STAGE

Software Requirements Specification

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 6

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

List of Figures

1 Introduction

1.1 Purpose

This requirements document defines the functional and non-functional requirements for the STAGE project. These requirements include, but are not limited to, the performance, interfaces, and hardware goals and limitations. The information presented here is intended for the development team and the external stakeholders, currently Dr. William Regli, Mr. Joseph B. Kopena, Mr. Joeseph P. Macker, and The U.S. Naval Research Laboratory.

1.2 Scope

The goal of the STAGE project is to allow for the testing of agents working together on separate nodes across multiple scenario and topology situations. STAGE allows the user to examine how effectively nodes (with agents running on them) interact with one another. This includes network connections between nodes, the fidelity of these connections, and the proficiency of the agents to process and transmit data.

STAGE consists of six main components: Visualization, Topology, Scenario, Agent Framework, Distribution Framework, and the API. The Visualization allows the user to see what is happening in the simulation. For example, network links and their strength, movement of nodes in the virtual world, as well as the movement of non-agent world objects all have a visual representation over a map of the world. The Topology is a scripted language that defines interfaces and network connections on each node. It also describes any changes in the network that do not result from mobility. The Scenario is a scripted language that defines the location and mobility patterns of each node and non-node world object. It also handles the timing of events. An Agent Framework indicates the agent(software) running on each node and permits different algorithms to be tested on different nodes. The Distribution Framework helps to distribute the simulation across multiple physical nodes in a cluster. An API is also provided to allow the user to interact with the simulation. For example, it allows a user to trigger outside events or query information being generated by the simulation.

Users of STAGE are researchers looking to improve or test their current agent/network interactions across different topology and scenario combinations.

1.3 Definitions, Acronyms, and Abbreviations

Agent Agents are simulated pieces of software that run on nodes in the network. They consist of different algorithms that are relevant for the user to test on different scenarios and topologies.

Distribution Distribution refers to the process of distributing the simulation across a multi-platform physical cluster. This allows the system to exceed the number of nodes per platform for a single simulation at the system's discretion. A framework will be provided to allow the user to distribute their simulation.

Node Nodes are virtual or physical machines that consist of agents and network interfaces. If nodes are virtual, many nodes may run on one physical machine.

Scenario Scenario is comprised of a scripted language indicating the location simulated nodes within the virtual world. These nodes consist of agents (see definition of 'Agent') and non-agent world objects such as planes, boats, ground vehicles, etc.

Terrain Terrain refers to the simulated landscape. This includes such geography as the slope of the land, the tree density, water v.s. land surfaces, etc.

Topology Topology describes time-dependent connections between nodes and their characteristics (e.g. radio model). It is described with a scripting language which specifies the details of network interfaces on each simulated node, including radio models and throughput characteristics. It describes any physical or wireless links that connect these interfaces. Further, it indicates changes in linkage over time such as a wireless interface switches, wireless LANs, or a physical link being created or severed.

Visualizer The Visualizer allows the simulations to be superimposed over real-world topography. This permits the user to examine the behavior of the agents. It also allows for overlays such as link quality, traffic rates, and other metrics deemed important to specific components.

1.4 References

These documents have been used as reference materials for various technologies involved with this project.

- SPEYES: Sensing and Patrolling Enablers Yielding Effective SASO: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=1559616
- Service Sniffer Requirements Document: http://servicesniffer.net/documents/requirements.html
- Developing an Agent Systems Reference Architecture: www.cs.drexel.edu/~dn53/papers/paper_cameraready.pdf

1.5 Overview

The rest of this document is organized as follows:

- Section 2 gives a high level overview of the project requirements.
- Section 3 gives more detailed project requirements.
- Sections 3.3 and 3.4 provides detail in the inputs and outputs of the system,
- Section 3.1 specifies what the software will do,

The rest of Section 3 explains other minor requirements.

2 Overall Description

2.1 Product Perspective

STAGE is used as a testbed for sensors and respective algorithms in the laboratory against virtual operations using data from simulated sensors and nodes. The ability to independently and quickly vary the network topology, application suites, environment, or any other variable is essential to collect data which would be cost-prohibitive to produce in a real-world scenario. Furthermore, the visualization tool is used for conceptual demonstrations.

2.1.1 System Interfaces

STAGE combines:

- Scenario Language - Defining the entire scenario in a very simple language
- Agent Framework - Supporting the customization of semi-intelligent agents
- Topology Creator - Creating a virtual topology of nodes defined in the Scenario Language
- Distribution Framework - Distributing a large-scale simulation across numerous physical machines
- Data-Collection API - Collecting all relevant data through a simple API
- Visualization Engine - Conceptually visualizing terrain, node movement, link status, etc.

2.1.2 User Interfaces

STAGE's user interface is two-fold, with each interface for different tasks and uses. The basic interface allows a user to simply and quickly define a scenario with the Scenario Language and begin an experiment, using the API to collect data in a way that is defined by the user. The Visualization Engine allows the user to interpret the experiment visually, without a need for data-collection or the API.

2.2 Product Functions

This software is a testbed, where the users specify an environment, network configuration, and agent software running in the simulation. Each of these items may be varied independently at runtime.

The primary function of this software is to coordinate an event-based simulation comprised of virtual nodes, other world objects, agent software, and network devices. Specifically the software starts simulations, handles all events, and determines when a scenario is complete. The simulation may run in either simulated time or real-time depending on the user's configuration.

STAGE also provides an optional visualization component, useful for monitoring the current simulation state. The visualization uses data provided by the simulation to display world objects including nodes and terrain, as well as annotations such as network-link changes and node paths.

Simulation of inter-node communications is handled by STAGE's networking component. This component provides realistic simulation of network links and virtual node networking stacks.

2.3 User Characteristics

The system has four different types of users. Each type of user has different goals and needs to use different parts of the system. These user roles, while they may be performed by the same user, are here split into the four distinct roles for clarity:

2.3.1 Fictional User: Alice, Project Manager

Alice uses STAGE for demonstrations and to acquire funding for research. Alice wants to complete the tasks below with the visualization:

- Start a simulation
- View the simulation in real or simulated time
- Move the viewpoint
- Visualize simulation events including link changes and entity movements
- View output from agents if implemented

2.3.2 Fictional User: Bob, Agent Algorithm Researcher

Bob researches agent algorithms and uses STAGE to compare their effectiveness. Bob wants to complete the tasks below:

- Use the API to implement agents which run on network nodes
- Use the visualizer to assure scenarios are properly setup
- Disable the visualizer and run scenarios many times
- Collect aggregate data from scenario trials via subscription to the API's event channel

2.3.3 Fictional User: Carol, Network Protocol Developer

Carol uses STAGE to test networking protocols. Specifically, she tries different network protocols with a single agent setup to find the best combination. Carol wants to complete the tasks below:

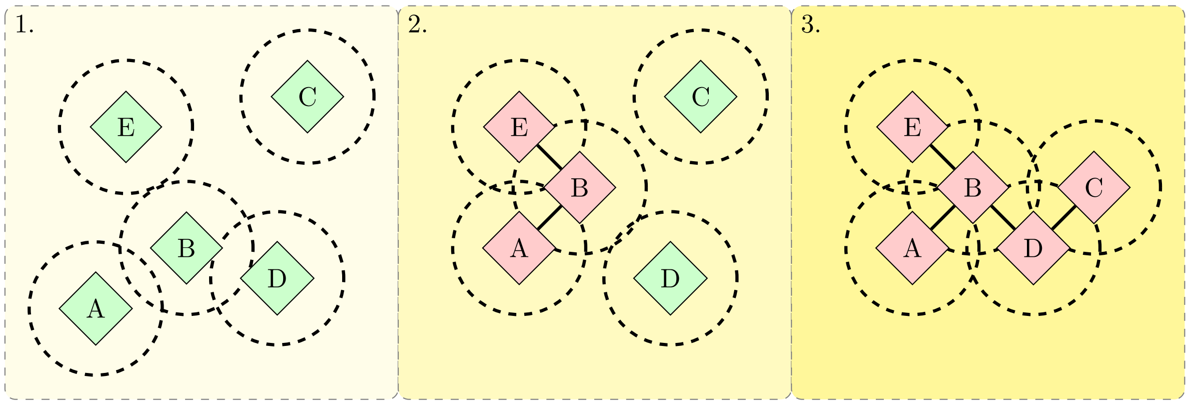


Figure 1: A prototype of what “Bob” may create: *1.* Agents have been configured to perform random walks within a boundary, and each contains a wireless interface. *2.* When nodes recieve an event indicating they are in range of another node, they hold their position and create a network link. *3.* A static network is formed.

- Use the networking component to implement networking protocols
- Disable the visualizer and run scenarios many times
- Collect aggregate data from scenario trials via subscription to the API’s event channel

2.3.4 Fictional User: Dave, Large Scale Simulator

Dave uses STAGE to run large scale simulations which run slower than real-time on his personal computer. Dave wants to complete the tasks below:

- Distribute agent instances to multiple physical machines
- Abstract the physical links from the simulation
- Coordinate data collection after simulations

2.4 Constraints

The system is able to interpret and/or process any agent or topology it is given. If this isn’t possible for any reason, the system informs the user of this. No piece of this system requires elevated privileges.

2.5 Assumptions and Dependencies

It is assumed that a user, administrator, or developer has the ability to install required libraries which will named in the future as a result of the development process.

3 Specific Requirements

3.1 Functional Requirements

3.1.1 Visualization

The following applies to STAGE’s Visualization Engine (VE):

3.1.1.1 Terrain The Visualization Engine displays varying types of terrain, as defined by the environment.

3.1.1.2 World Objects The Visualization Engine displays mobile and static world objects, as defined by the scenario.

3.1.1.3 Mobile Agents The Visualization Engine displays mobile nodes.

3.1.1.4 Path-finding The Visualization Engine finds the path for an object to move to a given coordinate.

3.1.1.5 Maximum Models The Visualization Engine is able to display 300 models in a given scenario.

3.1.1.6 Modularity The Visualization Engine component is modular, i.e. it can be removed from the framework without affecting functionality (with the exception of the visuals).

3.1.1.7 Timing The Visualization Engine runs in real-time.

3.1.1.8 Viewpoints The Visualization Engine is able to display the world from more than one point of view.

3.1.1.9 Viewpoints The Visualization Engine is able to display the world from any given point of view.

3.1.2 Simulation Engine

The following applies to STAGE's Simulation Engine (SE):

3.1.2.1 Timing The Simulation Engine runs in real-time.

3.1.2.2 Limitations of Movement The Simulation Engine limits the movement of models to specific areas or types of terrain (e.g. boats may only move on river).

3.1.2.3 Path-drawing The Simulation Engine finds the path for an object to move to a given coordinate.

3.1.2.4 Collision Prevention When finding a path, the Simulation Engine ensures the path will not steer an agent to navigate into world objects.

3.1.2.5 Maximum Models The Simulation Engine keeps in memory up to 300 nodes.

3.1.2.6 User Intervention The Simulation Engine allows for the interruption or change of events by human input.

3.1.3 Network Definition

3.1.3.1 Interfaces The definition language specifies interfaces on each virtual node. Specifically, each interface is described with a physical model (radio, wired, etc.) and an appropriate link-layer protocol.

3.1.3.2 Networks The definition language specifies networks to which interfaces may interact. Networks are described with a unique identifier and physical layer model (e.g. radio propagation model).

3.1.3.3 Links The definition language specifies one-to-one links between interfaces. A link is described by a physical layer model.

3.1.4 Scenario Definition

3.1.4.1 Location The scenario definition language defines the location of agent and non-agent world objects.

3.1.4.2 Mobility The scenario definition language describes the mobility or each virtual node as a function of time.

3.1.4.3 Events Mobility is alterable as a result of events, specified in the scenario definition.

3.1.4.4 Non-node Behavior The scenario definition language describes the movement of and events triggered by entities other than virtual nodes.

3.1.5 Agent Framework

3.1.5.1 Agent Location The agent framework specifies on which virtual node(s) a given agent should operate.

3.1.5.2 Startup The agent framework specifies the startup parameters of each agent.

3.1.5.3 Collection The agent framework specifies the output parameters of each agent instance.

3.1.6 Distribution Framework

The Distribution Framework uses the API to spread the simulation across a Physical Distribution, and to aggregate trial data. It consists of a network of one-to-many physical machines.

3.1.6.1 Physical Distribution The distribution framework distributes virtual nodes evenly across physical nodes. Further, it handles the establishment and initialization of agents on their respective physical node.

3.1.6.2 Simulation Server The distribution framework organizes physical nodes hierarchically to interact with the simulation server.

3.1.7 Data Aggregation

3.1.7.1 Real-Time Event Subscription The simulator has an open API channel which should be used to monitor real-time events published by the simulation.

3.1.8 Data Storage

3.1.8.1 Outside Databases No outside database system is used for data storage, unless it is introduced by the user via the API

3.1.8.2 Scenario Definitions Scenario definitions, including the topology and agents, are stored and parsed by the system

3.2 Non-Functional Requirements

3.2.1 Hardware Interfaces

STAGE requires that the machine running the system has one to many network interfaces to support distribution.

3.2.2 Software Interfaces

STAGE requires the use of OpenGL in order to use the visualizer.

3.2.3 Memory Constraints

STAGE requires a machine with at least 4GB of RAM.

3.2.4 Site Adaptation Requirements

STAGE requires that the system is configured per-site to create a distributed simulation, or to interact on-the-fly.

3.2.5 Operating System

STAGE is able to run on Unix, Linux, and Mac

3.2.6 Processor

STAGE requires at least a 32- or 64-bit x86 3.0GHz processor

3.2.7 Network Connection

STAGE requires any recent network card which is currently connected to the simulation network (if the simulation is distributed)

3.2.8 Visualizer

STAGE requires a graphics card which will support OpenGL or an equivalent

3.3 Input Formats

3.3.1 Scenario

3.3.1.1 Location Data The simulation is able to read location data for nodes.

3.3.1.2 Mobility Data The simulation is able to read mobility data for nodes.

3.3.1.3 Scenario Input The simulation is able to respond to event input from scenario input files

3.3.2 Topology

3.3.2.1 Network Interfaces The simulation is able to read network interfaces for nodes

3.3.2.2 Network Links The simulation is able to read links in the network between interfaces

3.3.2.3 Network Topology The simulation is able to read changes to the network topology

3.4 Output Formats

3.4.1 Scenario

3.4.1.1 Network Changes The system supports outputting network data changes over time post-simulation

3.4.1.2 Network Fidelity The system supports outputting network fidelity over time post-simulation

3.4.2 Topology

3.4.2.1 Topology Changes The system supports outputting topology changes over time post-simulation

3.4.2.2 Mobility Patterns The system supports outputting agent and non-agent mobility patterns post simulation

3.5 User Interface

The following applies to STAGE's basic interface:

3.5.1 Scenarios

3.5.1.1 Scenario The interface allows a user to define a scenario.

3.5.1.2 Scenario Language The scenario defined by the interface is written in the Scenario Language.

3.5.2 Experimentation

3.5.2.1 Beginning Experiment The interface allows a user to begin an experiment.

3.5.2.2 Ending Experiment The interface allows a user to end an experiment.

3.6 Extensibility

3.6.1 Customization

3.6.1.1 Custom Topologies The system provides a means for users to create custom topologies.

3.6.1.2 Custom Agents The system provides a means for users to create custom agents.

3.7 Testing

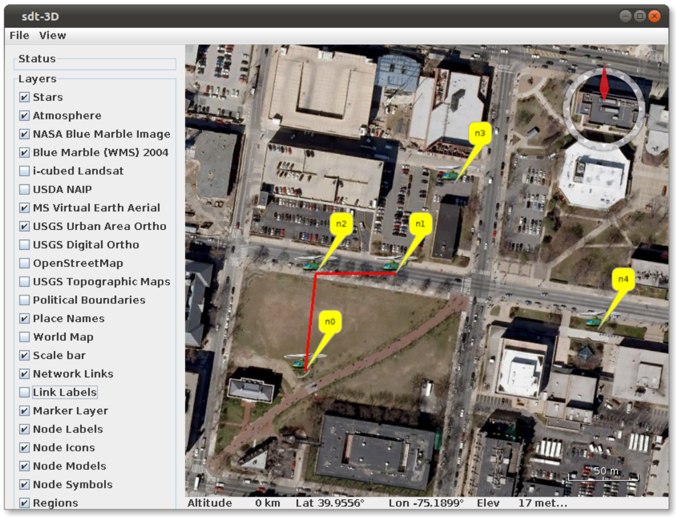
All code with in the system must be tested. Core functionality of the software has 100% code coverage with automated unit and integration tests. The visualizer testing is done manually as it is a smaller component and would otherwise require a disproportionately large investment in initial setup. Unit tests cover the algorithms used in our system while integration tests are used to test our data-flow.

4 System Evolution

With more time, the STAGE team could create additional tools to aid in the the development of more complex agents or topologies, so that user customization does not become too granular. Furthermore, tools which can automate or assist in data collection and parsing could be implements. Even further development could lead to a platform that included Windows or other operating systems.

4.1 Documentation

1. The software provides a User Manual containing step-by-step instructions for user perspective in Section 2.3.
2. The software provides Developer Documentation containing:
    (a) Scenario Language syntax and usage
    (b) Network Topology definitions and example networks
    (c) Agent Framework definitions and example agents



Layers
- ✓ Stars
- ✓ Atmosphere
- ✓ NASA Blue Marble Image
- ✓ Blue Marble (WMS) 2004
- ☐ i-cubed Landsat
- ☐ USDA NAIP
- ✓ MS Virtual Earth Aerial
- ✓ USGS Urban Area Ortho
- ☐ USGS Digital Ortho
- ☐ OpenStreetMap
- ☐ USGS Topographic Maps
- ☐ Political Boundaries
- ✓ Place Names
- ☐ World Map
- ✓ Scale bar
- ✓ Network Links
- ☐ Link Labels
- ✓ Marker Layer
- ✓ Node Labels
- ✓ Node Icons
- ✓ Node Models
- ✓ Node Symbols
- ✓ Regions

Altitude 0 km Lat 39.9556° Lon -75.1899° Elev 17 met...

Figure 2: The STAGE Display Tool (SDT) shows mobile nodes (represented by the helicopters) interacting on the world map. The terrain (the city below) can also be seen. The red lines between nodes represent a network connection. Connected nodes become stationary as unconnected nodes continuosly change their position in the simulation until they come within range of another node.