The Virtual Test Bed: Simulation Supporting Software Development Testing

Tien Nhan
Simis.inc
Portsmouth, VA
Tien.nhan@simisinc.com

Vernon Hayden
Simis.inc
Portsmouth, VA
Vernon.hayden@simisinc.com

Jesse Barboza
Simis.inc
Portsmouth, VA
Jesse.barboza@simisinc.com

ABSTRACT:

Software development is a recursive process of development, testing, and revision; all of which take up time, money, and man power. For companies that have limited budgets and timelines for completion, simulation techniques can assist in software and hardware testing and verification. This paper focuses on the development of software using simulation and references the Riverscout case study as a means to show the effectiveness of using simulation during software development. The highest priority of successful testing is the need to replicate the behavior of the system within a virtual environment. This is accomplished through our SimArchitechture and comprehensive data collection and modeling techniques. Often times, testing hardware components that integrate through the software can be time consuming and deteriorate the hardware over time. In order to reduce time and cost, a simulation can be constructed to represent the hardware, where the simulation then becomes the test suite for the software being constructed. This paper shows how simulation aided the Riverscout project's software development process by reducing cost, development time, and deterioration to hardware used in the project. Finally results and conclusions are presented along with how expanding the use of simulation can be effective in other projects using software.

ABOUT THE AUTHORS

Mr. Tien Nhan is a Software Engineer with SimIS Inc. He graduated from Old Dominion University with a Bachelor's Degree of Science in Modeling and Simulation. Currently he is also attending ODU for his Masters of Science in Modeling and Simulation. His areas of interest include Transportation Research, Robotics, and software development. Mr. Nhan's focus during this project is on functionality, development, and testing of software.

Mr. Vernon Hayden is an analyst with SimIS Inc. and graduate student studying applied mathematics at ODU. Working primarily with simulation environments, his mathematics background aids in the development and validation of data-driven models to power those applications. In his spare time, he enjoys submitting algorithmic solutions to high computation challenge problems, deriving mathematical models and mechanical solutions for game development, and senselessly working towards a closed-form derivation of Apéry's constant.

Mr. Jesse Barboza is a Senior Business Systems Analyst at SimIS, Inc. He is responsible for the business/systems analysis of Information Systems development and innovation of upcoming concepts of Modeling and Simulation solutions. Mr. Barboza brings experience to the development of information technologies, software development along with project management to his project teams. Mr. Barboza has a B.S.B.A. from Old Dominion University (ODU) where he majored in Information Systems and Technology with a minor in Business Analytics. Mr. Barboza is currently pursuing a Master's of Information Technology from Virginia Tech.

1.0 – INTRODUCTION:

Software development is a recursive process with testing as a major component. Testing can be time-consuming and intensive on hardware components that are needed for the project. Standard Modeling and Simulation (M&S) techniques can be used to significantly reduce time and costs of testing and make solutions more stable and reduce risk by providing a richer testing environment. In 1999 Alan M. Christie among others introduced simulation as an enabling technology for software development. [2] In the underlying project, software was developed for a semi-autonomous water surface vehicle and tested under the various operational and environmental constraints. The team used M&S to assess the autonomous vehicle's intelligence algorithms, perform trade studies, and exercise algorithms under virtual, yet realistic, conditions far beyond the possibilities of only physical prototypes.

There are several levels of autonomy, from being controlled by remote controllers with humans fully in charge, to a complete autonomous system that perceives its environment, makes its own decision, and acts on its plan without any interference of a human controller. The United States Navy is in stronger favor of keeping a human in the loop to be the ultimate control instance; therefore a Tactical Robotic Controller (TRC) is needed to give the sailor access to the controls of the robot. M&S is a strong contributor in testing the functionality, controls, and logic of the TRC. Unmanned hardware can include unmanned surface vehicles (USV) or unmanned aerial vehicles (UAV); both of which can be especially expensive and susceptible to accidents during development. Across the world simulation are aiding autonomous systems by providing a test suite for development. [1] Wang Hong-Jian supports the claim that simulations are a proper tool for software development by presenting a case study on autonomous underwater vehicles. The software development for such important and expensive hardware needs to be tested thoroughly before physical testing. However, without testing the software on the hardware, the developers are unable to determine if the software's functionality is correct.

Simulation is a viable tool for autonomous and semi-autonomous systems, and as a tool, reduces the amount of testing on the hardware. Since most developers have an idea of how the software should be working with the hardware, the hardware can be represented with a Virtual Test Bed. The Virtual Test Bed can substitute for the environment and/or hardware components by being represented in a simulation, the software then is able to be tested without the need for the physical system. Test cases can be made and tested in the simulation, thereby reducing the amount of testing needed in the real environment. When the developer has confidence that the software is working correctly, the physical system should be tested and any data should be extracted. In this paper a two-step process of testing is presented; software tests with the Virtual Test Bed, and software testing with the real system.

Without testing the software in the simulation, unknown problems can occur which could prove detrimental to the hardware. An example is controlling motion of the USV or UAV; the simulation can provide insight into how well the USV or UAV moves based on the controls developed in the software. Without proper testing, the movement can be erratic and cause the physical system to behave improperly and in the worst case scenario damage the hardware. The deterioration of the hardware can be limited by extensive testing in the Virtual Test Bed. While the replication of movement in the simulation is not accurately represented at first, basic concepts can be tested and once data is retrieved from the physical system, the Virtual Test Bed can be updated allowing more certainty of movement. Movement of an USV is first based on basic movement of the craft in the water. However, after several rounds of testing and data extraction, a new model for movement of the USV can be analyzed and developed thereby making a better representation of the USV in the simulation.

The paper first introduces the Riverscout Case Study both in terms of hardware and software as well as issues found during development. The Riverscout team developed a virtual testing environment to enhance training and material testing through simulations. The methodology for the use of simulation within our case study is presented along with the simulation architecture. A section on data collection follows and how it improved development. Section four presents the calibration process and data analysis on the movement of the USV within our case study. Finally, conclusions are shown along with extensibility and future work. The purpose of this paper is to show how simulation as the Virtual Test Bed supports software development represented by the case study.

2.0 – Background:

Simulation is a recognized tool that can be used to facilitate software development. [3]
The Riverscout project includes a water-borne USV and a TRC which runs a Graphical User Interface (GUI) software that presents the user with important information about the hardware, an interface to control the movement of the USV, and the capability to switch between modes. Some information that is shown includes viewing internal components of the TRC or USV, such as: battery life, TRC's GPS, current location of the craft, velocity, current heading, and status of communications between the USV. The TRC has buttons that allow the user to increase or decrease the speed of the USV and with the GUI the user can change the functionality of the craft through different modes allowing it some autonomy to full autonomy. The modes include cruise control, manual, idle, loiter, come home, and route. In Riverscout, the physical system includes the USV, the environment, and the TRC. The virtual system utilizes a simulation in order to replicate the USV and environment.

The TRC allows for mission planning that permits the user to lay down waypoints in a Geographic information system (GIS) and "save" them as missions. The user can then use route mode to direct the craft from waypoint to waypoint. Mission Planning also allows the user to view the craft's GPS location from a GIS map. Cameras on the USV allow the user to view the craft's perspective without the craft being in sight. Many concepts of the TRC software were able to be tested in the Simulation without the use of the USV, which proved extremely useful when the USV's hardware was experiencing some difficulty.

2.1 – Issues with Testing:

Testing in the simulation reduces the amount of testing in the physical system; therefore increasing the lifetime of the craft by decreasing exposure to environmental factors over the lifespan of the testing period. Instead of testing every concept of the software in the real environment, there are components of the TRC software that can utilize the simulation in order to test software concepts. Protecting the lifetime of the hardware helps reduce cost, and utilizing the simulation saves developers' time and money since developers can simply test on their own development computers instead of exposing the USV to harsh environmental factors. If the simulation replicates the hardware accurately, then the simulation allows developers to have some understanding of how the software reacts with the physical system.

The Riverscout case study includes extensive field testing for the USV in order to ensure movement and mission planning are efficient. Field testing can be intensive on the hardware and also increases the amount of manpower usage and travel costs. Testing the movement of the USV is necessary and was replicated in the simulation; however, the accuracy of the movement in the simulation is based on real world data. As such, recursive testing improves the movement in the simulation. When real-world data of the USV's movement matches the simulations movement data, the fidelity of the simulation increases and developers build confidence that the simulation tool is correctly representing the USV. Once confidence in the simulation is built, new concepts for software development can be added and tested utilizing the simulation. The new concepts can then be tested in the real-world environment.

Simulation allows for increased testing capabilities. M&S allows testing of extreme environments that may be too dangerous, costly, or not possible to observe and allow the collected data to be mapped to the established meta-models of the standardized system. Current testing methods include physical testing of logical concepts, modules, and boat movements in a real physical environment. However, Riverscout introduces simulation hardware-in-the-loop testing. In a hardware-in-the-loop test, testing of the hardware is done in the simulator, and then tested with real hardware, and this process is repeated. Testing is done using the simulation environment and the TRC software, without need of the real physical environment. This project initially utilizes a "bottom up approach" to include M&S capabilities that help certify engineering performance of a chassis and sensor packages, along with developing techniques that will incorporate high fidelity into the perceived models that are embedded into the simulation through real world data driven and physics based objects. The government will then be able to proceed with a "top down" approach to certification of the performance that decomposes high level mission requirements into parts of the developed Virtual Test Bed.

3.0– SIMULATION AND THE RIVERSCOUT CASE STUDY:

3.1– Simulation Architecture:

Early into the project, hardware outages, maintenance schedules, and in-climate weather underscored the demand for an aptly modularized simulation infrastructure. Unfortunately, accomplishing this required the virtualization of three distinct physical domains: the craft, the controller, and the environment. Moreover, both the craft and the controller would require wholly independent functionality, as well as completely imitate the live, physical system -- without exception.

Simplicity and familiarity both aided development and implementation of the overall simulation architecture. The craft's firmware already existed as a C++ project, the controller's software was executable on any desktop, and the testing environment was easily reproducible in Unity -- a software suite already familiar to each team member. Utilizing these conveniences, the team built a series of complementary simulation modules in place of physical entities (i.e. GPS readings, craft dynamics, networking) and built a dynamically-linked-library (DLL) replica of the craft, capable of interfacing with both the Unity environment and the controller software simultaneously. Once the craft's virtual replica had been constructed, interfacing with the Unity environment required little more than connecting the appropriate DLL-interfaces with their respective GUI controls and rendered representations.

Although the craft simulation requires the Unity program to execute, it should be emphasized that the Unity's scope is deliberately limited to environmental aesthetics and structural variable manipulation. The environment itself acts solely as a trigger for (and limited window into) an operational craft simulation -- already controlled by the TRC during both simulated and physical manifestations. Likewise, the DLL accepts parameters from the environment and continually communicates with the controller simultaneously, but the intersection of the environment's influence and the controller's capabilities must be null to maintain necessary modularity.

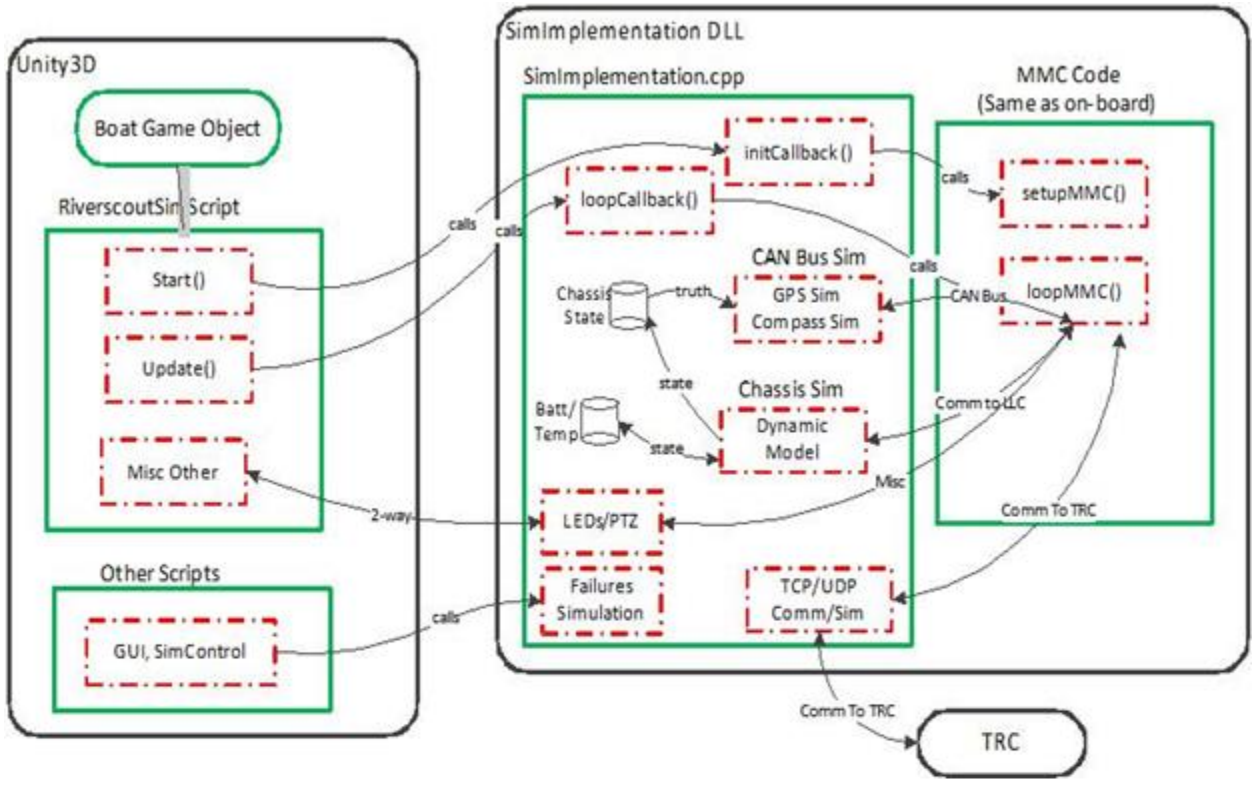

Figure 1: Simulation Architecture

Figure 1 represents the Riverscout's simulation architecture. The simulation architecture utilizes the actual software routine onboard the physical craft, with Unity playing a deceptively limited role overall. On initialization, Unity triggers the execution of SimImplementation, an external dynamic-link library (DLL) controlled by the TRC software through a user datagram protocol (UDP) network. Once SimImplementation begins its routine, Unity repeatedly calls for updates on craft state and variable information within that DLL, changing the visual environment according to changes within the onboard code and simulated modules. In addition to observing communication between the TRC and a simulated Riverscout craft, the simulation architecture allows users to manipulate actual values and operations within the onboard code during the simulation's operation in real-time, such as observing the residual effects of injected radio delays or outage.

During initial simulation development, analysis of live test data and operational observations revealed hardware-related issues related to craft dynamics. To avoid developing a data-generated model relevant only to outdated hardware, (in the interest of time) the team developed a simple second order linear state-space model within Matlab, operating within the simulator's implementation DLL. This simple model operates under the same modular principles guiding Riverscout's overall development and serving as a placeholder for more advanced future models once hardware updates complete. For model validation and verification (V&V), the simulation utilizes the TRC's data logging capability, exporting simulator-generated data files for comparison with live data.

3.2 – Networking TRC to the Simulation:

The simulation acts as the virtual environment and virtual boat, the information must be sent to the TRC in order to replicate real-time GPS and craft mechanics. A communication protocol is needed to facilitate communication between the TRC and the simulation environment in Unity. The simulation incorporates a User UDP C# Winsock application to establish communication. Once a network is established between the TRC and the simulation, packets are continuously sent back and forth representing actual data being sent back and forth from a real craft and the TRC. Due to our own packet protocol structure, a UDP network is chosen over a TCP network; a TCP network has a standard packet order and for the Riverscout purposes, a custom protocol order is preferred. Also, a UDP network is a lightweight, connectionless communication, allowing for fast communications with no need of acknowledgement, instead only an established connection. **Figure 2** shows the three communication protocols used for testing and developing the TRC, USV, and Simulation.

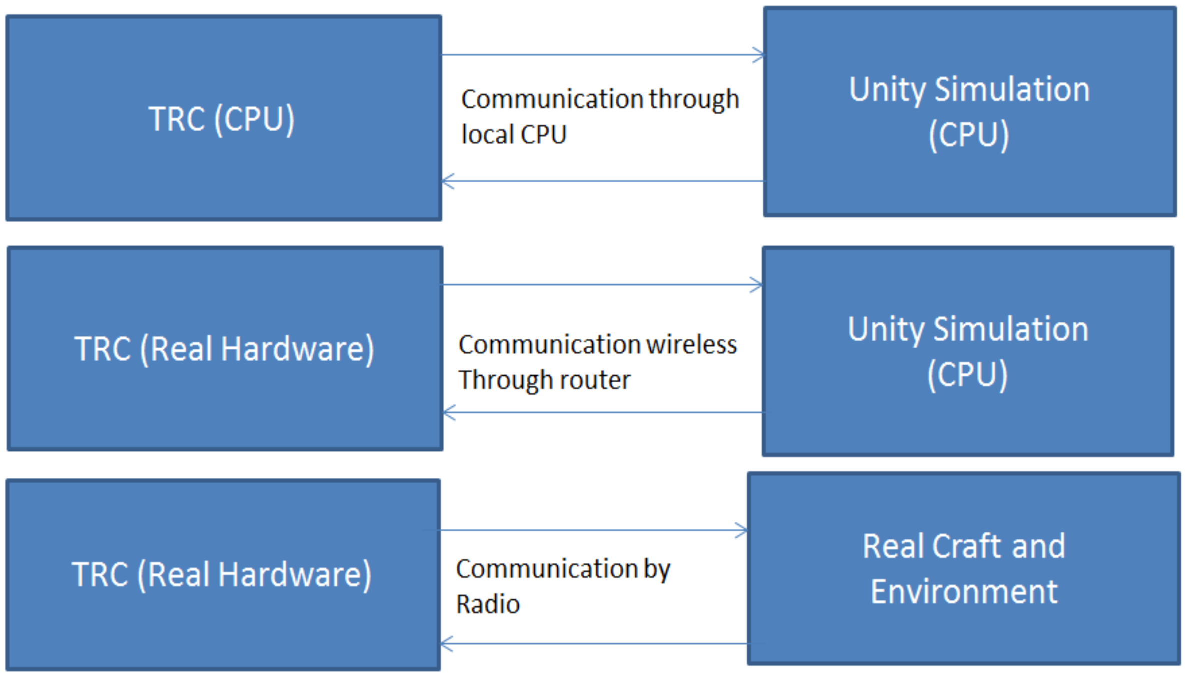

Figure 2: Communications Protocols for the Rivescout Project

3.3 – Methodology using Simulation:

In the Riverscout project, time and money was saved due to testing the TRC's software development with the simulation as a tool. The virtual test-bed allows concepts and platform changes of a USV to be rapidly tested. Proposed changes to hardware, weight, the position of cameras, and boat movement dynamics are experimented within a controlled virtual environment to provide feedback on effects to the autonomous system. By experimenting with the parameters, the simulation can verify the USV as operational and ready for commercial applications such as ocean security, patrols, and surveillance. Testing is conducted in two phases; the first being testing done with the simulation in order to build confidence that the software is working correctly. The second phase of testing includes recursive testing in the physical environment.

Test cases for changing modes can be represented in the Simulation. The USV in the simulation went from cruise control and idle as the TRC's command was sent. The USV also came to a complete stop allowing the developers to note that the command to move forward had indeed stopped being sent. Loiter involved a command being sent that the USV should use a certain amount of velocity and a specific turn radius in order to move in a circle continuously until a different mode is selected. Cruise control allowed the user to set the velocity at a specific rate. All these concepts were able to be represented in the simulation, since they involved commands being sent from the TRC to the simulation that represented the USV. The testing gave the developers confidence that the TRC and USV would perform in a similar behavior in the real environment.

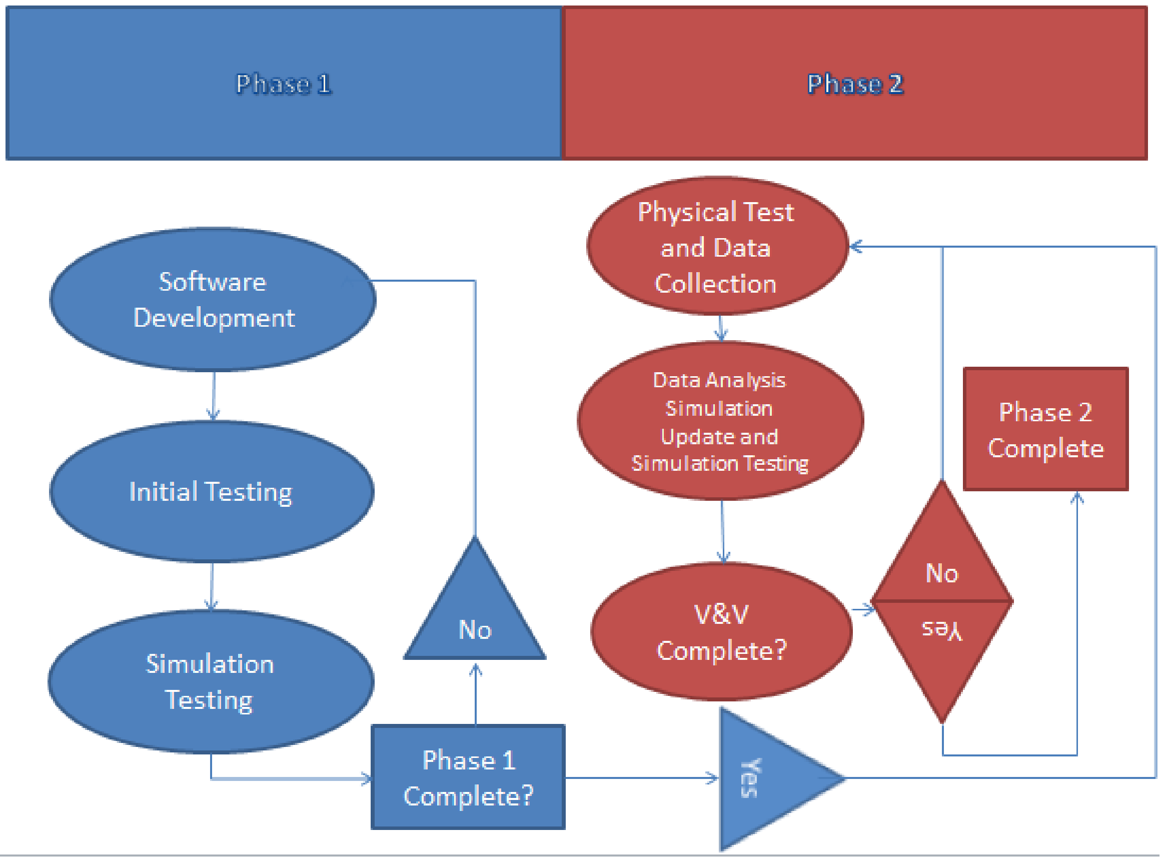

Figure 3: Phase One and Phase Two Testing Model

Figure 3 shows a model of phase one and phase two. Phase one displays initial software development and testing. Initial testing involves bench testing communication between the USV and TRC. The software components of the TRC are then tested with the simulation. Phase one is recursive until the developers are satisfied with the software's features and capabilities. Phase two begins by testing on the physical system and collecting data. After collecting the data, an analysis of the data is constructed and the TRC's controller is adjusted as needed along with the movement of the USV in the simulation. Phase two is repeated until all stakeholders are satisfied.

The simulation allows for other testing capabilities in our case study. One aspect was importing new GIS maps. As the USV can be deployed anywhere, available maps are needed. The simulation is able to represent the USV's latitude and longitude anywhere thereby letting the TRC read the represented location and display the correct imported map. Confidence building in the TRC software was essential to begin phase two. Bench testing helped build confidence in the communication process between the TRC and USV. Simulation testing built confidence in the TRC's capability to control mode selection and movement. Finally, testing all essential components that do not require the real USV are done to complete phase one.

Phase two involves testing in the real environment and data collection. Simulation is a part of phase two because the Simulation's USV movement is based off a Mathematical Model taken from water-craft movement. However, by recording data taken from the real live testing of the USV, the simulation craft's movement can be updated by analyzing the data collected. This allows the developer to both test the USV in the real world and improve the simulation for future development. Each time testing is done in the real world environment; data is collected and analyzed. Adjustments to the virtual USV in the simulation are then made in order to improve accuracy of the real USV's movement. The improvement to the simulation's precision aids the accreditation process of the simulation for future development work.

Simulation is a tool that can be used in order to debug software during development in order to limit potential software crashes. In Riverscout issues with the TRC not functioning correctly occurred during testing with the physical system. These issues were recorded and reproduced in the simulation in order to narrow down where the bug is being thrown. By observing steps that led to the TRC software's crash and recreating the issue utilizing the simulation, developers gain a test case that can be repeated in order to diagnose the issue. Once repeatability is established, the developer can isolate the problem in order to find the code that causes the crash. Developers are then able to reproduce the issue and test until figuring out the direct reason for the TRC software's crash. Instead of testing in the real physical system in order to fix software issues, software issues are fixed with the simulation in hopes that the issue is then fixed in the physical system test.

4.0– CONTROLLER CALIBRATION FOR THE TRC AND VIRTUAL TEST BED:

Calibrating the controller of the TRC is necessary in order to ensure the USV is moving accordingly. In order to correctly calibrate the controller, recursive testing in the physical system is needed. Live physical testing provides Data Collection capabilities that can be analyzed to determine the differences between controlling movement in the simulation and in the actual environment. As data on the actual USV's movement is extracted and analyzed, the TRC's controller can then be adjusted to account for differences between the simulation's USV and the real craft. By witnessing physical testing, the team observed that at times the USV's turn rate was either too low or too high making the USV's movement erratic. By understanding how much the turn rate actually was, the TRC's controller can be adjusted to account for the extra turn rate. Incrementing the speed of the USV is updated in a similar manor to the controls of the turn radius. The TRC's capability to increment the speed of the USV is first based on the simulation, but as data is collected and updated in the simulation, the TRC controller updates the amount of incrementing speed in order to better represent the real USV.

The Virtual Test Bed allows for early observations on the behavior of the TRC software and basic USV movement. Early observations allowed the team to have confidence that the TRC's simple functionality, such as forward movement and turning, was working correctly. During initial testing, the TRC's controller was adjusted with respect to the simulation's USV movement. As real world testing was being done, the Data Collection allowed the team to adjust the virtual USV's movement to more accurately depict the craft's real movement. After updating the simulation, testing was done with the simulation in order to update the TRC's controller as needed. Testing is recursive with each iteration improving the simulation's capability to represent the USV's movement allowing developers to update the TRC's controls accordingly. The calibration process can be shown in simple steps. Step one is utilizing the simulation for basic movement and basic TRC controls. Step two involves live testing in order to collect data on actual USV movement. Step three involves analyzing the data in order to update the simulation's

USV movement. Finally, step four allows the developers to update the TRC's controller to match the new movement model in the updated simulation. Steps one through four are repeated until the team and stakeholders found that the TRC's controls are appropriate. The process to update the TRC's controller by utilizing movement models is known as Model Based Testing and explained thoroughly in [4].

4.1– Modeling Craft Dynamics with Limited Data

Though touched on earlier, the challenges associated with modeling dynamics of a small, twin-jet USV merited some additional discussion. In-climate weather aside, unexpected hardware consistently plagued the testing schedule and limited the opportunities for data collection and subsequent time allotted for investigating the system more rigorously. Given these constraints, the team decided on a simpler approach employing dual state-space models for the systems in question.

When modeling the craft, the models needed to reflect the "planing" phenomenon -- a situation where the craft's nose lifts upwards at higher speeds, reducing surface contact and boosting the craft's velocity further. As the team already had a fair amount of low-speed (non-planing) data relating percent engine effort to craft velocity, Matlab generated a relatively accurate low-speed linear state-space model. Given the nature of linear state-space models, discontinuities in the domain space still map to continuous representations in discrete time -- this motivated the use of a front-end adapter.

Deriving the front-end non-linear conversion utilized a data-driven approach. Once the low speed models had been implemented and tested, Matlab scripts processed the physical craft's test data files, recording average speed and yaw rate at each available engine percent after a parameterized steady-state duration. The same script would then output lines of firmware code for a piecewise-linear mapping of engine percent to its appropriate output, dividing that tabulated value by the corresponding low-speed model's steady state unit output to produce the necessary effort input. This front-end modification produced models accurate enough to successfully calibrate the craft's PID controllers largely through simulation alone, naturally with minor adjustments being made on the live craft.

5.0– CONCLUSIONS

The Virtual Test Bed improved the development process of the Riverscout software. Without utilizing the simulation, phase one would be limited to bench testing communication between the craft and the TRC. The team would have no idea how well the controls developed would perform with the actual USV. Instead the TRC's modes and controls were tested utilizing basic water-borne craft movement in a virtual environment. As phase two began and testing in the real environment was performed, data collection allowed the team to analyze the USV's actual movement. Then the data was compared to the initial movement data model, and a new model allowed the simulation to improve the virtual craft's movement to accurately represent the real USV. As the precision of the simulation improved, new concepts are easily introduced to the TRC software. The simulation lacked accurate craft movement, but as real world testing continued, the accuracy of the simulation's craft movement increased.

Simulation as a tool provides other benefits that are not exclusive to software development. Training with the simulation helps users become familiar with the TRC's controls before operating the real USV. Since the USV is expensive, users should build confidence with the simulation in order to avoid causing unnecessary harm to the USV. Another potential is utilizing the simulation in order to depict different USV's. A USV can have a different size, weight, and engine power thereby changing the movement of the craft. By adding the size, weight, and engine power as parameters, different USV's can be represented in the simulation. Thus providing a commercial application, where USV developers can utilize the TRC as a controller in order to see how different characteristics of the craft affect movement. Simulation is a viable tool that extends extra capabilities to software development.

REFERENCES

Christie, A.M. (1999) 'Simulation – an enabling technology in software engineering', *Technical Article, The Software Engineering Institute (SEI)*, Carnegie Mellon University, http://www.sei.cmu.edu/publications/articles/christie-apr1999/christie-apr1999.html.[2]

Binder, Robert V., Bruno Legeard, and Anne Kramer.(2015) "Model-Based Testing: Where Does It Stand?" *Communications of the ACM* 02/2015 58.No. 2 (2015): 52-56. [4]

Hong-jian, W., Xiao-cheng ,S., Jie, Z, Juan, L., Ming-yu, F. (2004) A Semi-physical Virtual Simulation System for AUV. IEEE, Volume 3, Pages 1560-1563 [1]

Spasic, B., Onggo, B. (2012) Agent-based simulation of the software development process: A case study at AVL. IEEE. Simulation Conference (WSC), Proceedings of the 2012 Winter. Pages 1-11 [3]

Binder, Robert V., Bruno Legeard, and Anne Kramer.(2015) "Model-Based Testing: Where Does It Stand?" *Communications of the ACM* 02/2015 58.No. 2 (2015): 52-56. [4]