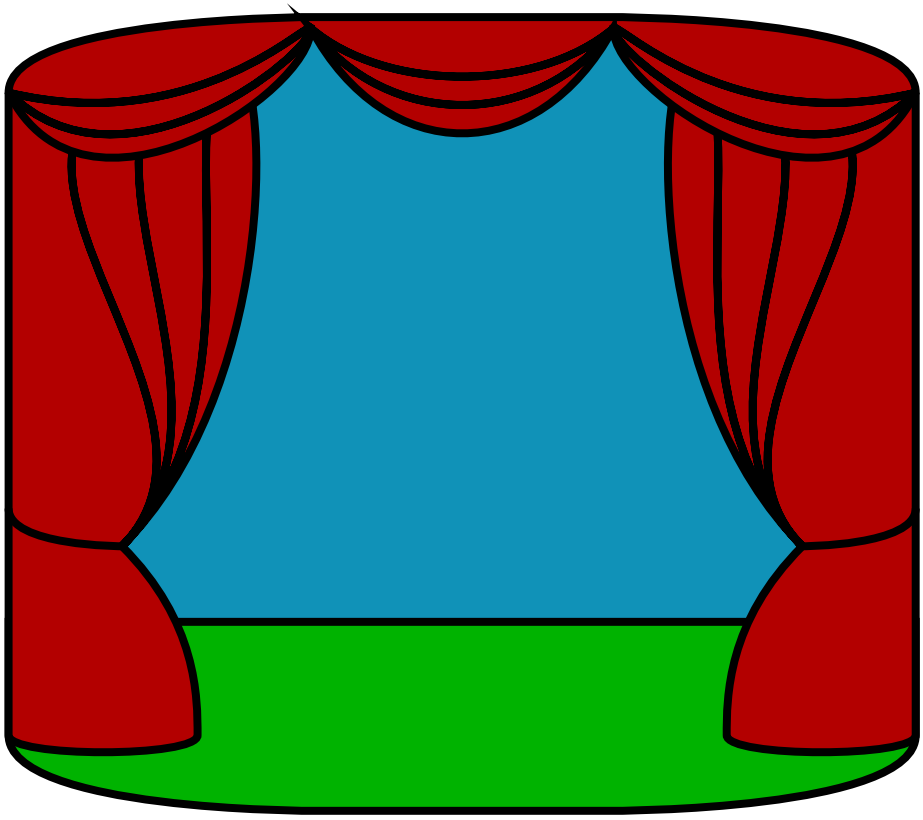

STAGE

Acceptance Test Plan

Frank Clark
francis.j.clark@drexel.edu

Dustin Ingram
dustin.s.ingram@drexel.edu

Maria Kolakowska
maria.j.kolakowska@drexel.edu

Aaron Rosenfeld
aaron.rosenfeld@drexel.edu

January 4, 2011
Version 1

Abstract

STAGE is an event-based simulation environment used to compare the effectiveness of different combinations of software agents, network configurations, and sensor data in real-world environments. It is comprised of a distributed simulation engine, visualizer, and programming interface through which developers create agent software and network topologies. Communication between virtual nodes is also simulated, providing highly realistic scenarios.

Contents

1 Introduction

1.1 Objective

This document provides the plan for completing the testing activities required for the Acceptance Test Plan of STAGE. The acceptance test plan for this system is based on the project requirements set forth in the Software Requirements Specification document.

1.2 Structure of the Document

- Section 2 describes the overall approach to the Acceptance Test Plan process.
- Section 3 describes in more detail individual issues covered or not covered by the Acceptance Test Plan process.
- Section 4 describes the criteria which have to be satisfied for the Acceptance Test Plan project.
- Section 5 describes the roles and responsibilities of the staff members involved in the Acceptance Test Plan project.
- Section 6 describes the test cases used during the Acceptance Test Plan.

1.3 References

For the STAGE project requirements, please see the Software Requirements Specification document for this project. The most recent version of this document is available at `http://code.google.com/p/stage/`.

1.4 Glossary

Test Team Leader The person in charge of all the testers.

Project Leader The person in charge of the whole project.

Client's Representatives Client's Representatives are people who overlook the Acceptance Test Plan execution on behalf of the customers.

Software Requirements Specification A Software Requirements Specification is a document which describes the behaviour of a system.

Functional Requirements Functional Requirements define the internal workings of the software.

Unit Tests A procedure in the software to validate that individual modules and other units of source code are working properly

Integration Test Integration Test is a test phase which is employed after the Unit Tests are validated and tests how all the different modules of a software system fit and work together with each other.

System Test System Test is conducted after completion of the Integration Test to evaluate the system's compliance with its specified requirements

2 Test Approach and Constraints

This section describes the overall approach, particular techniques and testing tools which will be used during the Acceptance Test Plan of the STAGE and any constraints that may apply.

2.1 Test Objectives

The Acceptance Test Plan process will prompt the client to evaluate STAGE and verify whether it performs in accordance with the client's requirements, listed in the Software Requirements Specification.

2.2 Test Structure

The Acceptance Test Plan will consist of a subset of test cases and methods, previously utilized in the Unit Tests, Integration Test and System Test conducted on the STAGE. The test cases will be carefully selected and agreed upon by both the developer and the client, and will allow for the most adequate verification of the functional requirements of the STAGE, as listed in the Software Requirements Document, without the extensiveness of the full-scale System Test.

It is essential that all appropriate Unit Tests, Integration Test and the System Test were successfully performed for STAGE prior to the Acceptance Test Plan and their results were reported and presented to the client.

3 Test Assumptions and Exclusions

3.1 Introduction

This section provides greater details about what issues and features of STAGE will be covered by Acceptance Test Plan process, and what issues and features of STAGE will not be covered.

3.2 Assumptions

It is assumed that all issues covered by the Acceptance Test Plan were also previously addressed by the Unit Tests, Integration Test and System Test of STAGE. The Acceptance Test Plan will cover:

- The functional requirements of the system listed in the Software Requirements Specification
- Usability of the system

3.3 Exclusions

It is assumed that all issues not covered by the Acceptance Test Plan were previously addressed by Unit Tests, Integration Tests and System Tests of STAGE. The Acceptance Test Plan will not cover:

- The non-functional requirements of the system (except the aforementioned Usability) listed in the Software Requirements Specification
- Structural integrity of the source code

4 Entry and Exit Criteria

4.1 Introduction

This section lists the criteria which must be satisfied in order for the Acceptance Test Plan to begin, as well as the criteria which must be satisfied in order for the Acceptance Test Plan to stop.

4.2 Entry Criteria

The Acceptance Test Plan can be initiated after the following preconditions are met:

- Successful completion of Unit Tests, the Integration Test and a System Test.
- Setup and inspection of the testing environment to satisfies the System Requirements of the Software Requirements Specification.
- Reciept of a copy of the latest version of the Software Requirements Specification.

- Reciept of a copy of the latest version of the user-related documentation.
- Appropriate resourcing of the latest released version of STAGE.
- Consent of the Project Leader.
- Consent of the Client.
- Consent of the Test Team Leader.

4.3 Exit Criteria

The Acceptance Test Plan should be halted after either of the following:

- All requirements were tested without any deviation from expected behavior. (Success)
- At least one requirement deviated from the documented specification. (Failure)
- By mutual agreement between Client's Representative and the Tester, in which both parties' supervisors should be notified and the Acceptance Test Plan should be rescheduled for a later date. (Failure)

5 Testing Participants

5.1 Introduction

This section describes the roles and responsibilities of the staff members involved in the Acceptance Test Plan, as well as the procedure of reporting the test results and any problems that came up during testing.

5.2 Roles and Responsibilities

For the Acceptance Test Plan, the following roles were assumed by the following people:

- Test Team Leader: STAGE Team Leader
- Client's Representative: A person in charge from the client's side who will overview the testing process.
- Tester: A person who will execute the use case tests.

5.3 Training Requirements

All parties involved in the Acceptance Test Plan should be familiar with the user interface of STAGE, as well as with the system documentation and the Software Requirements Specification.

5.4 Problem Reporting

Any problem pointed out by either the Client's Representative or the Tester must be documented and reported to the Test Team Leader. Later the problem report will be submitted to the project Leader, and addressed during a periodic or urgent staff meeting depending on the severity of the problems.

5.5 Progress Reporting

The Acceptance Test Plan Report will be compiled once, after testing process is finished by the Test Team Leader and submitted to the Project Leader.

6 Test Cases

The test cases are distributed in sections covering functionality elements and use cases in the Software Requirements Specification. Each of the following test cases is in the format:

- **Pre-Conditions:** Conditions needed to initiate the test case
- **Actions:** The actions expected form a tester
- **Post-Conditions:** The expected outcome of the test case

6.1 Functional Requirements

6.1.1 Visualization

6.1.1.1 Terrain

- **Pre-Condition:** The visualization engine is running.
- **Action:** The user navigates the visualizer to any specific real-world location.
- **Post-Condition:** The proper terrain for the location is shown.

6.1.1.2 World Objects

- **Pre-Condition:** The visualization engine is running, and the user creates a scenario containing mobile and static world objects.
- **Action:** The user loads and starts the created scenario.
- **Post-Condition:** The visualization engine shows mobile and static world objects.

6.1.1.3 Mobile Agents

- **Pre-Condition:** The visualization engine is running, and the user creates a scenario containing mobile nodes.
- **Action:** The user loads and starts the created scenario.
- **Post-Condition:** The visualization engine shows mobile nodes.

6.1.1.4 Path-drawing

- **Pre-Condition:** The visualization engine is running, and the simulation contains an object.
- **Action:** The simulator moves an object to a new location.
- **Post-Condition:** The visualization engine draws the path for the object to move.

6.1.1.5 Maximum Models

- **Pre-Condition:** The visualization engine is running, and the user creates a scenario containing 300 models.
- **Action:** The user loads and starts the created scenario.
- **Post-Condition:** The visualization engine shows 300 models.

6.1.1.6 Modularity

- **Pre-Condition:** The visualization engine is running, and the user is running a simulation.
- **Action:** The user terminates the execution of the visualization engine.
- **Post-Condition:** The simulation continues to run.

6.1.1.7 Timing

- **Pre-Condition:** The visualization engine is running, and the user creates a scenario containing timed events.
- **Action:** The user loads and starts the created scenario.
- **Post-Condition:** Movement and communication occur at real-time speeds in the visualizer.

6.1.1.8 Viewpoints

- **Pre-Condition:** The visualization engine is running, and the user is running a simulation.
- **Action:** The user executes a second visualization engine.
- **Post-Condition:** The new visualization engine shows a different viewpoint than the original.

6.1.1.9 Viewpoints

- **Pre-Condition:** The visualization engine is running.
- **Action:** The user changes the viewpoint.
- **Post-Condition:** The visualization engine shows a new viewpoint.

6.1.2 Simulation Engine

6.1.2.1 Timing

- **Pre-Condition:** The simulation engine is running, and the user creates a scenario containing timed events.
- **Action:** The user loads and starts the created scenario.
- **Post-Condition:** Movement and communication occur at real-time speeds as reflected by the logs.

6.1.2.2 Limitations of Movement

- **Pre-Condition:** The simulation engine is running, and the user creates a scenario mobile objects.
- **Action:** The scenario attempts to move the object into an invalid location.
- **Post-Condition:** The simulation engine moves the object as close to the invalid location as possible.

6.1.2.3 Path-finding

- **Pre-Condition:** The simulation engine is running, and the simulation contains an object.
- **Action:** The simulator moves an object to a new location.
- **Post-Condition:** The visualization engine finds the path for the object to move.

6.1.2.4 Collision Prevention

- **Pre-Condition:** The simulation engine is running, and the simulation contains mobile objects and static objects.
- **Action:** The scenario attempts to move one world object into another world object.
- **Post-Condition:** The simulation engine moves the object as close to the invalid location as possible.

6.1.2.5 Maximum Models

- **Pre-Condition:** The simulation engine is running, and the user creates a scenario containing 300 models.
- **Action:** The user loads and starts the created scenario.
- **Post-Condition:** The simulation engine shows 300 models.

6.1.2.6 User Intervention

- **Pre-Condition:** The simulation engine is running.
- **Action:** The user inputs an event into the simulation.
- **Post-Condition:** The event is reflected by the simulation engine.

6.1.3 Network Definition

6.1.3.1 Interfaces

- **Pre-Condition:** The user creates a scenario with a node which has an interface.
- **Action:** The user loads and starts the created scenario.
- **Post-Condition:** The simulator has a node with an interface.

6.1.3.2 Networks

- **Pre-Condition:** The user creates a scenario with a node which has an interface, connected to a network.
- **Action:** The user loads and starts the created scenario.
- **Post-Condition:** The simulator has a node with an interface, connected to a network.

6.1.3.3 Links

- **Pre-Condition:** The user creates a scenario with two nodes, each of which has an interface that are connected to one another.
- **Action:** The user loads and starts the created scenario.
- **Post-Condition:** The simulator has two nodes, each of which with an interface that are connected to one another.

6.1.4 Scenario Definition

6.1.4.1 Location

- **Pre-Condition:** The user creates a scenario with a node which has a location.
- **Action:** The user loads and starts the created scenario.
- **Post-Condition:** The simulator has a node at the specified location.

6.1.4.2 Mobility

- **Pre-Condition:** The user creates a scenario with a node which has multiple event-based locations.
- **Action:** The user loads and starts the created scenario.
- **Post-Condition:** The simulator has a node which moves to the specified locations as a response to their respective events.

6.1.4.3 Events

- **Pre-Condition:** The user creates a scenario with an agent which has an action based on the occurrence of an event.
- **Action:** The user loads and starts the created scenario.
- **Post-Condition:** The simulator has a node which appropriately responds to the specified event.

6.1.5 Agent Framework

6.1.5.1 Agent Location

- **Pre-Condition:** The user creates a scenario with a node which has an agent.
- **Action:** The user loads and starts the created scenario.
- **Post-Condition:** The simulator executes the specified agent on the specified node.

6.1.5.2 Startup

- **Pre-Condition:** The user creates a scenario with a node which has an agent and startup parameters.
- **Action:** The user loads and starts the created scenario.
- **Post-Condition:** The simulator executes the specified agent on the specified node and applies the startup parameters.

6.1.6 Distribution Framework

6.1.6.1 Physical Distribution

- **Pre-Condition:** The user networks multiple physical machines together and creates a scenario containing multiple nodes.
- **Action:** The user executes the simulation engine, and loads and starts the created scenario.
- **Post-Condition:** The simulator evenly divides the simulation of the virtual nodes across the physical machines.

6.1.6.2 Distributed Agent Initialization

- **Pre-Condition:** The user networks multiple physical machines together and creates a scenario containing multiple nodes with agents.
- **Action:** The user executes the simulation engine, and loads and starts the created scenario.
- **Post-Condition:** The simulator evenly divides the simulation of the virtual nodes and their agents across the physical machines.

6.1.6.3 Simulation Server

- **Pre-Condition:** The user networks multiple physical machines together and creates a scenario containing multiple nodes with agents.
- **Action:** The user executes the simulation engine, and loads and starts the created scenario.
- **Post-Condition:** The physical nodes communicate hierarchically with the simulation server.

6.1.7 Data Aggregation

6.1.7.1 Data Aggregation

- **Pre-Condition:** The user executes the simulation engine, and loads and starts the created scenario.
- **Action:** The user subscribes to the event channel API via an external application.
- **Post-Condition:** The user's application receives events from the event channel API in real-time.

6.1.8 Data Storage

6.1.8.1 Scenario Definitions

- **Pre-Condition:** The user creates a scenario.
- **Action:** The user loads and starts the created scenario.
- **Post-Condition:** The simulation engine stores and parses the loaded scenario.

7 Traceability Matrix

Column headings represent requirement identifiers. The beginning of each row is a test case identifier.

	3.1.1.1	3.1.1.2	3.1.1.3	3.1.1.4	3.1.1.5	3.1.1.6	3.1.1.7	3.1.1.8	3.1.1.9	3.1.2.1	3.1.2.2	3.1.2.3	3.1.2.4	3.1.2.5	3.1.2.6	3.1.3.1	3.1.3.2	3.1.3.3	3.1.4.1	3.1.4.2	3.1.4.3	3.1.5.1	3.1.5.2	3.1.6.1	3.1.6.2	3.1.7.1	3.1.8.2
6.1.1.1	x																										
6.1.1.2		x																									
6.1.1.3			x																								
6.1.1.4				x																							
6.1.1.5					x																						
6.1.1.6						x																					
6.1.1.7							x																				
6.1.1.8								x																			
6.1.1.9									x																		
6.1.2.1										x																	
6.1.2.2											x																
6.1.2.3												x															
6.1.2.4													x														
6.1.2.5														x													
6.1.2.6															x												
6.1.3.1																x											
6.1.3.2																	x										
6.1.3.3																		x									
6.1.4.1																			x								
6.1.4.2																				x							
6.1.4.3																					x						
6.1.5.1																						x					
6.1.5.2																							x				
6.1.6.1																								x			
6.1.6.3																									x		
6.1.7.1																										x	
6.1.8.1																											x