PROCEEDINGS

55. IWK

Internationales Wissenschaftliches Kolloquium
International Scientific Colloquium

13 - 17 September 2010

Crossing Borders within the ABC

Automation,

Biomedical Engineering and

Computer Science

Faculty of
Computer Science and Automation

www.tu-ilmenau.de

TECHNISCHE UNIVERSITÄT
ILMENAU

Impressum
Published by

Publisher:	Rector of the Ilmenau University of Technology Univ.-Prof. Dr. rer. nat. habil. Dr. h. c. Prof. h. c. Peter Scharff
Editor:	Marketing Department (Phone: +49 3677 69-2520) Andrea Schneider (conferences@tu-ilmenau.de) Faculty of Computer Science and Automation (Phone: +49 3677 69-2860) Univ.-Prof. Dr.-Ing. habil. Jens Haueisen
Editorial Deadline:	20. August 2010
Implementation:	Ilmenau University of Technology Felix Böckelmann Philipp Schmidt

USB-Flash-Version.

Publishing House:	Verlag ISLE, Betriebsstätte des ISLE e.V. Werner-von-Siemens-Str. 16 98693 Ilmenau
Production:	CDA Datenträger Albrechts GmbH, 98529 Suhl/Albrechts
Order trough:	Marketing Department (+49 3677 69-2520) Andrea Schneider (conferences@tu-ilmenau.de)

ISBN: 978-3-938843-53-6 (USB-Flash Version)

Online-Version:

Publisher:	Universitätsbibliothek Ilmenau ilmedia Postfach 10 05 65 98684 Ilmenau

Home / Index:
http://www.db-thueringen.de/servlets/DocumentServlet?id=16739

A PRACTICAL WORKFLOW FOR AUV CONTROL SYSTEM DESIGN

Marco Jacobi, Sebastian Matz, Frank Schrödel

Fraunhofer Application Center System Technology
Am Vogelherd 50
98693 Ilmenau, Germany

ABSTRACT

This paper will introduce a workflow for the design of control systems for autonomous underwater vehicles (AUVs) consisting of a testing and modeling environment and specialized tool chain. The workflow combines different steps for implementing algorithms from the idea to the final implementation. These steps include rapid prototyping, testing with simulations, testing in controlled environments and productive testing. We will outline the development process model and show the steps exemplary on a wall following algorithm.

Index Terms— Autonomous Underwater Vehicles; AUV; UUV; Inspection; Test; Software

1. INTRODUCTION

The importance of AUVs was rising in the past and will do also this in the future [1]. Main tasks of AUVs are exploration and inspection.

AUVs play a big role in the exploration of the ocean, which is an important part in our climate system and a source for a lot of resources such as food, minerals or energy and many parts are not discovered and many topics about are still unknown. Currently, the UUVs are often remote controlled, but for long term and deep sea missions they gain more autonomy to fulfill their missions.

The second task for AUVs is the inspection of a variety of underwater buildings which are difficult to reach and expensive to monitor frequently. These underwater buildings can be offshore wind park foundations, reservoir dams or sluices. The AUVs can be also used for harbor safety; they inspect for instance sheet pile walls to search for damages or corrosion. Also ship hulls need to be inspected to discover leakages or smuggling goods.

The project "CView" funded by the German government aims to cover these inspection tasks. The goal is to build an AUV which can handle these inspection tasks.

Like mentioned before, harbors, sluices and other underwater buildings need to be safe and unharmed. Therefore, the AUVs need to be function in a safe manner. They do not have to harm people or to damage these buildings due a malfunction.

With difficult under water communication, it is hard to observe the AUVs during an mission in difficult environments. Therefore, the vehicles have to function reliably. All functions of an AUV have to be developed and tested carefully. We introduce a workflow to improve the development process and the reliability of AUVs.

2. DEVELOPMENT PROCESS MODELS

All algorithms developed for UUVs are software; often distributed in different units like microcomputers or embedded (micro-) controllers.

Therefore, different software development process models exists [2]. We show some basic models and position our workflow within these. They can be classified as:

- the waterfall model [3],
- extreme programming (XP) [4] and
- the V-model [5].

The **Waterfall Model** is a sequential development process. The progress can be seen as flowing downwards like a waterfall (fig. 2, descending branch): if one step is finished the next can be started.

This model is suited for stable software projects, has an structured approach which can be easily communicated and understood, especial in fluctuating project groups, and it has clearly definable milestones, which provide a progress monitoring.

In [4], Kent Beck describes the philosophy of **Extreme Programming (XP)** that intends to consider requirement changes and improve the software quality. Extreme Programming is a type of agile software development [6]. This development philosophy is considered for short development cycles with early, concrete and continuing feedback; it relies on communication and tests. The principle of XP is illustrated in figure 1.

One of the XP goals is to get an initial runnable implementation as quick as possible. Within these the

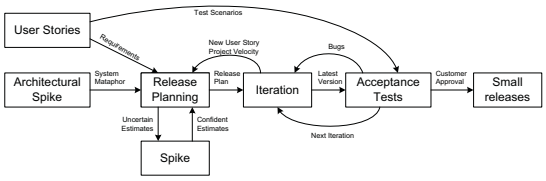

Fig. 1. Extreme Programming Workflow [7]

foundation for continuing improvement is made. Generally, all features are implemented when required. The biggest drawback of this model is the lack of an overall design specification and also documentation since most communication are oral. An improvement of the classic XP is industrial extreme programming [8].

The **V-Model** can be considered as an extension of the waterfall model to take the issue of quality into account (fig. 2). It introduces to each phase of the development process an associated phase of testing [2,5]. The V-Model XT (eXtreme Tailoring) is the improvement of the classic V-Model [9].

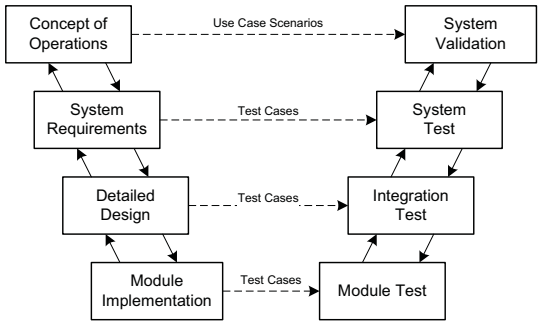

Fig. 2. V-Model

The V-Model XT can be considered as a guideline for systematic planning and execution of complex projects. It provides strategies for reducing risks, improving quality, cost control and improvement of communication between all project partners. The model itself can be adapted to the current project situation [2].

All philosophies of these development process models have advantages and disadvantages. The idea is to combine these development models in a **Combined Approach**. In every one of these models testing takes place. The difference in the time when the tests are defined and when they are performed; also the issue what is done with the results of these tests.

As mentioned before, the algorithms development for UUVs are usually an iterative process. When specifying the tasks for a vehicle the sensor configuration has to be specified within. The constrains and implementation details of the algorithms for the used sensors are not known in detail. The sensor and actor selection and configuration can be changed due the requirements of the used algorithms. This process can be good handled with the XP philosophy of highly adaption to new requirements.

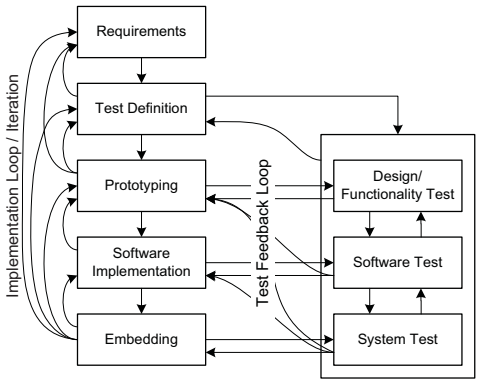

Fig. 3. Combined Model

As said, the UUV consists not only of software; it is a system of hard- and software, where frequently changes according to XP are expensive. Therefore, a planning stability is required. The V-Model provides this stability. This stability and the agile development have to be in a balance. A combined approach is recommended: using the planning and documentation for the system design and the agility for adapting this design evolutionary within development and testing using the gained knowledge (fig. 3).

3. WORKFLOW TOOLS

In order to obtain a high quality and safe algorithms for underwater vehicles especially for autonomous ones, all algorithms have to be tested extensively as mentioned in the combined development model. Therefore, different tools were introduced and a tool chain was build matching the workflow introduced in this paper.

Simulation and prototyping

The simulation environment is subdivided in two major phases: firstly there is a MATLAB/Simulink environment with a simple vehicle model for algorithm rapid prototyping and for the basic controller design. Different software tools can be used to provide this environment. One of the most used prototyping software system is MATLAB/Simulink. Therefore, various models for kinematic and dynamic vehicle simulation are specified [10] and implemented. For starting implementation and testing the basic algorithm design these tools produce fast results.

The algorithm and controllers can be tested on stability and basic functionality in noncomplex environments with this simulation models.

Secondly, with a complex simulation model and implemented algorithm plus complete vehicle software the

system is tested in a virtual reality environment which provides sensor simulation within complex 3d-scenes (fig. 4).

Fig. 4. 3d-Environment – ship inspection

After successful evaluation of new algorithms in the simulation environment the next step will follow: testing with real sensors in the test basin. This test needs not only the implemented algorithm; instead, it requires the whole vehicle software, which has also to be implemented for the tests in the virtual reality environment tests.

When these tests are successfully performed in the controlled environments then the vehicle is ready for sea trials.

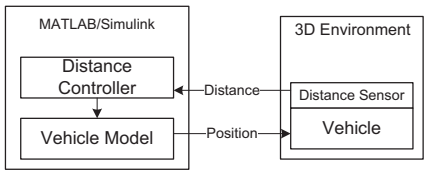

Fig. 5. Interaction VR and MATLAB-model

Visualization and complex environments

For vehicle simulation in complex environments with different sensor combinations like sonar or distance sensors these dynamic models and also the implemented algorithms can be coupled with a 3d virtual reality environment (CViewVR) (fig. 5) [11, 12].

This 3d environment is based on open source tools which are well documented, widely used, in active development and adaptable to own demands. The used engine here is ogre 3d [13]. With this tool a complex scene description was implemented for visualizing underwater scenarios and defining sensor configurations for a variety of underwater vehicles. A special toolset was compiled for creating these scenes consisting of CAD-Modeling software, 3d-modeling and animation software, landscape generators using height information and also an own scene and sensor description language was developed.

A big effort was done in simulation of sonar and distance sensors using physics engine for sonar beam modeling described in [11] and [12] for use in inspection tasks. Inspection objects can be ship hulls, harbor facilities, sluices etc; special scenarios were defined and scene files modeled.

Additionally, this 3d environment is also used for planning new missions, for mission post processing and can also used for visualizing and observing the vehicles during an actual mission when vehicle navigation data is available. The environment model can easily be extended with different other models using the 3d engine scene graph, such a model can be a pollution model for pipeline leak identification and tracking or something similar.

Vehicle dynamics simulation and control are separated from the whole environment model and can be considered as a distributed simulation. The reason therefore is mainly computation load balance and implementation issues. The virtual reality provides interfaces over Unix sockets for moving the vehicle and for sensor feedback accessible for the vehicle models (fig. 6).

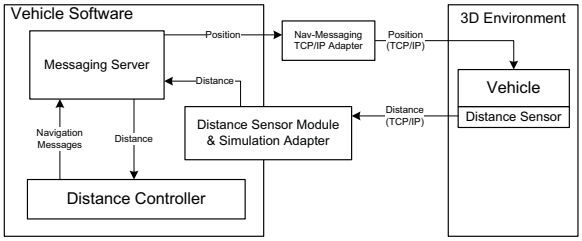

Fig. 6. Interaction VR, simulation and vehicle software

The vehicle control and navigation modules should be the same used in real mission in the vehicle itself. The algorithm developed with modeling tool is adapted to the vehicle control software and can be tested with the complete software environment used on the vehicle.

With the 3d environment interfaces and the vehicle dynamic models the navigation software can also be run on the vehicle computers itself. The vehicle manufacturers do not need to open their software for connecting with the test environment only implementing the interfaces. Interfaces for ConSys [14] and for the vehicle control software from our industrial partner are implemented.

In summary the complete simulation can be distributed on different computers and all modules are exchangeable and the step from virtual reality to realty is short.

System test

The 3d simulation environment is used for rapid testing during the algorithm development process. The simulated sensors are based on sensor models. To test the algorithms and the control software with real sensors in near real missions within a controlled environment,

the Fraunhofer Application Center System Technology built a test basin (fig. 7).

Fig. 7. Test basin with ExAUV

The vehicles can be tested in this basin as complete mechatronic systems: all sensors, actors, vehicle dynamics and the control algorithms can be validated in test missions. In this way sensor and navigation noise and errors has to be handled. Those test missions can be for instance the inspection of a sluices, ship hull sample section or sheet piles used in harbor constructions.

When the complete system had been successfully validated final tests should be in an environment where the missions of the vehicles take place. Sea trials are usually expensive and need a lot of time for preparation, execution and result analysis. Therefore, the system should be validated in virtual reality and in the test basin to save money and time.

All these steps take place one after another as in the V-Model. But every single test and validation run increases the amount of knowledge which can be used in every single step as the XP philosophy points out. For instance the tests in the basin can be used to optimize and adapt the simulation models and the algorithms. A test in the basin with a specific hardware configuration can be needed to specify the simulation model or to analyze the sensor and actor constraints for defining and designing the requirements and algorithms.

4. EXAMPLE

The presented workflow will exemplary applied in the development of a distance keeping control algorithm. The main goal for this algorithm is to follow a wall with a specific distance for inspection tasks. The implementation of the controller will use all steps of the presented workflow:

The initial requirement for the algorithm is to keep the specific distance which depends on the distance requirements of the used sensors for inspection. Additionally, a collision with obstacles or the wall should be avoided.

With these requirements the testing scenario was defined and a 3d scene for the virtual reality environment was set up.

The controller requires an accurate vehicle model. Therefore, a theoretical analysis was made with mathematical modeling of the actors, the vehicle inertia and hydrodynamic coefficients and finally the vehicle motion equations [10]. A signal noise analysis was done for all this models to obtain the model parameters. Finally, validation experiments were made within the test basin.

The development of the controller was started with controller prototyping in MATLAB/Simulink. With this prototyping process, different controllers (PID, sliding mode etc.) were evaluated. The virtual reality was used for sensor data input and visualization. Therefore, the Simulink model was coupled with the 3d environment (fig. 5 and 8).

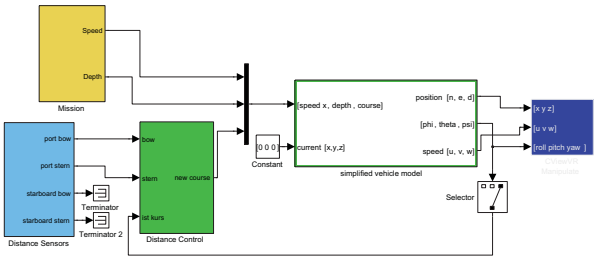

Fig. 8. Distance controller test in the 3d simulation environment

The next step was the implementation of the controller into the vehicle software based ConSys [14]. This system can also be coupled with the virtual reality environment for testing the software system (fig. 6).

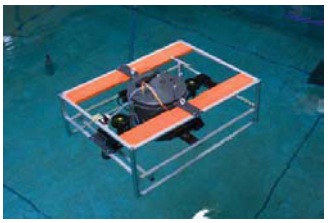

Fig. 9. ExAUV

This software system was deployed on our testing underwater vehicle *ExAUV* (Experimental AUV) (fig. 9) [15]. Therefore, the algorithms had to be adapted. The distance data is gained from the collision avoidance sonar and transformed to virtual distance sensors. This issue needed intensive testing with the sonar in the test basin to provide reliable data for the algorithm. Additionally, the algorithm implementation need to consider the low update rates of the distance data and is a prioritized state control algorithm:

The algorithm states with state 0 and is executed as loop.

1. state = 0
   (a) go forward to minimal distance
   (b) keep diving deep

(c) keep side distance

(d) Safety Check

(e) if minimal distance reached – change to state 1

2. state = 1

(a) keep diving depth

(b) turn vehicle

(c) if turned – change state to 2

3. state = 2

(a) count basin edges

(b) if forward distance is minimal – go to state 1

(c) otherwise – go to sate 3

4. state = 3

(a) keep diving depth

(b) keep side distance

(c) if reached – change state to 4

5. state = 4

(a) go up

(b) mission finalized

To avoid infinite loops, there is a timeout check in every drive state and sub drive state. If an Timeout is detected the sub drive state or drive state is decremented. For collision prevention the Safety Check is run at the end of all drive states and if necessary it changes the PWM Values to avoid impacts. So the drive state PWM Values will only be sent to the motor controller if there was no error or exception.

The algorithm was successfully tested. But different changes of the sensor configuration need to be made to improve the reliability of the algorithm.

The basin tests are also used for improvement of the simulation environment and for parameter estimation for the vehicle model.

5. CONCLUSION

In this paper a workflow for implementing coltrolling algorithms and for testing autonomous underwater vehicles during the complete development process was presented. The main tools of this workflow are simulation, a 3d environment and test basin. All experiences gained in the tests result in improvements of the control algorithms and the vehicles considering the presented development process models in this paper. This workflow helps to fasten the development process, to react on changing requirements during the development process and to improve the reliability of the algorithms and the AUV itself.

Acknowledgment

The authors would like to thank the German Federal Ministry for Economics (BMWi) for partially funding this research work under the project number 03SX262A. This project is with several partners from German industry, other Fraunhofer institutes and other German research facilities to develop an autonomous UUV for harbor and ship hull inspection.

6. REFERENCES

[1] P. Newman, R. Westwood, and J. Westwood, "Market prospects for auvs," *Marine Technology Reporter*, October 2007.

[2] Helmut Balzert, *Lehrbuch der Software-Technik: Softwaremanagement*, Spektrum, Heidelberg, 2. edition, 2008.

[3] Walker W. Royce, *Managing the development of large software systems: concepts and techniques*, August 1970, Reprinted in *Proceedings* of the Ninth International Conference on Software Engineering, March 1987, pp. 328–338.

[4] Kent Beck, *eXtreme Programming Explained*, Addison-Wesley, Reading Massachusetts, 2nd edition, 2004.

[5] "Das V-Modell," http://www.v-modell.iabg.de, June 2010.

[6] David Cohen, Mikael Lindvall, and Patricia Costa, "An introduction to agile methods," *Advances in Computers*, vol. 62, 2004.

[7] Denvan Wells, "extremeprogramming.org," http://www.extremeprogramming.org, 2000.

[8] "industrial xp," http://www.industrialxp.org, June 2010.

[9] Reinhard Höhn and Stephan Höppner, Eds., *Das V-Modell XT. Anwendungen, Werkzeuge, Standards*, Berlin, 2008. Springer.

[10] Thor I. Fossen, *Marine control systems : guidance, navigation and control of ships, rigs and underwater vehicles*, Marine Cybernetics, 2002.

[11] M. Jacobi, T. Pfützenreuter, T. Glotzbach, and M. Schneider, "A 3d simulation and visualisation environment for unmanned vehicles in underwater scenarios," in *52. Internationales Wissenschaftliches Kolloquium (IWK, International Scientific Colloquium) at Ilmenau University of Technology*, 2007, vol. 1, pp. 371–367.

[12] T. Glotzbach, T. Pfützenreuter, M. Jacobi, A. Voigt, and T. Rauschenbach, "CViewVR: A High-performance Visualization Tool for Team-oriented Missions of Unmanned Marine Vehicles," in *8th International Conference on Computer Applications and Information Technology in the Maritime Industries (COMPIT2009)*, 2009.

[13] "Ogre – open source 3d graphics engine," http://www.ogre3d.org, 03 2010.

[14] T. Pfützenreuter and H. Renkewitz, "ConSys – a New Software Framework for Underwater Vehicles," in *Oceans'10 IEEE Sydney*, 2010.

[15] Frank Weichert, Daniel Weber, and Andre Weiskopf, "Planung und Konstruktion eines Experimental-Unterwasserfahrzeuges ExAUV," 2009.