MISSION LEVEL DESIGN OF COMPLEX AUTONOMOUS SYSTEMS

Volker Zerbe
Department of Automatic Control and System Engineering,
Technische Universität Ilmenau, Germany
e-mail: volker.zerbe@tu-ilmenau.de

Invited Paper

Abstract – *In this paper the idea of Mission Level Design (MLD) for complex autonomous systems, especially for autonomous underwater vehicles is presented. Mission Level Design can be thought of as system design on an abstract level as well as an overall simulation in virtual worlds. It can be used to test the design of complex systems. Missions (generalized use cases) embed the system and describe user requirements for the system design. This concept helps to find problems in early design phases and offers developers a first idea of system performance and characteristics.*

MLDesigner (a design tool of the latest generation) is used for the design of an autonomous underwater vehicle (AUV). In this paper an overall system model with functional, architectural and environmental aspects is presented. This model is used to solve design questions through the simulation of user defined missions in a virtual world. The results of the simulation are shortly considered.

1 INTRODUCTION

Today more and more complex systems have to be eveloped.The complexity of electronic systems is doubling every 1.5years. We have now a complexity 100 times higher then in 1990. At the same time the time-to-market requirements are growing shorter, therefore the product life is shrinking. The complexity of a mobile phone is comparable to the complexity of a workstation build 15 years ago but will have an essentially shorter product life. The increasing complexity raises the probability of errors. 60 percent of all ASIC designs fail. Independent specialists design modules described by the system specification. If there is an error within the specification it will be found during the testing period of the system. But this probably causes iterations back to the design phase, which are expensive. Therefore it would be better to find this kind of errors in the early design phase.

Future systems will be so complex that an overall simulation in its totallity is impossible. The solution is an breakthrough in the design of complex embedded systems. This shifts the center of development to early phases of the design process. We call it the 4th generation of electronic-design-automation tools: the system design on mission level. Previous design concepts started to deal with more complex systems by using levels of abstraction. The first steps were hardware description languages and followed with those for algorithms and systems. The system level was reached by coupling different design tools. Examples are SES-Workbench with Object-Geode [5] or Matrix$_x$ with Statemate [6]. In spite of this, there is no general possibility to simulate both hybrid reactive/transformatic behavioral and architectural models in a common performance simulation like an overall system simulation. What is the actual situation? BoNES [4] and Cossap [1] are end-of-life, Cadence still sells VCC. Since 2000, MLDesigner, a tool of the 4th generation exists, figure 1. In this way, Mission Level Design (MLD) is just another step to the consequent formalization of the design process.

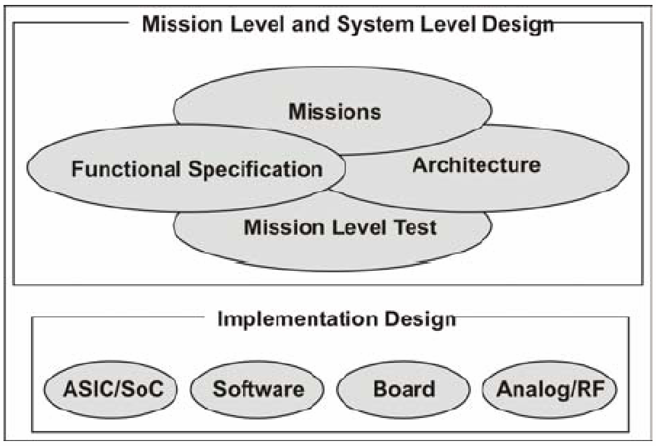

Figure 1: MLDesigner [9]: Mission level design approach

2 MISSION LEVEL DESIGN

What does system design on mission level mean? It stands for design at the mission or operational level in order to consider all dependencies early in the design. It is the description of a mission in a virtual reality like an overall system simulation for defined operational conditions.

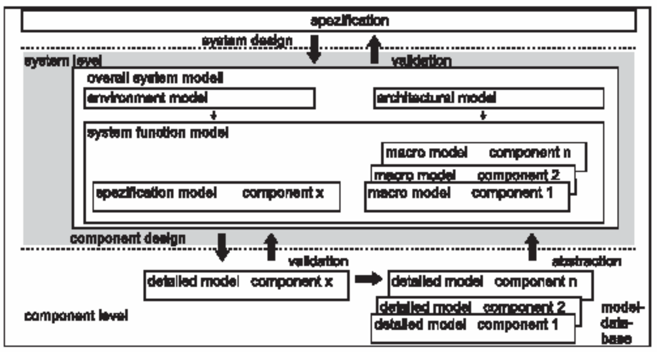

Figure 2: *System design on mission level*

It offers a closed design- flow from mission level to silicon and it allows the test of the design on mission level in a virtual world as well. The shift of the design automation to more abstract levels is based on a consequent formalization on all levels. Therefore the implementation can be done automatically, if the formalization starts early in the design phase. An advantage is that the time-to-market decreases. This results from the fact that in an early phase the design is proven as free of faults and the requirements are fullfilled. The breakthrough in the design of complex embedded systems will be achieved because of the overall approach. The overall system model consists of functional, architectural

and environmental components, which define different aspects of the system in a virtual world. The properties of those components are listed below:

- **Functional models** are the feasible parts of the specification of the system or its components. They describe the functional characteristics (continuous or discrete systems, software), therefore the behaviour of a system or component.
- **Architectural models** provide a description of the available resources. They are used to examine the performance characteristics of a system or component. Normally, one functional model is mapped on different architectural models to find the best configuration.
- The **environmental model**, a set of mathematical models, describes relevant parts of the surroundings. Not yet implemented components may be integrated into the environment, too.

Up to now, the system design process is top-down oriented. Starting from the system level the draft is subdivided and becomes refined multiple times until the implementation. Now it is possible to design the system on the system level only, as shown in Figure 2. The implementation is affected by automation of the specified components.

In chip production complex systems are designed based on predefined and re-usable components, called Intellectual Properties (IP). IP's are part of huge libraries of different application fields. They consist of functional and detailed descriptions for the implementation. The system build with IP's is put on the silicon. Specification models of the predefined components are integrated within the design phase. This process is a bottom-up design and is realized by abstraction, because the architectural models are derived from low-level IP's. A main advantage of this design concept is that predefined components are validated and verified and are proven fault-less because of that. The behavior of components in the system context is tested by embedding them in the overall system model. This validation of the specifications causes a reduction of design errors and time.

Mission Level Design deals with the fact that complex systems have too many parameters to check them all in every possible situation. Therefore typical operational requirements have to be defined and included as use cases in the formalization of the specification. The validation takes place by the simulation of the overall system model, where missions directly define driver and evaluation.

Formal methods alone do not guarantee the absence of errors in the resulting system. Because of the nearly infinite set of system states, a validation of the system behaivior can not be realized. The goal is to guarantee fault-free behaviour under all possible operational conditions. For mission critical applications and those with high financial risks a fault-free behaviour of the system under operational conditions is essential. That means possible operational conditions have to be defined as operational requirements and included in the formalization of the specification. These definitions base on missions. The sequence diagrams, which serve as description of scenarios within virtual worlds, are derived from those missions. So, Mission Level Design is the description of a mission in a virtual world, an overall system simulation for defined operational requirements and the design test on mission level.

3. AUTONOMOUS UNDERWATER VEHICLE

We use the Mission Level Design concept for the design of an autonomous underwater vehicle (AUV). An AUV is a highly complex system and its loss would be very expensive. The design risk is immense. The aim of our work is to find answers to the following questions:

- Are the manoeuvres successful under limited resource and given environmental conditions?
- What happens with the battery power during the mission?
- Is the data bus design correct?
- Do we get enough sensor data?

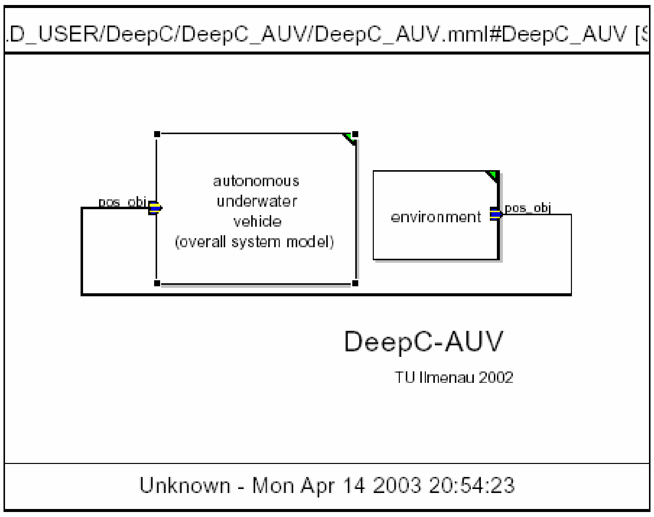
.D_USER/DeepC/DeepC_AUV/DeepC_AUV.mml#DeepC_AUV [S
autonomous underwater vehicle (overall system model)
pos_obj
environment
pos_obj
DeepC-AUV
TU Ilmenau 2002
Unknown - Mon Apr 14 2003 20:54:23

Figure 3: *Overall system model*

For this reason we need a simulation tool, which supports Mission Level Design, MLDesigner [9]. It allows the simulation of different model types in one model. There exist basic models for time discrete, synchronous data flow, finite state machines and continuous time models. It is also possible to import BONeS and Cossap models or to develop own models in C++.

We have started to build the overall system model for the autonomous underwater vehicle, which is hierarchical and parametric as depicted in figure 3. The environment and the autonomous underwater vehicle are placed at the highest level. Simultaneously, this mirrors the idea of Mission Level Design, which is to examine a system inside its environment.

The environment involves all relevant information about the surrounding. Therefore it consists of three relevant parts: First, there is the current of the sea, which is position dependent. Second, there may be other targets or obstacles. Finally, a collision detection for all the objects has been realized.

The autonomous underwater vehicle is modeled by a set of sub-models representing important components. For an overview look at figure 4.

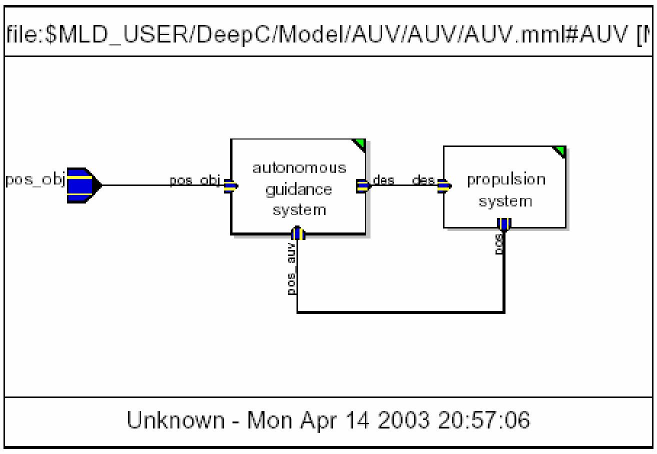
file:$MLD_USER/DeepC/Model/AUV/AUV/AUV.mml#AUV [M
pos_obj
pos_obj
autonomous guidance system
des
des
propulsion system
pos_auv
pos
Unknown - Mon Apr 14 2003 20:57:06

Figure 4: *AUV model*

It is taken from MLDesigner and shows the systems and modules of the AUV. These are:

- o *cooling system*
- o *payload*
- o *trimming system*
- o autonomous guidance system
- o propulsion system
- o *power generation*
- o *energy management*
- o *emergency system*

At the moment, most of the systems (the ones marked italic) are predefined only and will be implemented later. Each subsystem stands for fundamental functions. They will be designed for a real autonomous underwater vehicle by specialists on this topic. These experts use special models, which are too detailed for an overall simulation. Because of that, we use generalized versions of those models for the simulation.

The propulsion system includes engine and AUV dynamics. The equations of motion are adopted from Fossen[2] and Brutzman[3]. This model supplies the global position and the local velocity of the AUV, see 5.

The autonomous guidance system consists of the control computer, the sonar and the inertial navigation system (INS). This model is shown in Figure 7.

The computer system controls the velocity and the position of the AUV. A path planning module creates the desired values.

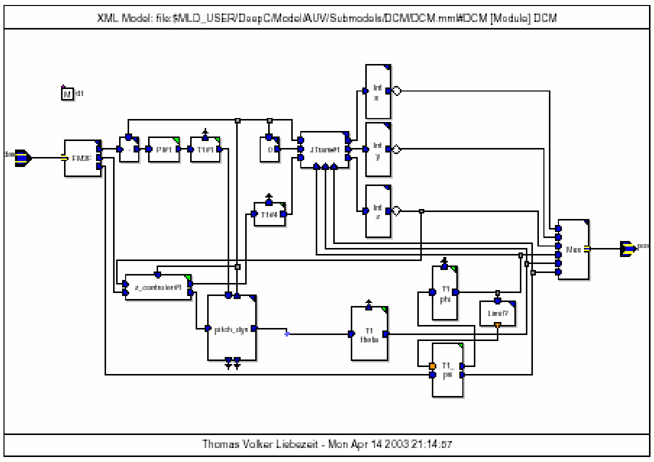
XML Model: file:$MLD_USER/DeepC/Model/AUV/Submodels/DCM/DCM.mml#DCM [Module] DCM
Thomas Volker Liebezeit - Mon Apr 14 2003 21:14:57

Figure 5*: Propulsion system model*

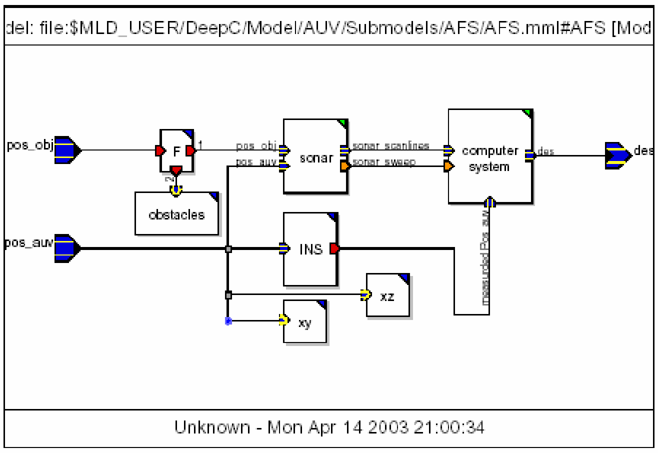

Figure 6: *Autonomous guidance system model*

Additionally, the computer system includes an image processing module to analyze the sonar image. Its results are used by a collision avoidance module, which is interacting with the path planning module.

The sonar model, depicted in figure 8, is used to detect objects in the surrounding of the vehicle. As a functional model it does not base on exact physical behaviour. Instead a geometrical approach is used mapping physical parameters to geometrical ones. The sonar model receives the position of the AUV (`pos auv`) and those of the other objects (`pos obj`), which are expressed in global coordinates. That's the reason why the object positions are first transformed into AUV coordinates. The SonarHardware-block controls the sonar's field of vision and behaves as follows: It pans from *minPing* to *maxPing* and back in the sonar plane with a steering increment of *delta*. The sonar pauses *wait* iterations after each ping. A complete sonar image will be signaled on the *sweep* output. The sonar position is used to transform the object positions in the sonar coordinate system. Here the SonarView-block checks, if the objects are in the field of vision. This happens in dependency of the *run* signal from the SonarMotion block. The accepted space for the field of vision is defined by ±*maxAngleH* and ±*maxAngleV* (in the horizontal and vertical plane, respectively) and the maximum distance *maxDist*. The SonarView output is a stream of objects in the field of vision. Their local sonar positions are transformed back to local AUV positions. These coordinates together with the *sweep* signal build the sonar output. For the visualization of the simulation result some values are logged by the print blocks.

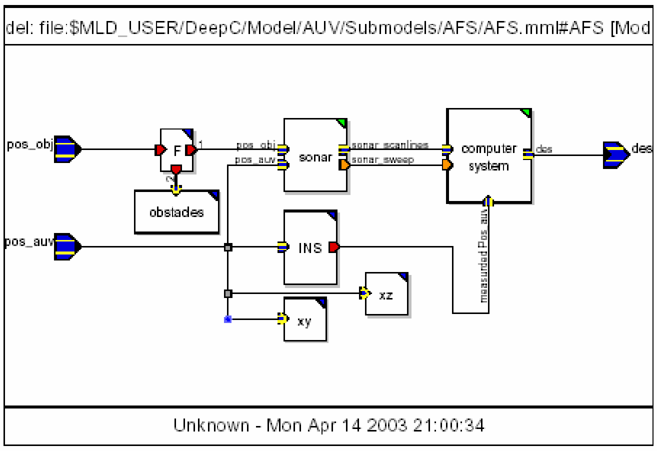

Figure 7*: Autonomous guidance system model*

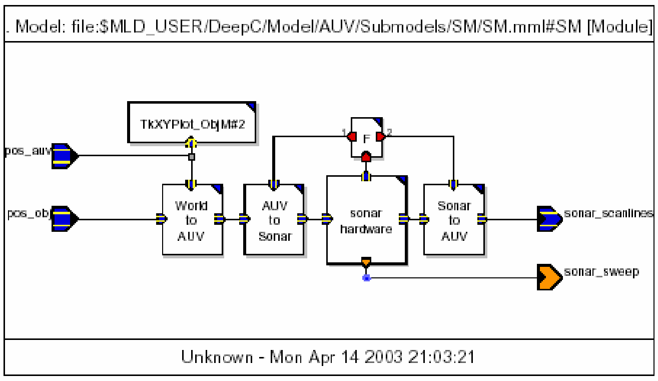

Figure 8*: Sonar model*

4. MISSION AND RESULTS

As a first test **mission** we have chosen the following situation: The vehicle has to keep its velocity and has to avoid collisions with appearing objects. To test it five fix objects are lying in the path of the AUV. This mission is to meander in 100 m Depth and to find all obstacles. The simulation results can be used to test the dynamic behavior of the vehicle and to test the parameter settings of the sonar.

The AUV starts at a position (0, 0, 0) with a velocity (2, 0, 0), whereas the (3×1)-vector describes the three translational positions (*m*, *deg*) or velocities (*m/s*, *deg/s*). The fix objects are at the coordinates (400, 500,-100), (650, 175,-100), (870, 300,-100), (900, 300,-100) and (900, 320,-100). The sea has a current of (1, 1, 0). As mentioned above, the goal of this mission is to hold the starting velocity and to avoid collisions with objects, that appear in the sector ±10. in front of the autonomous underwater vehicle. The sonar scans a sector of ±22.5. and pauses one simulation step after each ping, which has a beam shape of 1.8.. The maximum range of the sonar is 80*m* and the steering increment amounts to 1.8..

The simulation **results** are shown below.

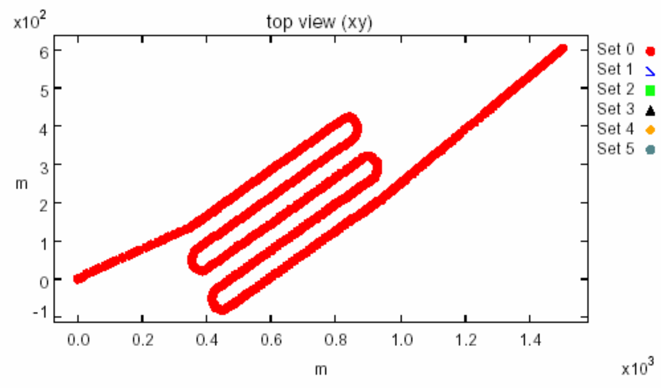

Figure 9: *Global trajectory - top view*

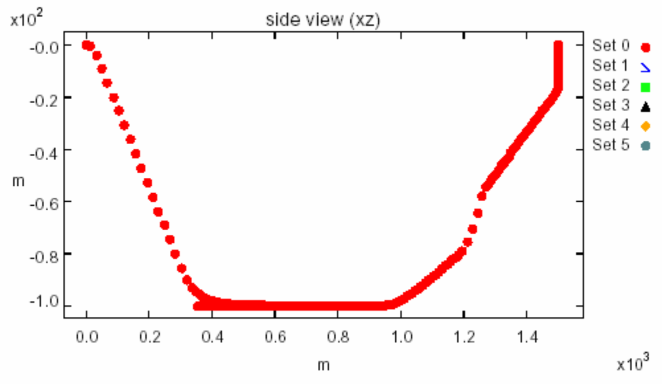

Figure 10: *Global trajectory - side view*

The global trajectory of the specified mission is shown in figures 9 and 10. Figure 10 explains that the vehicle does not reach the water surface at first. The mission stops at about 20 meters below. By the work of the thrusters a correction of the position takes place, the vehicle moves to the surface. The next diagram (Figure: 11) shows three curves. The upper one defines the course of the AUV. The yaw angles for meandering are clearly detectable. The straight line near zero shows the constant velocity of about 2 *m*

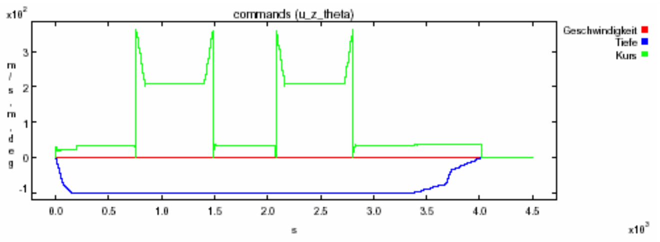

Figure 11: *uz-theta*

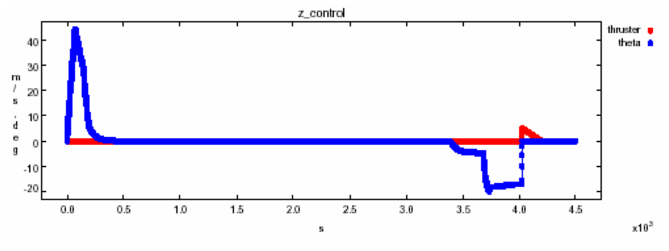

Figure 12*: z-control*

The last curve provides the control of the depth. The diving process is divided in two and the surfacing process in three parts.

Figure 12 gives the angle swings for the depth control and the activation phase of the thruster.

Figure 13 plots the obstacles and the figure 14 the scanned and detected obstacles. In the sonar field of vision three obstacles are detected. The problem is the reduced angle of vision while changing the course in curves like meandering. This requires an in time conformist control of the sonar depending on the predefined route in the mission plan.

For another detailed example, an earth observation satellite, see [7].

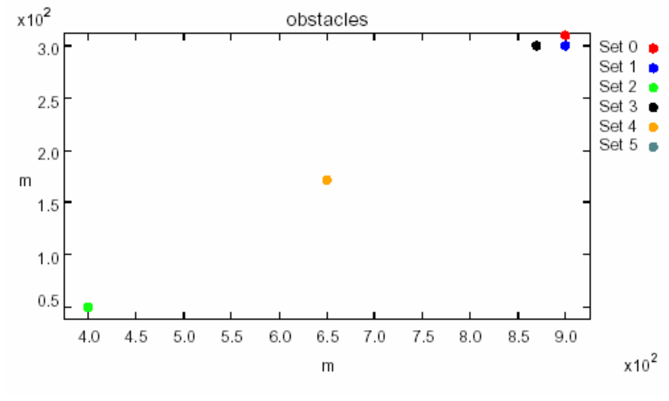

Figure 13*: Obstacles*

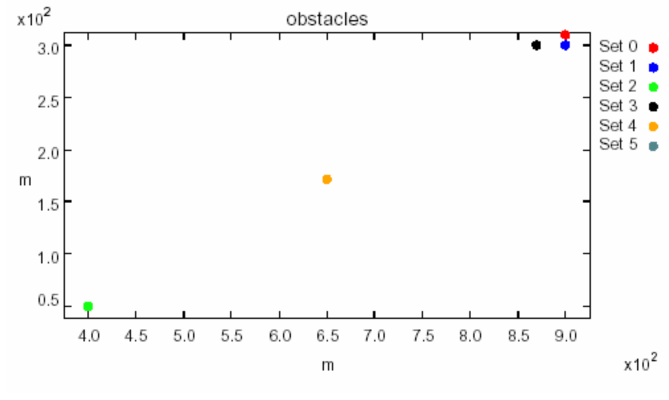

Figure 14: *Founded obstacles*

5. CONCLUSION

Several tools have been introduced during the last years, addressing system design problems separately, but leaving the problem of overall system verification unsolved. However, these tools rarely interoperate with each other, making the verification problem extremely difficult. To work around these limitations, designers are attempting to link different tools through co-simulation techniques. This tedious process distracts the designers from the main design task, is error prone and inefficient for large system verification. In Ilmenau we are in the process of developing the mission level design tool MLDesigner, visit the page http://www.mldesigner.de. It's multidomain simulator permits the seemless integration of the design flow from mission/operational level to implementation handoff and test of complex systems for the first time. We have presented the idea of Mission Level Design on the example of an autonomous underwater vehicle. The concept of MLD was explained, an overall system model was presented and used for testing the parameter settings of an AUV sonar. For a better evaluation of trajectories, a 3D visualization tool should be used. Another important fact of our future work is the management of different missions in one database.

REFERENCES

[1] Synopsis: Cossap Reference Manual. 1993

[2] T.I. Fossen, *Guidance and control of ocean vehicles*, John Wiley & Sons Ltd. 1994

[3] D. P. Brutzman, *A virtual world for an autonomous underwater vehicl*e, dissertation Naval Postgraduate School Monterey California 1994

[4] Cadence Alta Group: BONeS-Designer Modeling Handbook. 1996

[5] O. Rumpf, Eine rechnergestützte Entwicklungsumgebung für integrierte Verhaltens- und Leistungsuntersuchungen am Beispiel von KFZ-Datenbussystemen. TU München, Dissertation, 1996

[6] FZI: Abschlussdokumentation des FZR Projektes. FZI, 1996

[7] V. Zerbe, et al.: Mission Level Design of Control Systems. In: Proceedings of SCI/ISAS 5th International Conference on Information Systems, page 237-243, 1999

[8] V. Zerbe, Th. Liebezeit, T. Radtke, Mission Level Design in Robotics. In: Proc.-CD 10th International workshop on robotics in Alpe-Adria-Danube region (RAAD'01) Vienna, May 2001

[9] [9] MLDesigner: http://www.mldesigner.com