Mission Level System Design using UML 2.0

Tommy Baumann, Prof. Horst Salzwedel
tommy.baumann@tu-ilmenau.de

August 4, 2005

Abstract

Complex 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 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 in combination with how developments from UML 2.0 can be used for complex system design and what extensions are neccessary.

1 The challenge of complexity

Aerospace systems are characterized by architectural complexity, dynamic interaction between subsystems, and complex interdisciplinary functionality. The challenges of designing these complex 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, 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, 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 [Salzwedel u. Vincent 1984] 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 [Salzwedel u. a. 1985]. With the proliferation of electronics in nearly all type of engineering systems, and the rapid increase of the complexity of electronics, the major challenge of

system development has become the gap between system design and networked electronics 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 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. 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 the Saab 39 and 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 two spacecrafts to Mars failed because of a mix-up between units used by different design teams
- the Boeing 702 series of spacecraft exhibit major problems. The development team cannot predict when the problem will be fixed
- 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 German automated toll collection system has been delayed because very little worked when independently developed subcomponents were put together.

All these examples exhibit similar problems in the design flow. The subsystems are designed from written specifications. When they are put together, the system does not work. The problems are fixed on a local level. Despite very high validation and test cost critical problems remain.
A recent ESPITI study [Schienmann 2002] 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 is 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.

2 Mission Level Design

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 are validated against these specifications. 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. Even with our improved computational capabilities, it is no longer possible to simulate the HW/SW implementation of components and subcomponents against the mission requirements. This led to failures like the first Ariane 5 and the 1999 missions to Mars.
Mission Level Design (MLD) [Schorcht 2000] is a hierarchical approach that generalizes the design approach for deep space missions:

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

Experience shows that 80-90% of the design decisions can be made at stage two of this approach. In a Mission Level Design flow, the top-level architecture and performance requirements are modeled using discrete event (DE) models and finite state machines (FSM). Initial HW models abstract resources such as memory, buses, 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 [Schorcht u. a. 2003]:

- 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 and architecture dependent interactions
- dealing with use cases and mission scenarios

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 systems total cost, performance and time to market.
Functional level models in electronic design for signal processing and control applications typically also require Synchronous Data Flow (SDF) and Dynamic Data Flow (DDF) execution models for descriptions of signal processing algorithms. Descriptions by FSM's permit to apply formal methods for verification, and significantly reduce the risk of construction errors.
The SW performance estimator [Lohfelder u. a. 2004] 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.
The design and analysis software system MLDesigner [MLDe 2005], has been developed to implement the Mission Level Design flow, which is pictured in figure 1. Besides the execution models mentioned above, it also includes a SystemC execution model, in order 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 into XML model descriptions of MLDesigner. Interfaces permit co-simulation with thechildrenMatlab, SatLab and Octave of the matrix language tool Ctrl-C and with Mathematica. Code-generators generate code for HW and embedded SW [Rath u. Salzwedel 2004]. Socket interfaces provide the ability to dynamically interface to other design tools, to internet resident applications and to hardware. Applications include 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.

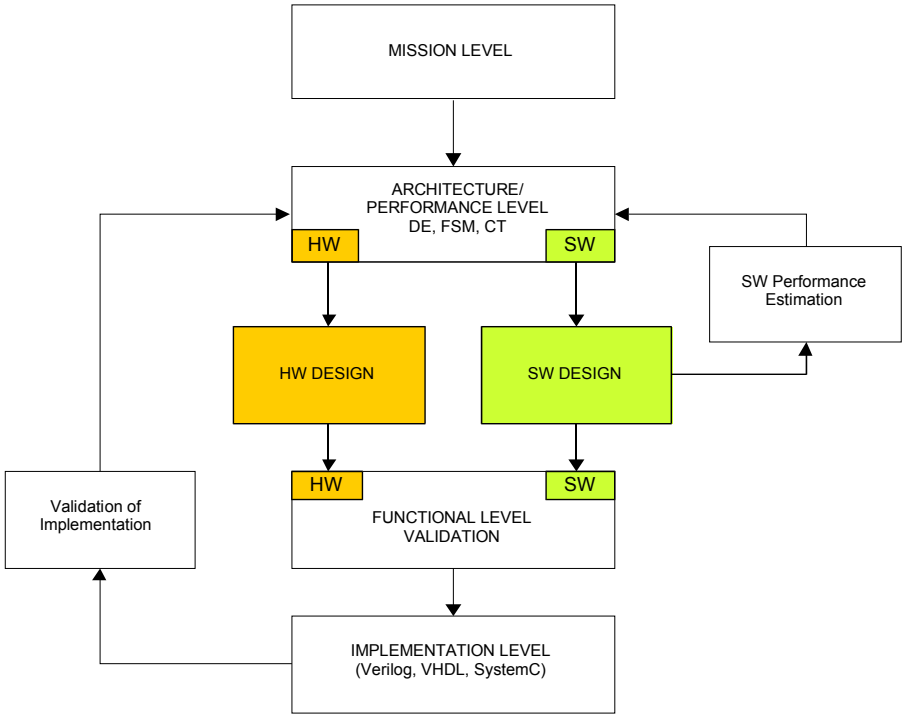

Figure 1: Mission Level Design Flow

3 UML 2.0

The Unified Modeling Languages (UML) includes textual and graphical description languages for specification, visualization, construction and documentation of systems and languages. Version 1.x had some weaknesses concerning the development support for modelling real time systems with component-like architectures. Furthermore the focus lay rather on software development from given specifications. While the Object Management Group (OMG) made only small changes to UML during the last years, with version 2.0 [OMG 2004] a major steps toward overcoming the weaknesses have been undertaken. Many features are taken from UML 1.x profile UML-RT, which is used by the CASE-tool Rational Rose Realtime® [IBM 2003] to develop software for embedded systems (like the capsule mechanism with structure diagram and state diagram). The following list shows fundamental improvements of the new standard UML 2.0:

- new meta model that avoids redundances and inconsistences
- definition of a domain model for model execution
- transformations between models and diagrams
- modelling of real time systems by new description elements with extended semantics, e.g. ports for the improved description of interfaces or parts as a fully hierarchical component concept
- improved description of sequences and their involved information elements
- the modelling of component architectures by improved constructs concerning packaging and communication

4 Mission Level System Design using UML 2.0

This chapter clarifies which diagrams and models from UML are used and can adapted for development of complex systems based on the Mission Level Design Flow in figure 1. The

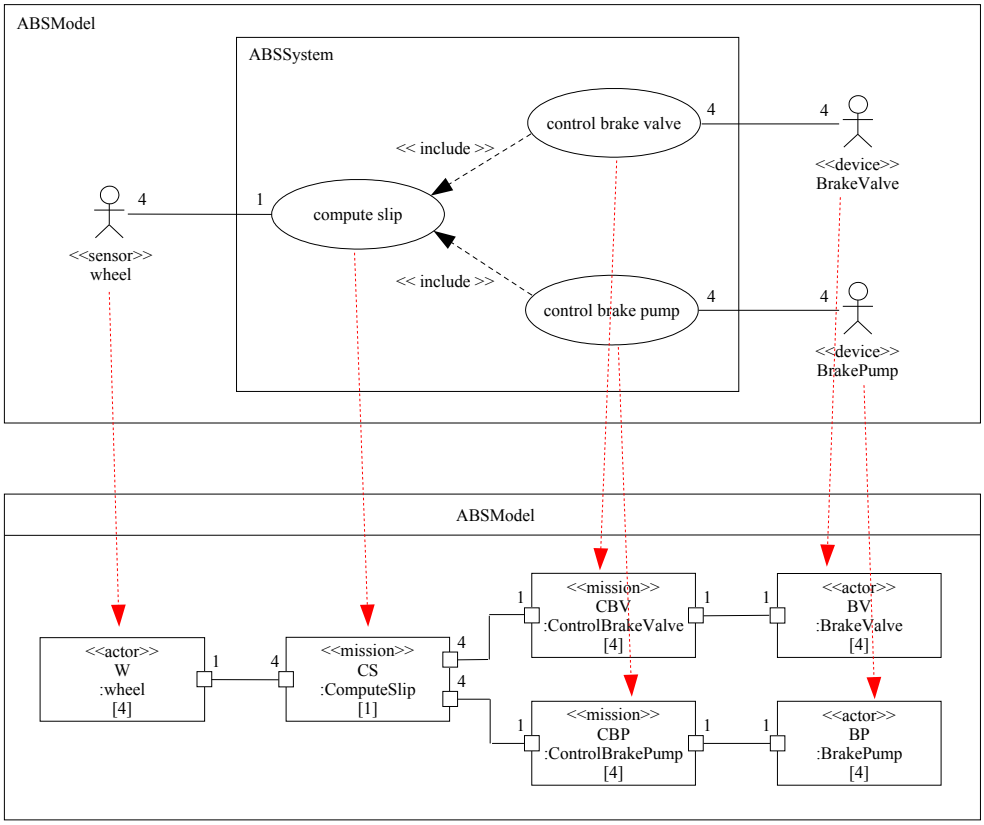

Figure 2: equivalence of use cases and missions

design flow handles architectural complexity, dynamic interaction between subsystems and complex functionality of systems. Because of the increasing amount of system parts and dependencies between, it is essential for complex system designs that each part specification and the complete specification together is executable and verifiably. Thru the increased system complexity Hardware in the Loop validation and verification is no longer feasable. It follows that every model or diagram description used by the presented design flow has to be executable. The Mission Level Design Flow is applied by the system design tool MLDesigner [MLDe 2005] to handle the elucidated complexity problem. Hereby the major challenges of high probability of critical problems due to specifications and the not validated SW against the use of it together with HW are respected (integrated design of HW and SW).
The UML alone is insufficient to solve the problems of system specification and verification, because it is only a standardized textual and graphical description language. It has to be bound into a design flow, made executable and adapted to requirements of it. This is achieved thru the selection of relevant executable diagrams, the relations between them and the definiton of extensions. Such extensions are defined by UML profiles consisting of stereotypes, tagged values and constraints.
The hierarchical top-down approach of the Mission Level Design Flow starts with the requirement analysis on *Mission Level*. This is done by the usage of missions, which are equivalent to use cases to defining single requirements. They are parts of a Composite Structure Diagram (figure 2 below), which allows the decomposition of the system. Hereby missions use the sterotype $<< mission >>$, actors the stereotype $<< actor >>$. Such stereotyped parts are used on all abstraction levels, because each can have its own specification and therefore own missions. This means that executable use cases on different abstraction levels are supported. UML Use Case Diagrams are useful to sketch out some ideas, but not for definiton of verifiably executable specifications. Figure 2 shows that the adapted Composite Structure Diagram can express the same requirements.

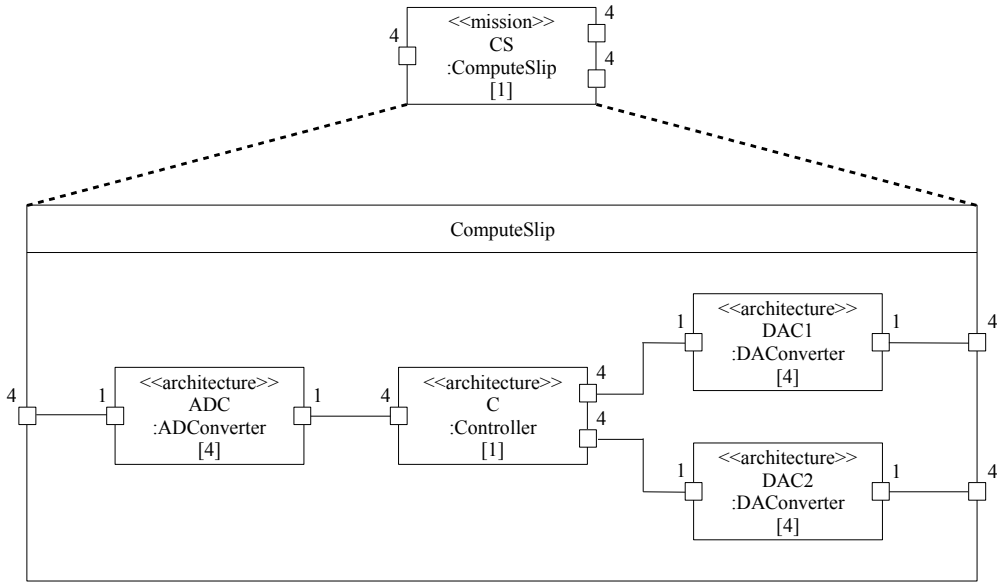

Figure 3: mission refinement into architecture parts

Composite Structure Diagram are new in UML 2.0 and consist of parts, ports and relations. They are suitable to arrange the system into parts to represent different levels of abstraction hierarchically. To execute such diagrams they have to be extended with execution domain information (DE,SDF, CT). For example the domain DE (discrete event) stands for an event based compution model. This means that parts interacting thru events, whereby each event owns a value and a time stamp. The event occurence is organized by a scheduler depending on time stamp. Combining diagrams from different domains permits to develop multidomain systems. Other neccessary extension are stereotypes to specify different part types ($<<$ *mission* $>>$, $<<$ *architecture* $>>$) and attributes for data retention. Attributes are used as local or global (link to other abstraction levels) memories, events or resources.
Composite Structure Diagrams are also used at *Architecture/Performance Level* and *Functional Level*. At *Architecture Level* distributed components of the real architecture are modeled as parts with stereotype $<<$ *architecture* $>>$. Figure 3 shows the refinement of a mission into such parts. Each of them has to be further refined into functional parts. *Performance Level* means that parts do not contain functional information but performance values like execution time, memory size, bandwidth and error rates. A performance level model that meets the mission requirements is an executable specification. At *Functional Level*, system functionality is modeled (stereotype $<<$ *function* $>>$) and can be further decomposed into parts or directly specified via source code.
Parts can be mission, actor, architectural component or capsuled functions. They can be realized thru source code or other diagrams, like Composite Structure Diagrams, State Diagrams, Activity Diagrams or Sequence Diagrams. The new classifier concept of UML 2.0 defines the usage of composite structures and behaviors (package *InternalStructures* and *Common Behaviors*). Different diagrams can be used in combination. For example an Activity Diagram can be part of a Composite Structure Diagram (figure 4). Activity Diagrams are suitable to design control and data flows of dynamic function sequences using petri network execution model. Such embeddings are also possible in reverse. As an example, Composite Structure Diagram can be assigned to an action of an Activity Diagram.
Information exchange between system parts with the classifier environment is defined by connected ports (package *Ports*). Each port stands for an interaction point of an owning classifier and can require or provide interfaces. This interfaces are realized by port types, which are also classifiers. When an instance of a owning classifier is created, instances corresponding to each of its ports are created and held by this classifier. Since ports are typed

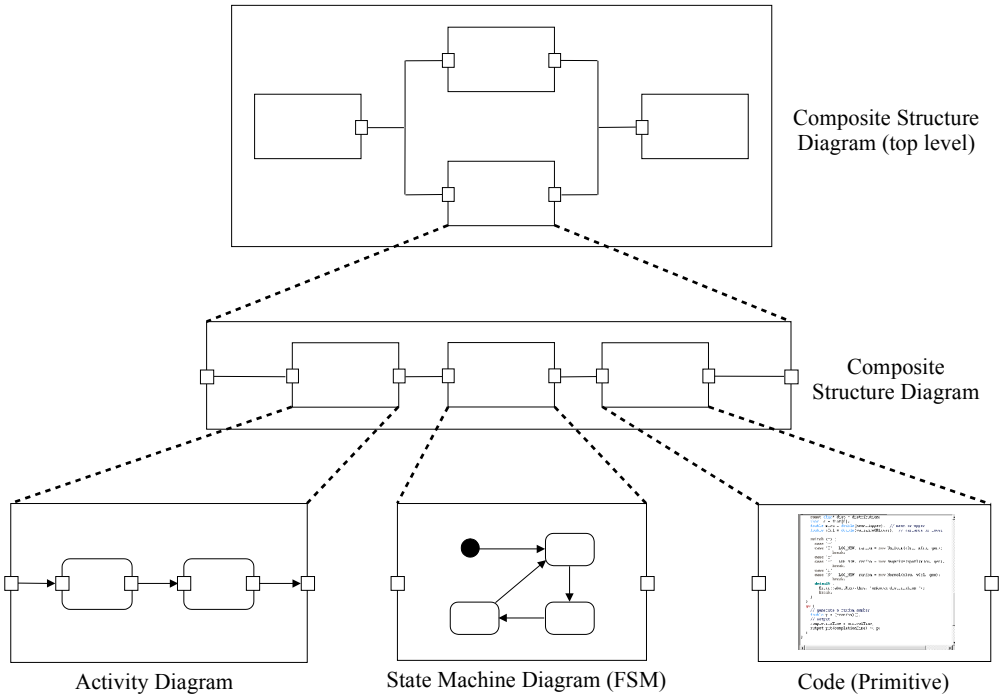

Figure 4: diagram embeddings survey

by classifiers, they can optionally define own behavior (e.g. FIFO). Ports are compatible thru the model hierarchie when their types are compatible. The UML profile mechanism is applied to define standard types.
At *Implementation Level* modeled diagrams are used for code generation, e.g. C, C++, Verilog, SystemC or VHDL. Hereby stereotyped architecture parts define the type of source code representation and target architecture. The generated model can be executed for validation against the specification. To debug temporal system information, Sequence Diagrams are used. They show snapshots of selected functions and temporal interactions of involved parts.

5 Examples

5.1 Satellite Data Handling System (DHS)

DHS controls the payload of satellites and sends telemetrie data to ground stations. For the development of a data handling system in a satellite, the ESA Packet Utilization Standard (PUS) was to be used. PUS defines the interface between services and application units in a satellite. Two approaches were used for the design. In the first approach the waterfall like standard UML design methodology was used (requirement analysis, design, implementation, deployment/testing). After six months development time some problems still remained and it was not clear how long it would have taken to fix them. In the second approach, a mission level model of the overall system was developed (figure 5), consisting of two ground stations and a satellite. The satellite contained parts that used PUS. The first ground station generates commands in form of PUS data structures. These are send to the satellite by telemetry. The packets are picked up by the COM-subsystem of the satellite and distributed to all the satellite parts in PUS format. These subcomponents interpret the data according to PUS. Response and report packets are send back to the second ground station for representation. The mission parts of the top level system are refined thru further diagrams according the Mission Level Design Flow. For example the part Satellite - Data Handling System in figure 5 is refined into architectural parts, e.g. UpLink, DownLink, and some functional parts, e.g.

Figure 5: Satellite Data Handling System

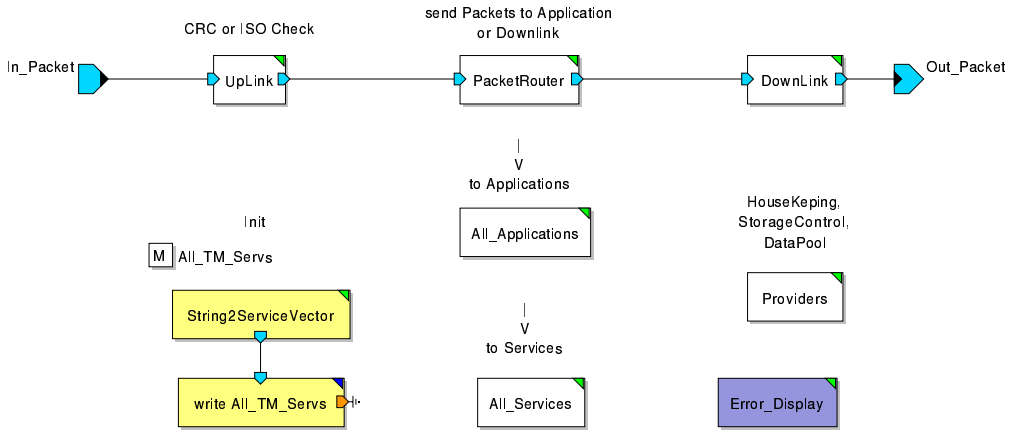

Figure 6: refinment of part Satellite - Data Handling Sytsem

String2ServiceVector (figure 6). A validated model was developed within four days. The standardized PUS data structures were defined in MLDesigner within half a day. The model was developed within a day and iterated on for three days.
The analysis of the huge differences in development times showed that a main time sink in the waterfall approach was the development of the data transport system. This was taken care of by the tool in the second approach. Further in the second approach only that level of architectural and functional abstraction was modeled, that was neccessary for design and validation.

5.2 Electronic Valve Application

In [Salzwedel u. Zens 2004] a Mission Level Design approach was investigated for development of architectures for electronic valve applications. Camless engines using electromagnetic valve actuators have demonstrated fuel savings of up to 20%. Despite the large amount of electronics required to implement electronic vales, they are not permitted to be more expensive than current generation valves, driven by camshafts. The reliability also must be as high as that of mechanical valves. Optimizing costs and reliability of electronics means optimizing architecture for the mission of electromagnetic valve application.
In order to achieve confidence in the design process, it is embedded in a validation environment, consisting of exchangeable parts for architectural, environmental and functional models of hardware and software. Missions are driving simulation model and evaluation model, in order to asure that all design decisions are validated against the mission of the overall system. Architecture, functional and environmental parts of the model are mapped into models of different execution domains of MLDesigner. Because of the big difference be-

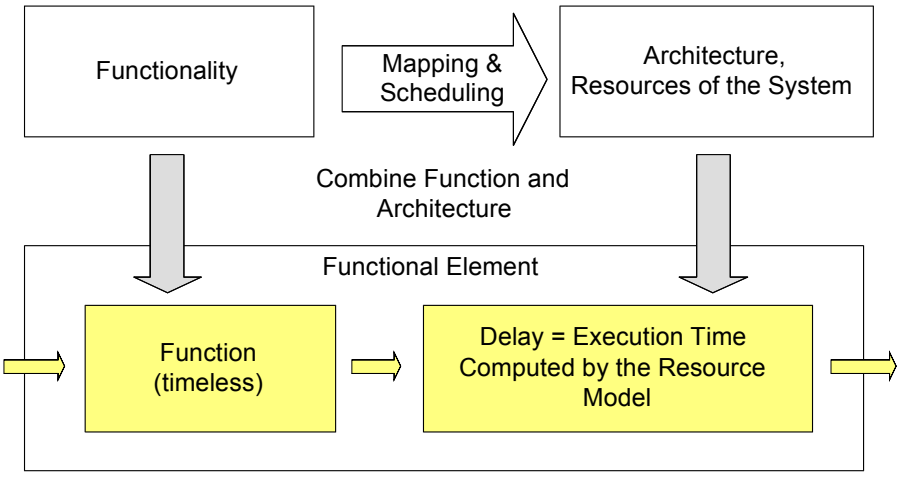

Figure 7: architecture mapping

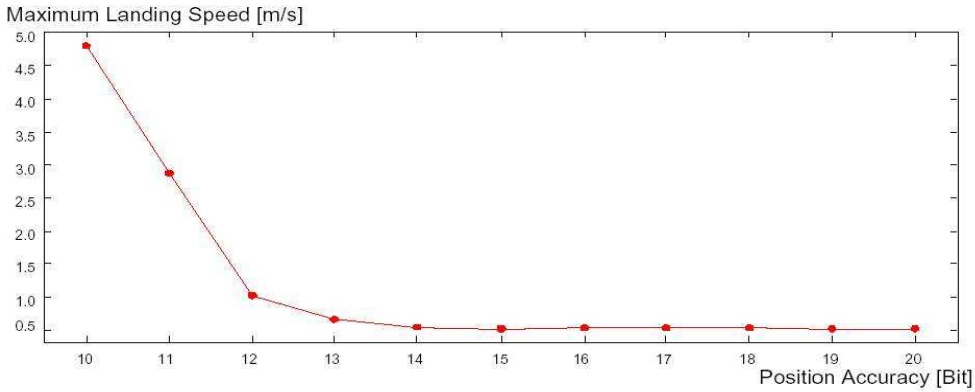

Figure 8: result from reliability analysis

tween execution times for simulating architectural components and functional components, the strategy was used to perform analysis and design as much as possible at the performance level and include functional components only when necessary. Thru such a performance evaluation decisions about optimization of bus architecture, memory architecture and transport protocols can be determined.
The functionality of system components consist of functional algorithms, resources and needed time delays for execution. To connect these, the algorithm is mapped into a timeless function and into the system architecture with resources (7). A Instruction Resource Model (IRM) is used to map the function of a data flow model to the resource that will execute it and schedules it, in order that it will execute at the right time. Execution requests of a data flow model are send to a central server, which sends an achnoledgement to the function when the resource has executed the function. The algorithmic execution of the function is executed in the function of the data flow graph itself. The IRM therefore performes the function of a real time operating system (RTOS).
For the functional evaluation of the electronic valve control system, the effects of sampling time and time delays on the system were investigated. The main measure of acceptance was the maximum landing speed of the valve. The accuracy for current, position and PWM and

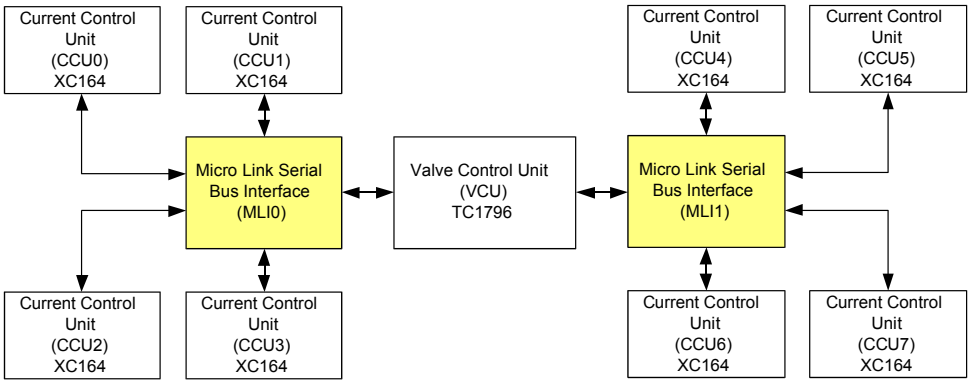

Figure 9: architecture partitioning

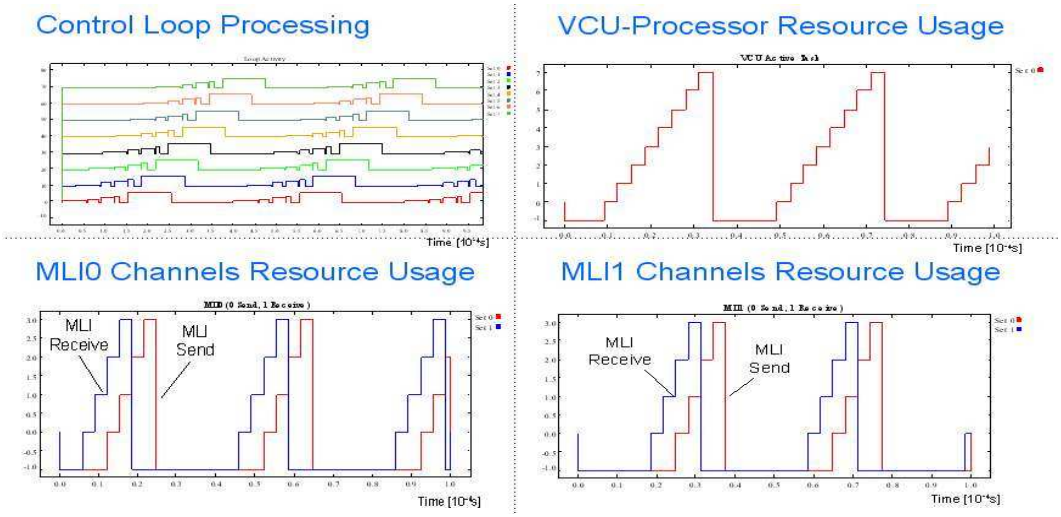

Figure 10: resource usage

ist impact on maximum landing speed were determined. For reliability analysis, the error rates of data transport on the busses were investigated. The graph shows that at position has to be measured to at least 14 bits (figure 8). For the architectural evaluation, the timing characteristics of the electronics are analyzed and validated. This included the analysis of all the control loops, determination of the level of usage of the resources, and analysis of the distribution of tasks of individual control loops over the resources of centralized and decentralized architectures. For decentralized partitioning, 1 Infineon XC164 processor was selected for voltage control, A/D, and D/A conversion of 1 valve, and an Infineon TC1796 processor for outer loop position control of 8 valves (figure 9). The simulation was performed only with the architectural performance model since the functional evaluation was done before, significantly shortening the simulation times. The simulation results of this architecture show the usage of the resources: control loop processing, VCU, MLI0 channel and MLI1 channel (figure 10).
This example shows that complex electronics can be developed, analyzed and optimized by an integrated simulation containing mission, architecture, and function of hardware and software. This methodology automatically validates the design against mission requirements and significantly reduces requirements for redesign after test and hence reduces development time.

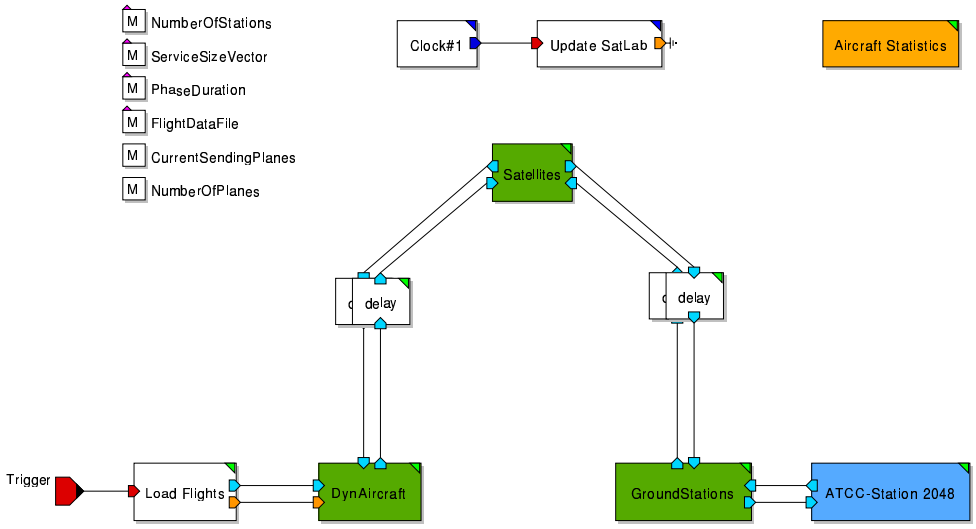

Figure 11: Satellite Data Traffic System

5.3 Satellite Data Traffic System

Air traffic in europe increases by 4% every year. To handle this problem new systems for traffic control are needed. With help of the satellite based communication system Air Traffic Management (ATM) a more efficient airspace usage and earlier danger detection is possible. Hereby aircrafts and control centers communicating thru the Aeronautical Telecommunication Network (ATN), which sends data packets via satellites. To dimension such satellite systems, simulations about the occuring data traffic thru satellites are neccessary. In cooperation with EADS Astrium, a model of an aeronautical data traffic system was developed as an executable specification for the system and for specifying the requirements for the communication satellites.
The systems was modeled in MLDesigner. It includes models of different data packets and real data of aircraft routes, satellites and ground stations. These are taken from SatLab. Figure 11 shows the top level system with mission parts (e.g. LoadFlight), function parts (e.g. delay) and architecture parts (e.g. ground stations). The mission scenario was defined by loading flight routes and schedules from OAG's Worldwide Flight Database [OAG 2003]. While executing the system aircrafts are created dynamically and held by part DynAircraft. After that an aircraft sends packets to a satellite, which sends them to the ground stations. The ground stations sends them to all control centers. The corresponding control center receives the packets, all other ignoring. In the result packet traffic informations from satellites, ground stations and aircrafts were collected and are the basis for sizing the communication satellite system. An extension of this work for the analysis of capacity dimensioning for future long-haul inflight entertainment streaming services can be found in reference [Unger u. Salzwedel 2004].

6 Summary

It was shown that the UML 2.0 can be adapted for the design of complex systems with the Mission Level Design Flow. Used diagrams have to made executable and combinable with each other, e.g. Composite Structure Diagram, State Diagram and Activity Diagram. Hereby the Composite Structure Diagram has been found to be particularly useful, since it permits system decomposition and modeling of different abstraction levels hierarchically. Most of the descripted extension mechanisms are available in MLDesigner, others are added in the ongoing research. UML 2.0 is not a graphical interface to formal languages, it is rather a common base notation for next generation system design tools. The examples show the

great advanteages of integrated spiral design flows (Mission Level Design Flow) compared with often used waterfall design flows.
A definiton of profile UML-MissionLevel (UML-ML) is currently being developed. It is to investigate, which parameters, constraints and tagged values are neccessary for different stereotyped parts. Protocols have to be defined to descripe multiple signal interfaces. Furthermore the data transport mechanism between parts has to be optimized. For instance the usage of references and Copy on Write to accelerate simulations.

References

[IBM 2003]
IBM Inc.: *Rational Rose Realtime documentation.* 2003. – http://www.ibm.com/software/rational

[Lohfelder u. a. 2004]
Lohfelder, Thomas ; Rath, Holger ; Salzwedel, Horst: Software Performance Estimation for a Mission Level Design Flow / 49. Internationales Wissenschaftliches Kolloquium IWK'2004, Ilmenau. 2004. – Proceeding

[MLDe 2005]
MLDesign Technologies Inc.: *MLDesigner documentation.* 2005. – http://www.mldesigner.com

[OAG 2003]
OAG: OAG Database. 2003. – Proceeding. http://www.oag.com

[OMG 2004]
OMG: UML 2.0 Superstructure Specification. 2004. – Forschungsbericht. http://www.omg.org/cgi-bin/doc?ptc/2004-10-02

[Rath u. Salzwedel 2004]
Rath, Holger ; Salzwedel, Horst: ANSI C Code Generation for MLDesigner Finite State Machines / 49. Internationales Wissenschaftliches Kolloquium IWK'2004, Ilmenau. 2004. – Proceeding

[Salzwedel u. a. 1985]
Salzwedel, Horst ; Calhoun, R. ; Murdock, Lt. P.: Unbiased Transfer Alignment Filter Design for Air Launched Weapons / National Aerospace and Electronics Conference, Dayton, Ohio. 1985. – Proceeding

[Salzwedel u. Vincent 1984]
Salzwedel, Horst ; Vincent, James H.: Modeling Identification and Control of Flexible Aircraft / AFWAL, TR-84-3032. 1984. – Proceeding

[Salzwedel u. Zens 2004]
Salzwedel, Horst ; Zens, Matthias: Development of embedded automotive electronics at architectural/performance level / 49. Internationales Wissenschaftliches Kolloquium IWK'2004, Ilmenau. 2004. – Proceeding

[Schienmann 2002]
Schienmann, Bruno: *Kontinuierliches Anforderungsmanagement.* Addison-Wesley, 2002. – ISBN 3–8273–17–87–8

[Schorcht 2000]
Schorcht, Gunar: *Entwurf integrierter Mobilkommunikationssysteme auf Missionsebene.* Logos Verlag, 2000. – ISBN 3–89722–462–3

[Schorcht u. a. 2003]
Schorcht, Gunar ; Troxel, Ian ; Farhangian, Keyvan ; Unger, Peter ; Zinn, Daniel ; Mick, Colin K. ; George, Alan ; Salzwedel, Horst: System-level simulation modeling with MLDesigner. In: *Modeling, Analysis and Simulation of Computer Telecommunications Systems 2003. MASCOTS 2003. 11th IEEE/ACM International Symposium*, 2003. – ISSN 1526–7539, S. 207–212

[Unger u. Salzwedel 2004]
Unger, Peter ; Salzwedel, Horst: Capacity Dimensioning for Long-Haul Inflight Entertainment Streaming Services / 49. Internationales Wissenschaftliches Kolloquium IWK'2004, Ilmenau. 2004. – Proceeding