MOVING DESIGN AUTOMATION OF NETWORKED SYSTEMS TO EARLY VEHICLE LEVEL DESIGN STAGES

Horst Salzwedel, TU Ilmenau, Nils Fischer, Gunar Schorcht, Mission Level Design GmbH

Abstract

Networked systems in automobiles and aircraft are designed by groups of people, departments and/or companies of different expertise. Commonly used design methodologies in complex system design partition the work at high level of uncertainty into teams of specialists. These teams have to make assumptions and design decisions without being able to evaluate the impact of their decisions on the overall design. Many design decisions have to be made during integration. This design methodology does not permit optimization at vehicle level and results in very low probabilities of not having critical design errors [1]. Hence it causes high integration costs and redesigns.

This paper shows why the current design methodologies cause unverified designs and critical errors, and how system level design automation technologies can be extended to early vehicle level design stages for verifiable vehicle level executable specifications and architectural optimization at vehicle level. Using extensions of the integrated design tool MLDesigner©, the developed design methodology is demonstrated for examples executable specifications of vehicle power management and power train. In an example of vehicle level architecture optimization cable length is reduced by two third and system availability improved by several orders of magnitude.

Introduction

Complex systems like networked systems in vehicles (aircraft, spacecraft, automobiles, ships, trains, autonomous systems), networks between vehicles, IT systems, communication or organizational systems can only be developed by groups of people, departments and/or companies. To meet time to market requirements, the system design is partitioned into subsystems and subsystem design is distributed to specialist teams before uncertainties about individual subsystems and interactions between subsystems has been resolved. The individual engineers make assumptions based on experience on what worked before. It is not possible for these engineers to understand the impact of their decisions on the particular system being designed. Since the models passed on for system integration do not include the model uncertainties, the uncertainty of the integrated system is not determined. The integrated system model or design specification can therefore not be validated. John Hynes: “We Don’t do design correctly!”[1] The likelihood that this design approach will work without critical problems is less than 4% [2]. Very time consuming and expensive integration processes try to resolve conflicts and critical performance issues.

In this paper a system design methodology is developed based on executable design flows. Bounded and statistical uncertainties of early design stages are included and bounds on integrated designs based on uncertainties at different design stages are determined. Examples will demonstrate the new design methodology.

The new design methodology not only permits to treat uncertainties in early design stages, but permit optimizing the design, as well as the design flow for different criteria like quality in design and design cost. Design changes can easily be implemented and design tracking automated.

In order to solve this complex system design problem, we determine first where these critical errors occur, secondly why they do occur and thirdly develop methods to overcome the complex system design problem.

Where do critical design errors occur?

Figure 1 [3] depicts the probability of critical problems as a function of phases of the design process. The probability of critical problems is very high in early design stages and is low in late design stages. More than 65% of critical design errors are caused by poor design specifications. About 25% of critical design errors are caused by mistakes during design, and 15% during system implementation.

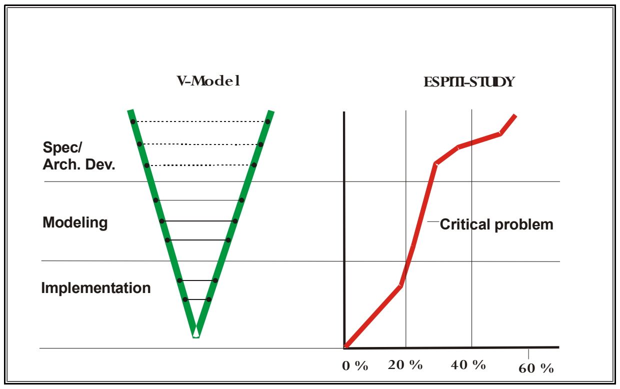

Figure 1: Critical problems in design of complex systems [3]

Why doesn't the current design methodology work?

The main reason for the high probability of critical errors during system specification is that only written specifications are developed. These cannot be validated at system level. The high uncertainty about a system at this stage of development causes a high probability of errors. Let us investigate this further by taking a look at product uncertainty.

Information, knowledge and uncertainty about a product changes significantly during product life time. This knowledge about the product includes knowledge about the product itself, knowledge about the intended use, and knowledge about the environment in which the product is to be used. This uncertainty is bounded and may sometimes be described by bounded parameter sets, bounded statistical distributions, and/or bounded functional behavior described by e.g. H^{∞} norms for continuous systems.

Figure 2 shows a typical behavior of the mean of product uncertainty for an automotive subsystem development. Overlaid over the product uncertainty is the expenditure for the development. At the start of a development project we don't know anything about the product. The initial product uncertainty is $U_0 = 100\,\%$. During development the knowledge about the product increases with a given learning rate, reducing product uncertainty. Each of the development phases, i, has it's own learning rate, l_i. The learning curve can be described by an exponential function,

$$U_{D_i}(t) = U_{0_i} e^{-l_i(t - t_{0_i})} \qquad t_{0_i} < t < t_{D_i}, \qquad l_1 < l_2 < l_3$$

where t_{0_i} and t_{D_i} are initial and final times of development phase i, respectively.

At the end of the development phase, the product is delivered to customers, who will find additional insufficiencies that are reported to support and corrected by the development team. The learning effect due to customer testing and support can be described by,

$$L_S(t) \;=\; \begin{Bmatrix} 0 & \forall \;\; t \le t_D \\ L_{S0}(1 - e^{-l(t - t_D)}) & \forall \;\; t > t_D \end{Bmatrix}$$

However information about a product is also lost during lifetime of a product. The reasons are that people leave development teams, are reassigned, or just not remembering all the decisions which went into the design of a product. Additionally, as time progresses, products are often used for applications that were not considered during the development phase. These effects impose additional uncertainty on the product, which can be approximated by,

$$U_L \;=\; U_{L0} e^{Lt^2}$$

As development of products with long life times like the Space Shuttle showed, people even have to be called back from retirement in order to overcome this loss of product information.

The uncertainty about a product can then be expressed by,

$$U_P(t) \;=\; U_D(t) \;+\; U_L(t) \;-\; L_S(t)$$

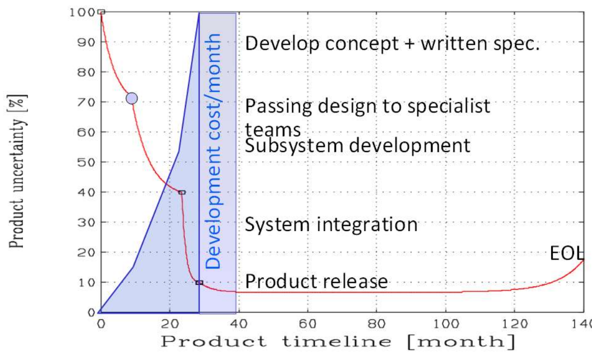

Figure 2: Product uncertainty

Comparing Figure 1 with Figure 2, we observe that observed occurrence of critical design errors and product uncertainty are highly correlated. We may conclude that critical design errors are primarily due to information uncertainty about a product and design processes that do not consider this uncertainty in the right way and do not validate and verify the system considering this uncertainty.

In addition to product uncertainty we have to consider the risk of development. This is the expenditure for development which is at risk. The capital at risk is,

$$\boldsymbol{R} = \int U\, C_{Month}\, dt = 296.1 * maxC_{Month} + Overrun$$

where U is the product uncertainty, C_{Month} is the cost for development per month, $maxC_{Month}$ is the scaling factor for the cost curve shown, and $Overrun$ is the unplanned development cost when the project cannot be completed in time.

For the shown development process we see that many critical problems are only solved when the development expenditure rate is at a maximum. The development process is therefore a high risk development process.

Complex System Design

Complex systems are always designed and developed by groups of people, departments and/or companies. The system is partitioned into subsystems and subsystems are assigned to different groups. Each group typically has unique knowledge in a particular field and will use their own methodology, models and software for the design of their assigned component. Some will use different software systems.

For design, validation and verification design groups will consider a range of possible parameters and use approximate models for analysis. Different designers/developers will exchange information about their components to other designers according to design or process graphs, Figure 3. The exchanged information typically will not include all assumptions and ranges of uncertainty considered in the design of components. The reason is that some of the uncertainties are very specific to the field of knowledge used for a component and other designers will not understand this information nor will they be able to properly evaluate potential interactions between components in different subsystems. Other reasons include that the variability of a design may be too large to be considered for manual system integration.

Ten designers doing a conceptual design and each is passing on a minimum and a maximum of one component specific parameter to the integration team, the integration team would require to integrate 2^{10} different system designs for analysis and verification. This would be too time consuming when done manually. It is therefore not done. For system integration, each team will pass on their "best" design to the integration team.

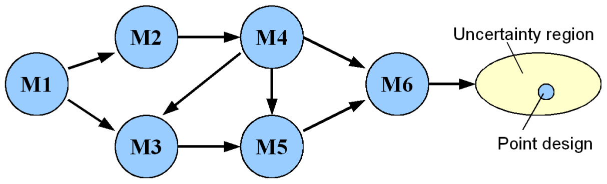

Figure 3: Design process graph

The integrated system is therefore not verified for variability due to uncertainties in subsystems and interactions between components located in different subsystems. As a result, the probability that this coupling is causing critical errors in the design is very high. Testing (expensive and time consuming) should eliminate most critical errors, but customers increasingly find unresolved critical errors after delivery of products. Additionally, different designs based on the same platform design will exhibit very different probability of failures.

What should be done?

To solve the complex system design problem, two technologies have to be developed. These are,

- Technologies have to be developed for automated design at the system level that considers uncertainty
- Technologies have to be developed for system level optimization of architecture and function.

For an automated design, a simulation capabilities must meets the following requirement,

1) modules, that are complete independent simulations of design methodologies
2) the simulations may be executed by the same or different simulation tools
3) a design process graph connects the design methodologies
4) Monte Carlo simulation capabilities, in order to be able to analyze sets or ranges of parameters
5) simulation model generation, in order to iterate over different architectures
6) requiring standardized modeling of architectural components
7) optimization methodologies that can optimize a design with respect to system level objectives
8) the simulation must be connected to a data base, in order to keep track of the large volume of information
9) models and data must be stored in a standardized way, in order to ease comparison and analysis

Figure 4 depicts such an automated design process simulation. Each designer has to develop an executable model of her/his design methodology. Process engineers design the process design graph that connect the executable design methodologies.

The simulation generates a set of feasible designs and maps component uncertainties into system uncertainties of coupled designs and can determine which designs meet system requirements for all uncertainties in parameters, architecture, missions/use cases, and environment. If the system performance falls within the permissible performances of the design specifications, the design is verified.

System uncertainty determined by the range of design variations also depicts the level of uncertainty at a given stage of development. As different members of the design team get up the learning curve, Figure 2, this uncertainty is going to decrease. Unacceptable levels of system uncertainty can be analyzed, causes determined and eliminated. The automated design process can track design changes and their impact on overall system design criteria, including performance, cost, quality of design. The benefits in cost and risk reduction of such an automation of design would be enormous. However, some critical technologies like modeling of design methodologies are still missing.

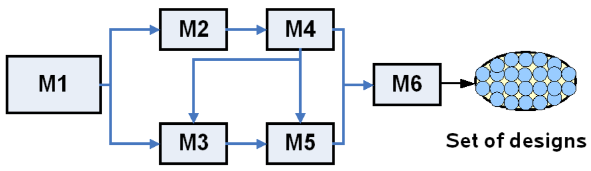

Figure 4: Executable design methodologies, connected by a design process graph

What can be done now?

Many technologies for automation and optimization of designs and design processes exist today. In [4] models the development process for a railway switching system and optimizes for team selection/load. Critical components of the development process were identified and eliminated that would have doubled development time. [6] optimizes a development process for automotive electronic control units using genetic algorithms, finds the optimal combination of design methodologies (team selection), and performance (timing, BER, cost and quality.) Areas for significant improvements were identified. Simulation set capabilities for integration of independent simulations are in [5].

Most critical today are early model based design methodologies for the overall system that reduce uncertainty before a design is partitioned to different design teams, Figure 5, plus technologies for optimization of architecture of overall design, e.g. optimization at vehicle level. This is made possible by the following technology developments,

- standardized modeling of architectural components of networked systems [7]
- Automated mapping of function into architecture and mapping of combined architectural and functional models into implementations [8]
- Development of standardized XML based data formats formats [9]

- Moving reliability analysis to early design stages [8]

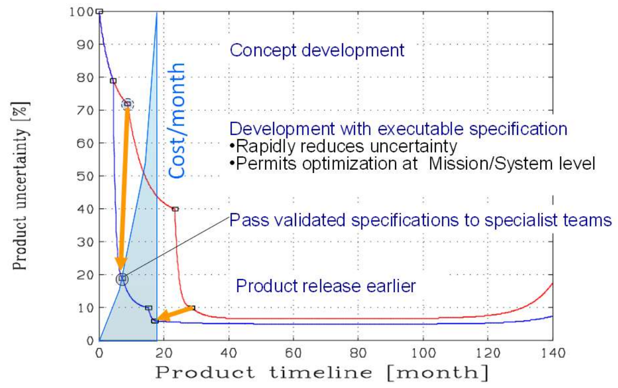

Figure 5: What can be achieved now

Computing the risk for this Mission Level Design process for the same automotive subsystem development we get,

$$\boldsymbol{R}_{\boldsymbol{MLD}}=\int U\,C_{Month}\,dt=61.0*maxC_{Month}+Overrun$$

The ratio of risk between the traditional design process and the mission level design process is,

$$\frac{R}{R_{MLD}}=4.9$$

We observe a significant risk reduction for the Mission Level Design process.

The benefits of executable specifications and vehicle level optimization are now demonstrated for the case of an executable specification of a power management system, a virtual prototype of a power train, and an vehicle level architectural optimization.

Executable specification of power management system

The problem: The power management system of a car had a critical error that lead to loss of battery power in worst case winter conditions. Low level analysis of the integrated system took more than two month and was ill suited to solve this problem.

The solution: An executable specification of the power power management system of the car, consisting models of worst case rular and urban driving scenarios, performance level models of engine and generator, battery model, FSM of power management logic and 27 models of systems using power in the vehicle. The development of the prototype model took seven days. It's simulation time was less than 1sec. Because of the short simulation time, the desired characteristics and parameter settings for the power management system could be determined within a day. In close to critical situations non-critical power users were turned off and the RPM of the engine was increased as necessary.

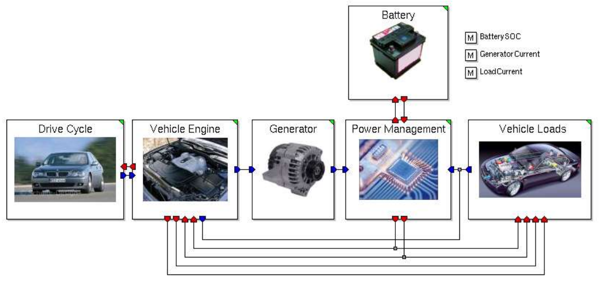

Figure6: Executable specification of power management system

Figure 7, 8, and 9 show engine RPM, and battery SOC for the worst case scenario. The minimum SOC could be adjusted to the desired value.

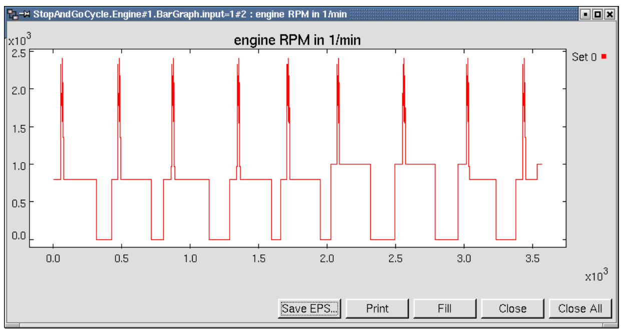

Figure 7: Load current for worst case scenario

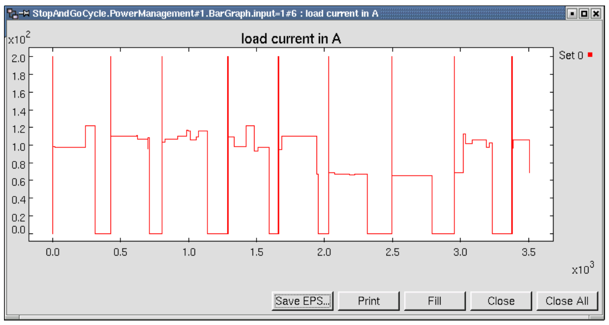
StopAndGoCycle.PowerManagement#1.BarGraph.input=1#6 : load current in A
load current in A
x10^2
Set 0
x10^3
Save EPS...
Print
Fill
Close
Close All

Figure 8: Load current for worst case city scenario

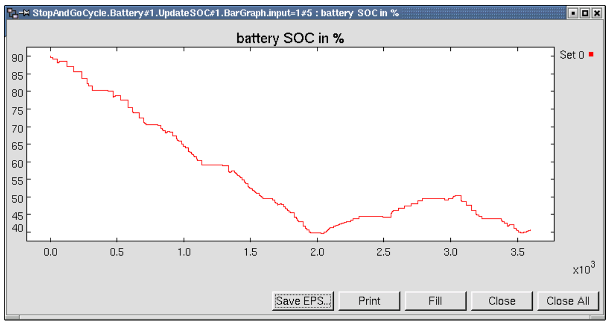

Figure 9: Battery SOC in %

Virtual prototype for engine management system

The problem: To cope with complexity in automotive electronics, interfaces like AUTOSAR are investigated. These define standardized functional interfaces. However, they impose a computational overhead to the system. Development of a prototype and and hardware-in-the-loop test is late in the development cycle and very costly for design iterations. Hence it reduces the competitiveness of an automotive supplier.

The solution: A virtual prototype for engine management systems, Figure 10, has been developed that includes models of software and hardware. With this model the functional behavior of protocols and software as well as the performance behavior of busses and processors can be investigated at mission level. Failures of the controller because of processing overhead, Figure 11, can be detected much before a prototype can be build.

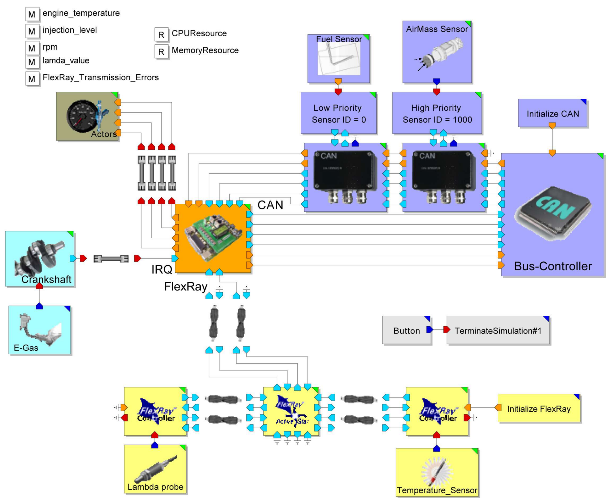

Figure 10: Virtual prototype of engine management system [10]

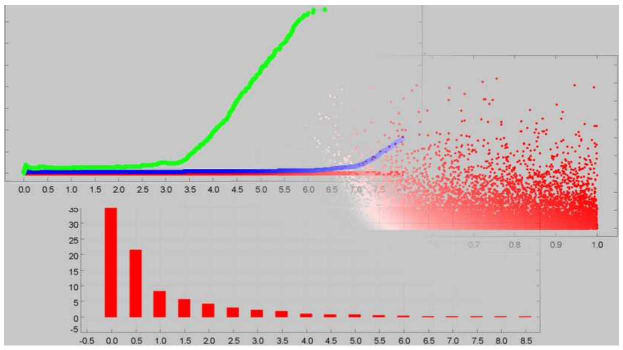

Figure 11: Virtual engine prototype analysis results [10]

Vehicle level architecture optimization

Avionics systems in aircraft are today highly complex with more than 1000 electronic control units, located centralized behind the cockpit. Long cables pass though the aircraft to actuators, sensors, communication and display units. This architecture is very sensitive to design changes in the aircraft.

Cabling cost optimization of an distributed architecture using simplified cable routing shows that the cost is minimized for a distributed architecture with 6 to 7 electronic racks, Figure 12. Six maintainable places were selected close to feasible locations that meet maintenance requirements [8].

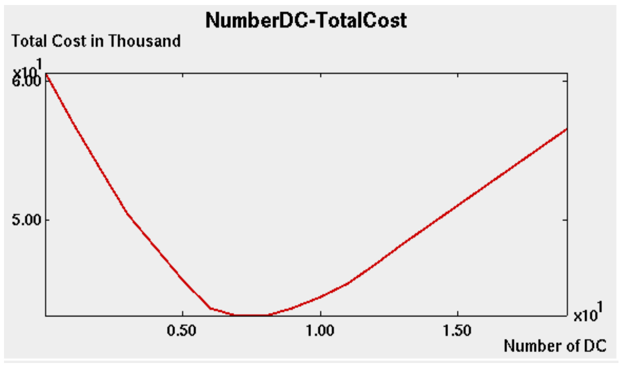

Figure 12: Cable cost optimization of distributed electronics

Different electronic IMA boxes are mapped into the six locations, Figure 13, optimizing different design criteria. The result is a XML description of the architecture.

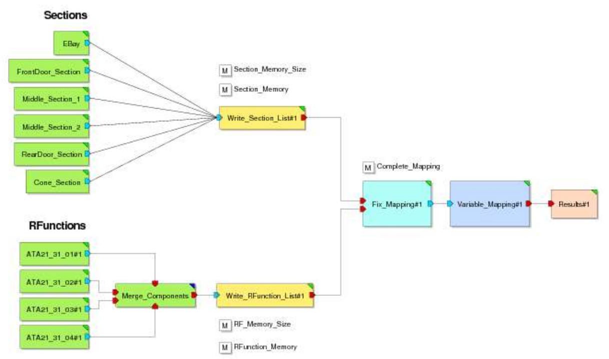

Figure 13: Architecture optimization

Behavioral models of ATA functions are mapped into IMA resources Figure 14.

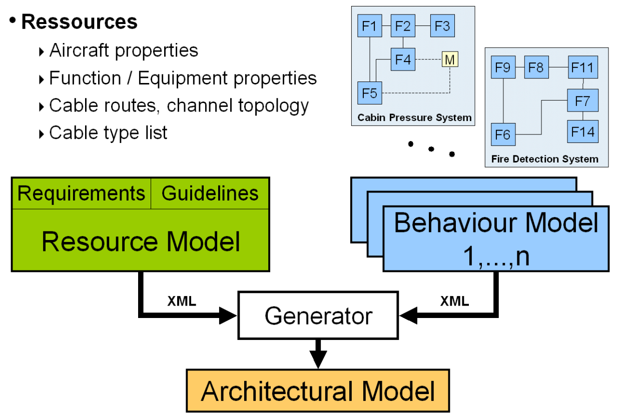

Figure 14: Models and design graph for architecture optimization, functional and performance verification

The diverse tasks of the design flow like topology optimization, model generation and functional and performance verification in the uncertainty region of the Behavior Models (BM) can only be automated with simulation set capabilities, described before.
BMs determine the functional performance of each aircraft subsystem included. Models at different levels of abstraction may be used for the subsystems.

The Resource Model (RM) implements aircraft design guidelines concerning power, space and other installation aspects as well as requirements of included aircraft subsystems and implementation of their behavioral models. Several optimization algorithms are implemented in the RM to find a Pareto-optimal solution for the networked systems compound on aircraft-level. The output of the RM is a XML-file describing the optimized architecture topology. A behavior to architecture mapper automatically generates the Architectural Model (AM). The AM contains the functional behavior of aircraft subsystems and performance components like interfaces and channels, Figure 15. The AM is used to evaluate different design criteria for the overall system behavior and interactions of its subsystems.

Figure 16 compares a reference architecture with the optimized architecture. Optimization reduced wiring by 68%, reduced cost by over 70% and increases availability by orders of magnitude.

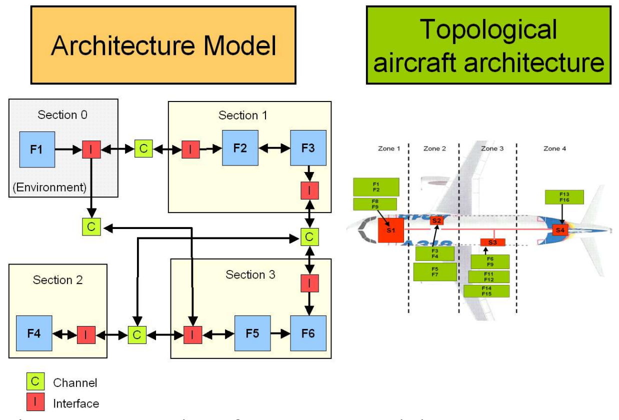

Figure 15 Results of resource model

Performance item	Reference System	Optimized System
Cost of architecture [$]	100%	28.4%
Reduction of cables [%]		68
Weight of architecture [kg]	280.06	201.00
Weight of cables [kg]	108.21	29.15
Length of cables [feet]	1531.46	453.84
MTBF [h]	55632.88	57534.15
MTTR [h]	31.63	29.89
MTBUR [h]	49164.46	50874.30
Availability	0.99	0.999999999998

Figure 16: Comparison of old and new design

Conclusions

Current design methodologies in complex system design can neither optimize systems at vehicle level nor can it treat uncertainty in design and integration. This causes designs much under todays technical capabilities, a high probability of critical problems after design and expensive corrective actions.

A new complex system design methodology and enhancement to the development tool MLDesigner have been developed hat overcomes these problems. The new technology is demonstrated for the examples of a power management system, an engine management system and a vehicle level network optimization. It is shown that this new technology can find critical problems early in the development cycle, significantly improve networked architectures and can significantly reduce development time, cost and weight.

REFERENCES

[1] John Hines, We Don't Do Design Correctly!, Keynote speech at IEEE *9th International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems*, MASCOTS 2001, Cincinati, Ohio, Aug. 15-18, 2001.

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

[3] Horst Salzwedel, Mission Level Design of Avionics, *AIAA-IEEE DASC 04 - The 23rd Digital Avionics Systems Conference 2004*, Salt Lake City, Utah, USA, October 24-28, 2004.

[4] Manuela Tröbs, *Analyse, Optimierung und Simulation des Simis IS Entwicklungsprozesses*, Thesis, Siemens/Technical University Ilmenau, June 2006.

[5] MLDesigner® Manual v3.0, http://www.mldesigner.com

[6] Tom Dengler, Host Salzwedel, Optimierung von komplexen Enwicklungs-prozessen mittels simulationsgestützter Prozess-analyse, *52th International Scientific Colloquium*, Ilmenau, September 10-12, 2007.

[7] Tommy Baumann, Horst Salzwedel, Mapping of Electronic System Level (ESL) Models Into Implementation, *DATE'07*, Acropolis, Nice, France, April 16-20, 2007.

[8] Nils Fischer, *Design of a Plug-and-Play Development Environment for Optimizing Avionics Systems Architectures*, Thesis, Technical University Ilmenau, July 2007.

[9] Stefan Riehmer, *Standardizing Data Storage for MLDesigner and Octave using the Open Office Document Standard*, Thesis, Technical University Ilmenau, August 2007.

[10] Till Resmer, *Modellierung und Simulation von Echtzeitsystemverhalten nach dem Mission Level Design Ansatz*, Thesis, Erfurt University of Applied Sciences, January 2008.