A Tool Chain for AUV System Testing

Marco Jacobi
Fraunhofer IOSB-AST
Ilmenau, Germany
Email: marco.jacobi@iosb-ast.fraunhofer.de

Thomas Rauschenbach
Fraunhofer IOSB-AST
Ilmenau, Germany
Email: thomas.rauschenbach@iosb-ast.fraunhofer.de

***Abstract*— Developing algorithms and parameterizing controllers for autonomous underwater vehicles are tasks which needs excessive testing. Therefore, we developed an environment and a tool chain for these difficult and mostly expensive tasks. This tool chain consist of different steps for implementing algorithms from the idea to mission ready. These steps include rapid prototyping, simulation testing, testing in controlled environments and finally productive testing. As an example we chose the development of a distance controller, which is required for inspection tasks.**

I. Introduction

> *An effective way to test code is to exercise it at its natural boundaries.* – Brian Kernighan

During the last time autonomous underwater vehicles (AUVs) gained more and more importance and will do also this in the future [1]. Main tasks of AUVs are exploration and inspection.

Unmanned underwater vehicles (UUVs) play a big part 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 autonomous underwater vehicle (AUV) which can handle these inspection tasks.

Like mentioned before, harbors, sluices and other underwater buildings need to be save 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 an 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 developed a tool chain to improve the development process and the reliability of AUVs.

II. Basic Development Process Models

All algorithms developed for Unmanned Underwater Vehicles (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 tool chain within these. These models can be classified as:

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

A. Waterfall Model

This model is a sequential development process. The progress can be seen as flowing downwards like a waterfall (fig. 1): if one step is finished the next can be started.

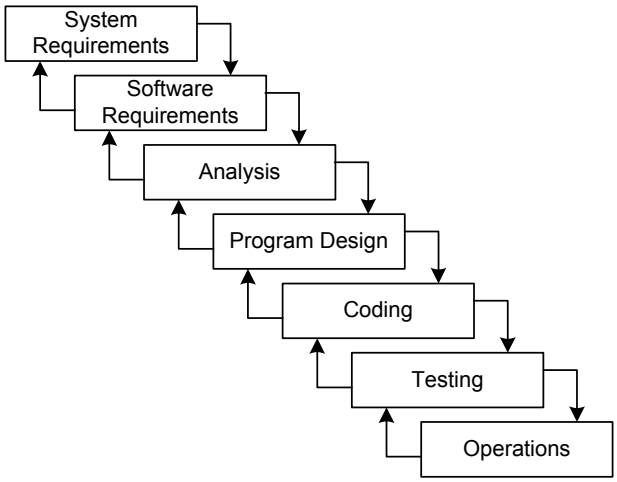

Fig. 1. Waterfall model

This model has its origins in the construction industries and can be considered as hardware oriented model adapted for the software development process.

The basic idea behind the waterfall model is: time has to be spend early to define the system requirements and the design in an absolute correct manner with the goal to save time and money later in the development process. This idea is also one of the great disadvantages: if the requirements are

978-1-4244-5222-4/10/$26.00 ©2010 IEEE

changing, design problems or implementation issues occur, the process has to be started at square zero. Therefore, 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 has clearly definable milestones, which provide a progress monitoring.

B. Extreme Programming

In [4], Kent Beck describes the 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 [7] and is based on communication, feedback, simplicity, courage and respect. 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 2.

One of the XP goals is to get an initial runnable implementation as quick as possible. Within these the foundation for continuing improvement is made. Generally, all features are implemented when required. Therefore, XP produces fast results at the beginning of the project also when most of the implementation issues and requirements are not yet known. That happens often during development of completely new systems or adapting existing systems to new requirements.

This process needs continuing communication with the customers and among the programmer itself. Also after every new implementation step and change like refactoring the system needs testing. The biggest drawback of this model is the lack of an overall design specification and also documentation since most communication are oral. Also for projects with an large number of developers it is not suited very well; an improvement therefore is industrial extreme programming [8].

C. V-Model and V-Model XT

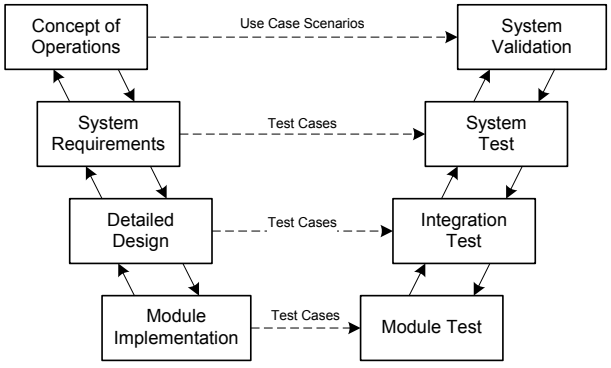

Fig. 3. V-Model

The V-model can be considered as an extension of the waterfall model to take the issue of quality into account (fig. 3). It introduces to each phase of the development process an associated phase of testing [2], [5]. The descending branch is similar to the waterfall model and shows the process from design to implementation, the ascending branch consists of the corresponding validation and test phases. Several German Federal Authorities use the V-Model for public projects and started enhancement the V-Model which was not updated since 1997, the result was the V-Model XT[1] [9].

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].

D. Combined Approach

All these philosophies have advantages and disadvantages. The idea is to combine these development models. 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 development of algorithms 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 and also 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.

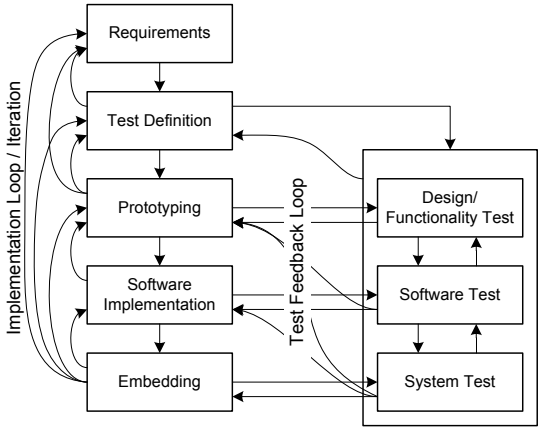

Fig. 4. Combined Model

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 has 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. 4).

[1] eXtreme Tailoring

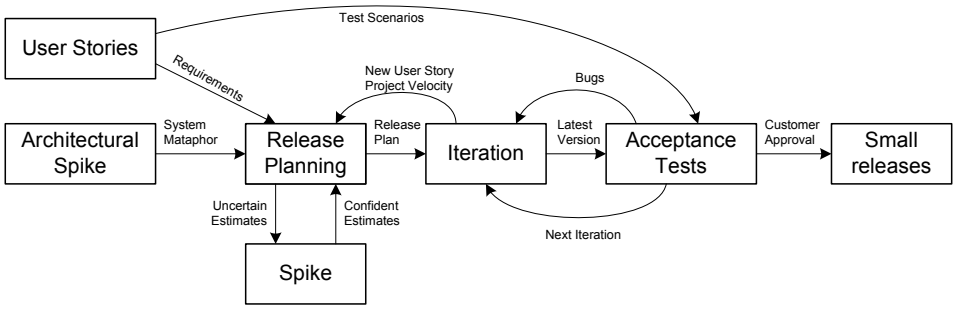

Fig. 2. Extreme Programming Workflow [6]

III. The Tool Chain

In order to obtain a high software quality and safe algorithms for underwater vehicles especially for autonomous ones, all algorithms has to be tested extensively. Therefore, different tools were developed. The most significant tools in our system are a simulation environment and a test basin.

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 basic controller design. 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. 5).

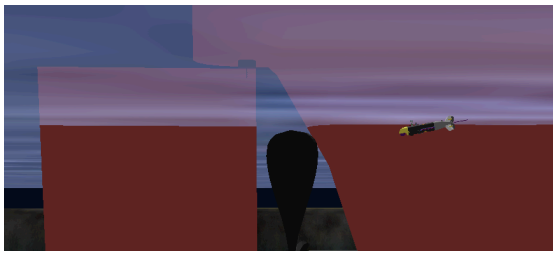

Fig. 5. 3d-Environment – ship inspection

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.

A. Simulation Environment

For rapid prototyping of algorithms different tools can be used. One of the most used 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.

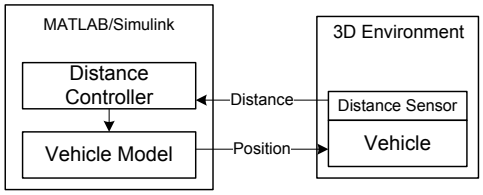

Fig. 6. Interaction VR and MATLAB-model

For vehicle simulation in complex environments with different sensor combinations like sonar (fig 7) or distance senors these dynamic models and also the implemented algorithms can be coupled with a 3d virtual reality environment (CViewVR) (fig. 6) [11], [12].

Fig. 7. 3d-Environment – ship inspection

This 3d environment is based on open source tools which are well documented, in active development, widely used 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.

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. When needed 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 an 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 8).

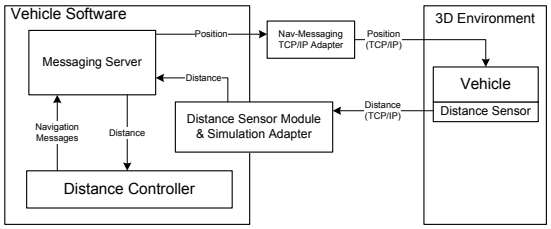

Fig. 8. Interaction VR, simulation and vehicle software

The vehicle control and navigation modules should be the same used in real mission in the vehicle itself. The algorithms developed in the modeling tool (e.g. MALTAB/Simulink) 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.

B. Test Basin

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 control software with real sensors in near real mission environment, the Fraunhofer Application Center System Technology build a basin (fig. 9) for testing where different inspection tasks can be done within a controlled environment.

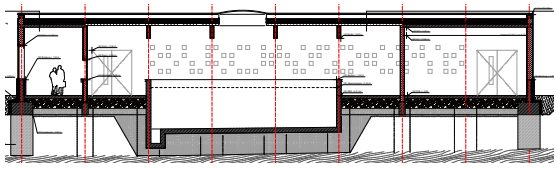

Fig. 9. Test basin

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 an 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 takes 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.

IV. Example

As an example for an appliance of this tool chain, a controller for distance keeping used in harbor inspection. During inspection for instance of sheet pile walls or ship hulls. The controller works with only data from distance sensors and also current vehicle position and maneuver data will be used.

The workflow is separated into these steps:

- Requirement specification,
- Test scenario definition (3d scene definition),
- Distance sensor model definition (3d scene),
- Simulation environment set up in MATLAB/Simulink,
- Controller definition and implementation in MATLAB/Simulink,
- First Controller tests,
- 3d simulation environment set up,
- Running tests and validate the controller setup in the 3d simulation with sensor feedback
- Test and validation in the test basin and
- Sea trials.

Firstly, for the controller the basic requirements are defined: during a scan of a wall or ship hull keep a specific distance to this wall. This distance depends on the requirements of

the used sensors for inspection. Additionally, a collision with obstacles or the wall should be avoided.

With this requirements a scenario for testing was defined and an 3d scene for the simulation environment built (fig. 5). With the specifications of the used distance sensors the sensor simulation models were set up. The vehicle sensor configuration consists of two pairs of sensors for each side of the underwater vehicle.

The MATLAB simulation environment consists of a kinematic vehicle model with a specified interface and is easily exchangeable to a dynamic model with a wide variety of simulation details. There are also interfaces to the 3d environment simulation which is an external tool and provides the sensor simulation.

In this example a controller for the port-side is implemented using the distance from the sensors provided by the environment simulation as shown in figure 10.

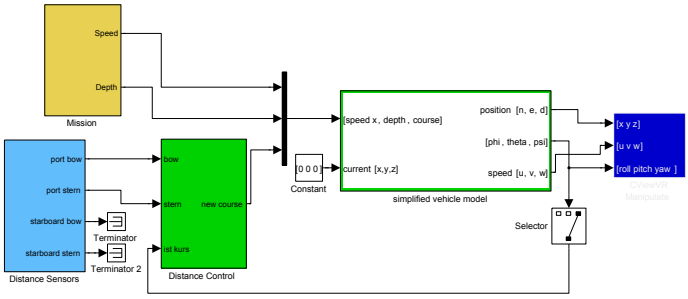

Fig. 10. Distance controller test in the 3d simulation environment

The controller uses the distance data from both sensors. It calculates the course correction value to keep the distance from the inspection object and to gain a parallel orientation to the object.

The next figure illustrates a test run with the environment simulation (fig. 11).

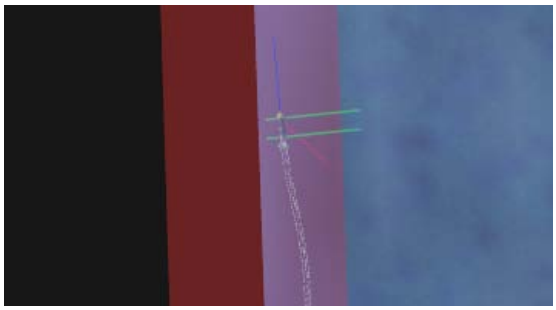

Fig. 11. Distance controller test in the 3d simulation environment (top view)

After successfully testing the algorithms within the 3d simulation environment. The algorithms are implemented in the vehicle software and continuously tested with the simulation environment and compared with the results of the MATLAB simulations.

When the implementation of the algorithms are complete and the tests successfully executed then the software will be tested in the basin as as described in this paper earlier. Currently, the software is only tested in the simulation environment because some basic functions are not implemented at the moment. Several tests in the basin are planned to find errors in the implementation, to discover security issues and to improve the fault tolerance of the complete system.

V. Conclusion

In this paper we presented a tool chain for testing autonomous underwater vehicles during the complete development process. The main tools 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 tool chain helps to fasten the development process, to react on changing requirements during the development process and to improve the reliability of the AUVs.

Acknowledgment

The authors would like to thank the German Federal Ministry for Economics (BMWi) for funding this research work under the project number 03SX262A. It is a project 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.

References

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

[2] H. Balzert, *Lehrbuch der Software-Technik: Softwaremanagement*, 2nd ed. Heidelberg: Spektrum, 2008.

[3] W. 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. [Online]. Available: http://www.cs.umd.edu/class/spring2003/cmsc838p/Process/waterfall.pdf

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

[5] (2010, 3) Das v-modell. [Online]. Available: http://www.v-modell.iabg.de/

[6] D. Wells. (2000) extremeprogramming.org. [Online]. Available: http://www.extremeprogramming.org

[7] D. COHEN, M. LINDVALL, and P. COSTA, "An introduction to agile methods," *ADVANCES IN COMPUTERS*, vol. 62, 2004.

[8] (2010, 03) industrial xp. [Online]. Available: http://www.industrialxp.org/

[9] R. Höhn and S. Höppner, Eds., *Das V-Modell XT. Anwendungen, Werkzeuge, Standards*. Berlin: Sprinter, 2008.

[10] T. 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*, vol. 1, 2007, 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] (2010, 03) Ogre ñ open source 3d graphics engine. [Online]. Available: http://www.ogre3d.org/

[14] T. Pfützenreuter and H. Renkewitz, "Consys – a new software framework for underwater vehicles," 2010, will be published at Oceans 2010.