting from insufficient communication between design engineers of different areas, the believe that verification of implementation is sufficient to catch mistakes (e.g., Saab 39), and use of incompatible modeling techniques and tools.

Multidisciplinary research led to modeling and design methodologies that considered engineering expertise and the limit of it for the design flow, e.g., for the development of integrated flight propulsion control systems. Multidisciplinary research sponsored by AFWAL [2] let to the development of generic software tools like Ctrl-C®, MatriX® and their derivatives Matlab® and Octave™, that permitted to combine functional level models from different disciplines to reduce these problems.

An example of successful multidisciplinary modeling can be found in the design of a transfer alignment filter. This filter is responsible for the transfer of navigational information from an aircraft navigation system to that of a missile under a wing. Early developments for this filter considered a rigid body connection between the aircraft and the missile, and treated all other effects as white noise, causing large alignment errors and large alignment times. Large research efforts and tests could not solve the problem. The availability of Ctrl-C permitted to easily combine models from structural dynamics, aerodynamics and flight control into a unified model. This permitted to identify structural flexing as the major problem for filter accuracy and alignment time. The inclusion of these effects improved the accuracy by more than a factor 100 and reduced the alignment time by more than a factor 100 [2].

With the proliferation of electronics in nearly all type of engineering systems, and the rapid increase of the complexity of electronics and software, the major challenges of system development today are validated specifications, and the gap between system design and networked hardware/software implementation.

In the 1970s chip masks were designed with CAD systems. In the 1980s chip complexity had increased by a factor of 100. Engineers could no longer handle this complexity. Languages like Verilog and VHDL were introduced to design chips at the logical level and translate designs into masks. At the beginning of the 1990s, the complexity had again increased by a factor of 100. Software tools like SPW® and COSSAP® were introduced to raise the abstraction of electronics design from the logical level to the functional level. This permitted e.g. the design of modems in 3 month. The problem remained to translate behavioral models into RTL level models and validation of the resulting RTL level models against design specifications at the behavioral level.

Today (2000s) the complexity of electronics has increased by a factor of 10000 since integrated functional level design tools have been introduced for system design, and by a factor of 100 since design of complex electronic moved from logical level to functional level. Additionally, chips have become systems and complex systems like aircraft, spacecraft, automobiles and communication systems are dominated by networked electronics with embedded software. This compounds the problem of the gap between design and implementation. This increase in complexity has caused major problems throughout the industry, including,

- Flight control systems failures such as those of Saab 39, Osprey, …
- The failure of the first Ariane 5 rocket because of a value overflow. The implementation was not tested against the mission
- The development of the Teledesic satellite system was discontinued after it was found that major design specifications had to be revised late in the design
- In 1999 2 spacecraft to Mars failed because of a mixup between units used by different design teams
- The Boeing 702 series of spacecraft exhibit major problems.
- Electronic control units in automobiles exhibit failure rates of up to 3000ppm (required <10ppm)
- Luxury class automobiles have to be recalled because of unwanted interactions between large number of networked electronic subsystems

The systems above exhibit the following problems in their design processes:

From general requirements for the overall systems, written specifications and an overall architecture of networked systems is derived, Figure 1.

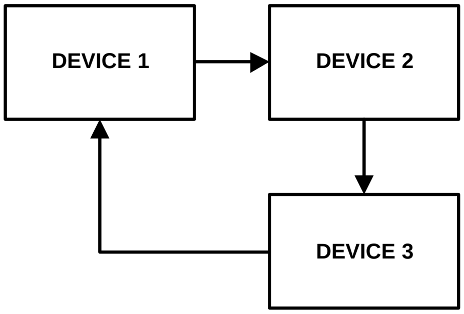

Figure 1: Architecture of networked system

The subsystems and networks are designed by established processes, Figure 2.

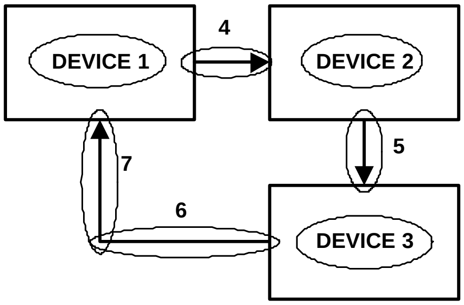

Figure 2: Networked System

Each of the design engineers/design teams makes certain assumptions about how their design will interact with near or far subsystems of other design teams. These assumptions will not be the same. Some will not be documented. Additionally, many subsystems may be reactive to the environment and hence events cannot be predicted. Hardware descriptions may be incompatible for those of different subsystems. Insufficient communication between design teams will prevent that all required information is passed. Simulations of the overall functional or implementation models are not possible and the overall system cannot be validated and verified on a computer. Coverage analysis does not cover performance values and constraints. Coverage analysis is not possible by hardware-in-the-loop simulation or hardware tests.

When the independently developed subsystems are put together for the overall system, it first does not work. Problems are fixed on a local level. Validation against overall system requirements cannot be made since they are not executable and may be inconsistent. Far effects are often discovered late in the design or during operation by customers.

A recent ESPITI study, Figure 3, shows which stage of a design flow causes critical problems in system design. The probability is nearly 60% that the specifications cause critical problems. The probability for critical problems caused by modeling and design is about 25% and less than 20% by implementation. These critical problems could be in hardware or software, but often in the coupling between them. A major contributor to this problem is that the design are done at the functional level. However, complex systems can no longer be simulated as a whole at functional level. Specifications can therefore not be validated at this level of design abstraction.

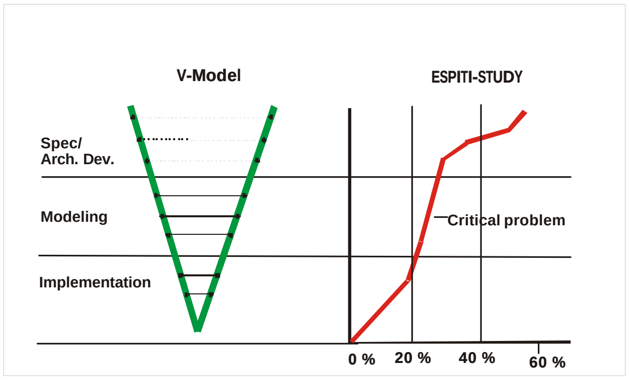

Figure 3: Critical problems in the design of complex systems

In order to achieve the required up to a factor 100 and more quality improvements in system development, all stages of the development process have to be significantly improved. The main effort has to concentrate on improving the quality of specifications and design, where most of the problems are. The following sections describe recent technology developments to achieve this quality improvements and significantly reduce development times.

Improving quality of specifications

Three developments have been most promising to solve the problem of inconsistent and wrong specifications and on meeting specifications in design. These are,

- Making specifications executable
- Finding a common base for the description of diverse requirements from different engineering disciplines that contribute to aerospace system design
- Making it possible to test functional level designs against executable specifications of the overall system

Making Specifications Executable

For the development of complex planetary and interplanetary space systems, detailed mission analysis precedes the design phase and implementation phase, in order to get validated, executable specifications for the overall system. During the design phase, all components of the design and implementation are validated against these specifications. Where possible, hardware in the loop test are performed to test the implementation against the design and the mission level specifications. This design approach led to highly reliable systems that roam the solar system up to the outermost planets.

Today components and subcomponents like telecommunication systems, operational infrastructure, embedded systems, and processors have complexities that far exceed those of early spacecraft to the outer planets. Since Shannon’s law on algorithmic complexity asks for more than Moore’s law on processor performance can deliver, it is no longer possible to simulate the HW/SW implementation of components and subcomponents against the mission requirements. Additionally, a design engineer can no longer comprehend the complexities of a design at a functional level. Designs have to be performed at higher levels of abstraction.

In Ref. [3] is a hierarchical Mission Level Design (MLD) approach is developed, that generalizes the design approach for deep space missions and makes design decision where quantitative information is first available:

1. A validated and executable mission is the behavior of the system that uses a component to be designed
2. Validated specifications of the functional behavior are generated by validating the high level architecture and performance of the component against the mission level requirements
3. The functional behavior of HW and SW is verified and validated separately and in combination against the specifications stemming from the architectural/performance model

The top-level architecture and performance requirements are modeled using discrete event (DE) models and finite state machines (FSM), Figure 4.

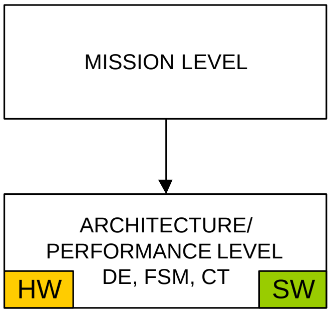

Figure 4: Design stage 1: Definition of mission and top-level architecture

Initial HW models abstract resources such as memory, busses, CPU cycles, etc into quantity and server resources. Software is modeled by their execution time on a target HW (this is later updated with estimated values from the SW performance estimator.) The mission model may be diverse and is typically modeled by DE, FSM and continuous time (CT) models. Challenges that have to be addressed at the architectural/performance level include [4,5],

- Dealing with complex architectures, with complex functionality in each subsystem und a high degree of concurrent processing,
- Dealing with dynamic events with complex interactions between subsystems,
- Dealing with data, tasks & architecture dependent interactions, and
- Dealing with use cases and mission scenarios

Architectural/performance models have the required quantitative information to make decision on what components have to be implemented in HW and which in SW.

Design iterations at this level of abstraction reduces the risk of design errors by testing the design early in the design process, where errors are easy to fix. Experts suggest that design at the performance/architectural level can determine as much as 80-90% of a system's total cost, performance and time to market.

Some system developments for which this level of abstraction have been used are,

- Development of specifications for an Air Traffic Management (ATM) system (Case 1 of Ref. [4]). Figure 5 shows the required PDF for an ATM for an Inmarsat type satellite

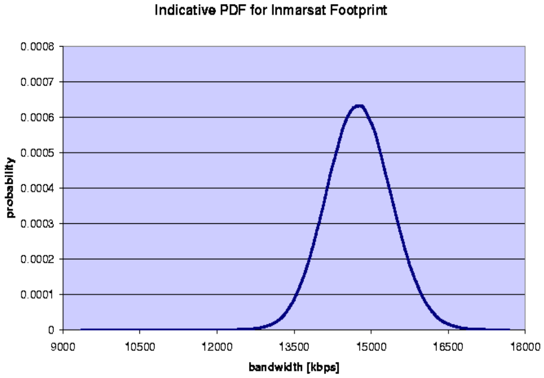

Figure 5: Required satellite PDF for ATM

- Development of a performance level model of HW and SW of a terrain following system, Figure 6.

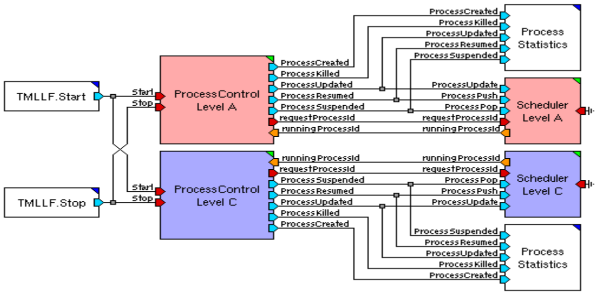

Figure 6: Performance level model of HW and SW of a terrain following system

- Development of a virtual prototype for an optical avionics network [6], Figure 7.

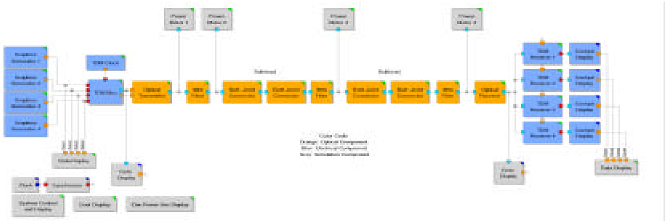

Figure 7: Optical WDM avionics network

- Development of specifications for a global satellite based communication system
- Development of an automated data communication system between aircraft [DARPA project]
- Development of a new design flow for automotive electronics
- Architectural development time for a general purpose processor has been reduced by a factor 7
- The number of design iterations for a wireless communication processor was reduced by a factor 3
- Development time for an embedded SW for satellite communications (ESA PUS standard) was reduced by factor 10

XML, a common description language for diverse specifications and models

Documents currently describe all phases of product development. These are textural requirements and specifications, models, simulations, tests, etc. at different development stages. Standardized descriptions like UML can be used for specifying models, however, they are not sufficient to specify the diverse technologies being used in the design of aerospace systems and their avionics. UML is not sufficient to describe diverse models of executable specifications.

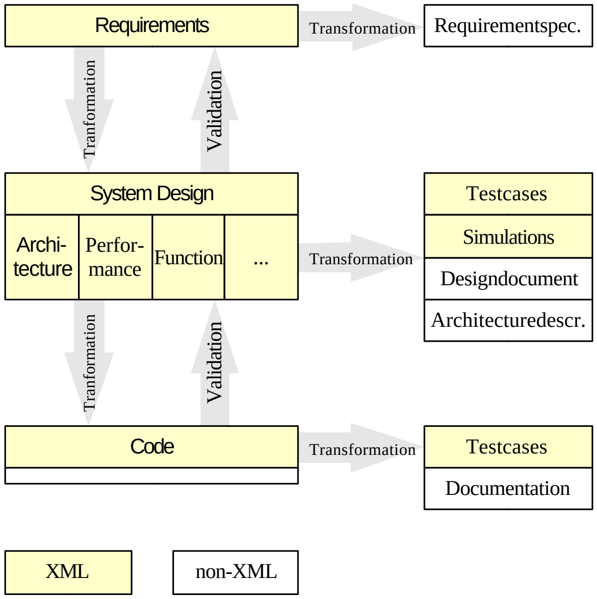

Figure 8: XML descriptions and mappings for a design process

XML is a framework for definition of domain specific description languages and for standardizes description of documents. For the definition and validation of a XML dialect the XML language XML Schema can be used. Because of the formal structure of XML documents, they can be translated in other XML dialects, or other documents, like targeted software. XML is therefore an ideal base for describing all phases of system development. XML documents, models, netlists, schemata can be used by requirement tracking systems, to assure that design rules are followed. XML can be the base between different hardware and software descriptions and as a basis for translations between them. It is therefore increasingly used for development of complex embedded systems in aircraft and spacecraft systems, or other electronic systems like software radio.

Design to specification

Model based design techniques have been developed in the 1980th to integrate the functional design process of systems. With inclusion of high-level architectural/performance abstraction for specification development, an integrated design process has to be developed, including appropriate design software for the design of HW and SW, in order to avoid design gaps, that reduce the gains made with developments at the architectural/ performance level.

Integrating the design process

Figure 9 depicts an integrated design process from mission level requirements to implementation.

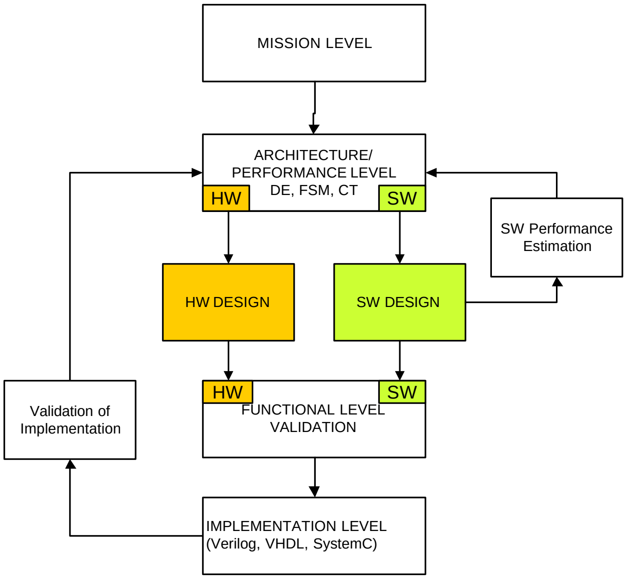

Figure 9: Integrated mission level design process

Having developed models of executable specifications that meet the mission requirements we can determine for which subsystems there are existing of the shelf products, what new HW and new SW has to be developed, including the corresponding specifications. The functional level development can now commence. Functional level models in electronic design for signal processing and control applications typically require Synchronous Data Flow (SDF) and Dynamic Data Flow (DDF) execution models for descriptions of signal processing or control algorithms. Descriptions by FSMs permit to apply formal methods for verification, and significantly reduces the risk of construction errors in HW and SW (more about this in the section on validation and verification). However, it will generally not be possible to validate the SW with the developed HW at the functional level for the overall system.

The SW performance estimator [6] determines the number of cycles, an embedded SW takes to execute on a processor as a function of compiler optimization. This updated value is used in the architecture/performance model to check the design against the specifications of the overall system.

Design software for the mission level design process

The design and analysis software system MLDesigner [7], has been developed to implement the Mission Level Design flow, Figure 10.

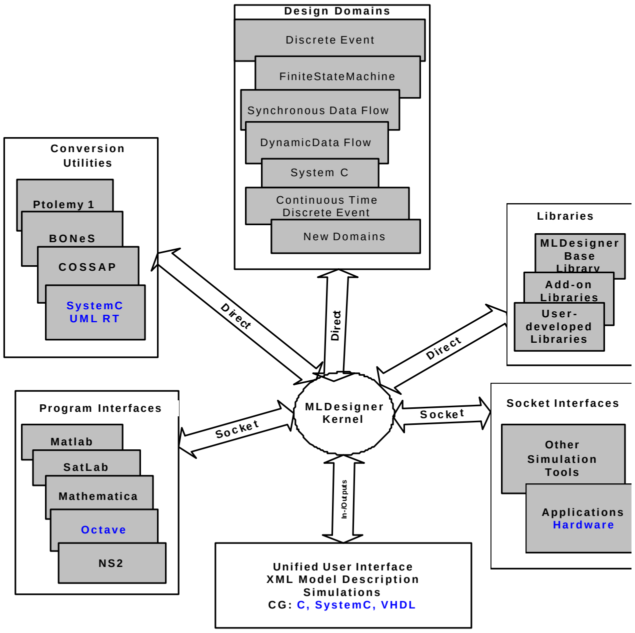

Figure 10: Components of MLDesigner software system

To meet the diverse requirements of an integrated design flow from mission level requirements to implementation, a single kernel, multi-domain software system has been developed. All models are saved in XML. A SystemC execution model permits to validate RTL level implementations against behavioral models of the design. Conversion utilities translate models from 1st generation products, such as Ptolemy, BONeS and COSSAP, and from UML design tools into XML model descriptions of MLDesigner. Interfaces permit co-simulation with the "children" Matlab, SatLab and Octave of the matrix language tool Ctrl-C and with NS2.

Code-generators generate code for HW and embedded SW [8]. Socket interfaces provide the ability to dynamically interface to other design tools, to internet resident applications and to hardware.

MLDesigner is extensible; users can add design domains, and add application libraries to augment the existing libraries. Development of new primitives (basic building blocks) in C, C++ is guided with automated templates. Primitive blocks can be compiled to conceal the implementation detail of the IP. Some of the current applications, including embedded system design, processor and computer architecture performance analysis, SOC co-design, wireless chip, handset and system architectural performance, and production, workflow and design process design and analysis.

SW design to validated specifications

Software development plays an increasing role in design of avionics systems. The development of UML significantly increased quality of the designed software from given specifications. The major challenges is the 60% probability of critical problems due to incorrect specifications, and the developed software is not validated within the networked environment of the overall system. To overcome this problem, software development may be integrated in a mission level design flow, Figure 11, where specifications for the integrated HW/SW-system are validated before SW development and the developed SW is verified together with the HW model before implementation.

An executable and validated model of the architectural/performance level specifications is developed from mission level use cases, environmental models and other specifications. This model already includes a consistent data transport model, that typically takes 80% of the development time when performed at the functional level. Moving the data transport to the front end of the design significantly reduces the development time. We compared this design flow with an UML design flow for the development of an embedded software system for the ESA packet utilization standard (PUS). The UML design flow took 6 month. In the mission-level design flow the data definition and transport problem was solved within one day and the total development took 10 days.

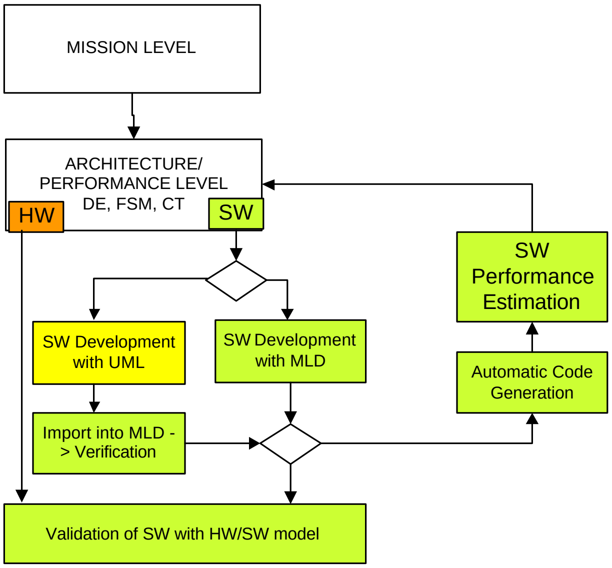
MISSION LEVEL
ARCHITECTURE/ PERFORMANCE LEVEL DE, FSM, CT
HW
SW
SW Performance Estimation
SW Development with UML
SW Development with MLD
Automatic Code Generation
Import into MLD -> Verification
Validation of SW with HW/SW model

Figure 11: Integration of UML software development into a mission level design flow

In the MLDesigner (right hand side) design flow [8], a FSM model of the embedded SW is developed in MLDesigner. It uses the verifier of the MLDesigner FSM for verification and validates the FSM model by simulation in a virtual test environment of the target system. The ANSI C-code generator generates the code for the verified and validated software model for the target system. The execution time of the generated code is determined by the software performance estimator and compared with the assumptions in the architecture/performance model.

In the MLDesigner/UML (left hand side) design flow [10], a FSM model of the SW is developed in the UML tool Real Time Rational Rose®. The sequential RTRR FSM model of the SW, is translated into an MLDesigner parallel Statechart FSM model, required for combination with HW models and saved in XML, Figure 12. The validation is again done in two stages. The validation with the networked environment of the overall system is done at the architectural/ performance level. The validation and verification with the target HW is done at the functional level or hardware-in-loop testing.

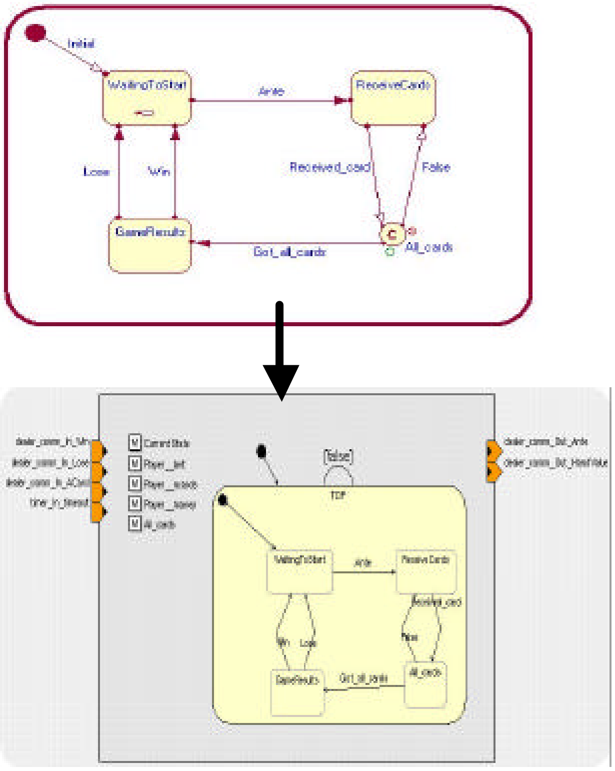

Figure 12: Mapping from RTRR model to MLDesigner model

Validation and verification

Significant progress has been made during the last decade in model checking based on FSM type model descriptions. The basic model can be transformed into edge labeled Mealy machines, state labeled Kripke machines or state and edge labeled Moore machines. Exhaustive model checking can find most construction errors in the designs. However, the complexity cannot exceed certain limits. The main bottlenecks are verification of the networked overall system; particularly performance and load limited characteristics.

The analysis of architectural and performance limitations of the system and complex interactions of subsystems can only be determined with architectural/performance models that include models of hardware, software and the operating system, driven by the test vectors of the missions of the system [9], Figure 13.

Detailed timing analysis of hardware and the use of resources like CPU, bus, memory, battery can be determined for mission level requirements and mission level use cases. As an example, the results from [13], Figure 14, show resource usage by control processing, processor, and channels.

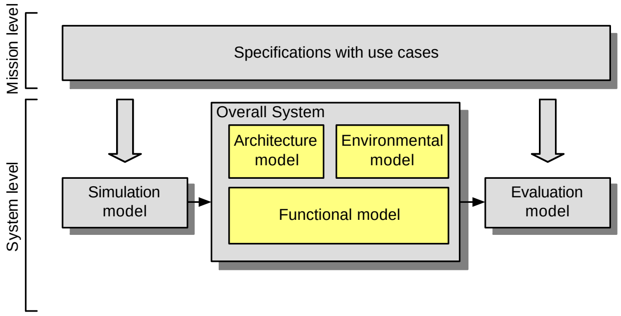

Figure 13: Integrated architectural and functional system analysis architecture

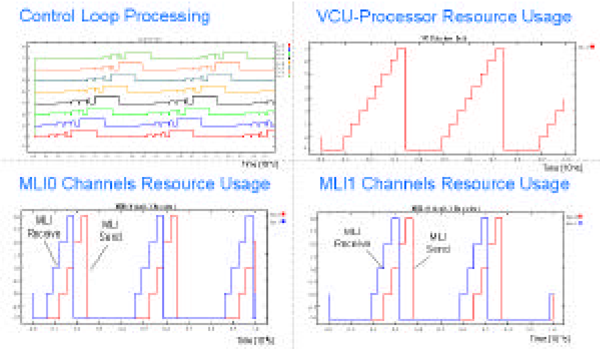

Figure 14: Performance level analysis results

Exhaustive hardware-in-the-loop test of complex networked systems are no longer possible. However, an integrated mission level design flow, that passes the mission level requirements directly to the test vectors of the hardware-in-the-loop test, can concentrate test on where the system is being used and increase the test coverage.

References [11,12] show that the reliability and failure effects analysis and safety analysis of an avionics system can be performed at the architectural/performance level. The model of the avionics system as well as the model of the reliability analysis model are modeled with the same design tool, making a close integration in an integrated mission level design flow possible. Signal flow type models as well as event driven failure models have been analyzed. The latter requiring the development of mapping of development models into event driven failure mode models, in order to integrate this approach without translation errors due to manual translation.

Optimizing the design process

Research has started to model and simulate quality processes and design processes in the design of complex hardware and software. It is expected that this research will increase the quality in development of embedded system by a factor 100. First results look very promising. However, this research is concentrated currently at the development process of electronic control units. It is too early to draw general conclusions from this research.

Conclusions

This paper shows some recent developments in system development methodologies and design software to solve the complexity problem in avionics design. It is shown how modeling and simulation at the architectural/performance level permits to develop executable specifications, significantly reducing the probability of critical design errors and reducing the number of design iterations and hence reducing cost of development. An integrated design process is described that integrates the design from mission level requirements to hardware/software implementation and verification.

Critical issues are standardization of models at the architectural performance level, validation at the architectural/performance level. To be solved is the problem that electronic hardware becomes obsolete much faster than software and will not be available for the lifetime of an aerospace system. How can hardware be replaced without changing the embedded software.

REFERENCES

[1] Salzwedel, Horst, James H. Vincent, 1984, Modeling Identification and Control of Flexible Aircraft, AFWAL-TR-84-3032.

[2] Salzwedel, Horst, R. Calhoun, and Lt. P. Murdock, May 1985,Unbiased Transfer Alignment Filter Design for Air Launched Weapons, National Aerospace and Electronics Conference, Dayton, Ohio.

[3] Schorcht, Gunar, July 2000, Design of Integrated Mobile Communication Systems at Mission Level, Dissertation at Ilmenau Technical University.

[4] Schorcht, Gunar, Ian Troxel, Keyvan Farhangian, Peter Unger, Daniel Zinn, Colin Mick, Alan George, and Horst Salzwedel, 2003, System-Level Modeling with MLDesigner, Proc. of 11th IEEE/ACM International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems (MASCOTS), Orlando, FL, October 12-15, 2003.

[5] Ian Troxel, W. Chris Catoe, Ramesh Balasubramanian, Alan D George, HCS Research Lab., Jeremy D. Wills, Sung J. Kim, Gregory A. Arundale, Rockwell Collins, John L. Meier, Boeing, Colin K. Mick, MLDesign, November 2004, LION: Virtual Prototyping for Advanced Optical Military Networks, MILCOM 2004, Monterey, CA.

[6] Lohfelder, Thomas, Holger Rath, Horst Salzwedel, 27-30 September 2004, Software Performance Estimation for a Mission Level Design Flow, 49th International Scientific Colloquium, Ilmenau.

[7] MLDesigner® Manual v2.5, 2004, http://www.mldesigner.com

[8] Rath, Holger ,Horst Salzwedel, 27-30 September 2004, ANSI C Code Generation for MLDesigner Finite State Machines, 49th International Scientific Colloquium, Ilmenau.

[9] Paluch, Nils, Achim Schönhoff, EADS, 27-30 September 2004, Anwendung von C-Design für verteilte Echtzeitsysteme, 49th International Scientific Colloquium, Ilmenau.

[10] Baumann, Tommy, Horst Salzwedel, 27-30 September 2004, Integrating UML Software Models in an Integrated Mission Level Design Flow for the Design of Hardware and Software, 49th International Scientific Colloquium, Ilmenau.

[11] Nicol, Davis M., Daniel L. Palumbo (NASA), Michael L. Ulrey (Boeing), 1995, A Graphical Model-Based Reliability Estimation Tool and Failure Mode & Effects Simulator, IEEE 1995 Proceedings Annual Reliability and Maintainability Symposium.

[12] Ulrea, Michael L. (Boeing), Daniel L. Palumbo (NASA), David M. Nicol (College of William and

Mary, Williamsburg), 1996, Case Study: Safety Analysis of the NASA/Boeing Fly-By-Light Airplane Using a New Reliability Tool, IEEE 1996 Proceedings Annual Reliability and Maintainability Symposium.

[13] Salzwedel, Horst, Matthias Zens, 24-30 September 2004, Development of Embedded Automotive Electronics at Architectural/ Performance Level, 49th International Scientific Colloquium, Ilmenau.