Abstract of Presentation at AlaSim 2014

Multiagent Test Range: Fostering Disciplined Software Engineering Practices in Students via Modeling, Simulation, Visualization, and Analysis

Dan Tappan, Eastern Washington University, dtappan@ewu.edu

The testing strategy of typical undergraduate software engineering students is a shotgun approach of unfocused, nonrepeatable tests of questionable rigor and value. Testing is an ad hoc afterthought because they have no experience with developing a disciplined, rigorous test plan, a formal methodology to carry it out, and a persuasive means to demonstrate the results. This pedagogy-oriented system showcases the highly successful deployment of a richly extensible, student-friendly Java integrated modeling-and-test environment for discrete-event simulation of component-based agents within a virtual test range based on underlying models that are well defined, connected, executed, and evaluated in controlled experiments through scientific method.

Derived from the principal investigator's role as lead systems engineer and software architect for accredited modeling and simulation projects at White Sands Missile Range, Aberdeen Proving Ground, Electronic Proving Ground, and elsewhere, this system provides an accessible, student-friendly architecture for rigorously integrating continuous design, implementation, and testing of agent-based software with consideration toward verification, validation, and accreditation.

The model-view-controller architecture clearly separates its concerns: the model is what agents are and can do, the controller or simulation is the behavior of the model in operational scenarios of interest, and the view is real-time two- and three-dimensional visualization of the test range, as well as extensive logging of internal data for quantitative postanalysis. It capitalizes on familiar software design patterns. Specifically, the creational aspects address defining agents and their subcomponents; i.e., actors (ships, airplanes, and submarines), unguided and guided munitions (bombs, shells, depth charges, torpedoes, and missiles), and active and passive sensors and fuzes (radar, sonar, thermal, acoustic, depth, distance, and time), all with a variety of performance characteristics (maximum speed, acceleration, rate of turn, fields of view and regard, power, sensitivity, yield, blast radius, etc.). The structural aspects address hierarchically connecting them in an appropriate plug-and-play manner for the desired mission/experiment-specific load-out. Finally, the behavioral aspects address directly controlling course, altitude/depth, and speed of the actors, as well as indirectly controlling the munitions to deploy whenever and however appropriate. Metacommands support defining and executing repeatable experiments and managing the breadth and depth of results.

Even this relatively small set of agents and properties leads to an intractable combinatorial explosion of test cases for both correctness of the model code and its performance within the test parameters of the battlespace. The mitigating approach closely aligns with concepts and practices of software quality assurance taught in parallel, especially critical thinking applied to compositional, functional, and integration tests that capture representative behaviors and combine in defensible ways to contribute to meaningful, persuasive test reports with annotated and narrated data and graphs that compare and contrast expected versus actual results. In addition, it provides practical grounding to many abstract concepts in the required discrete mathematics, probability, and statistics courses that students perceive to be mostly irrelevant to their major.