Defence Research and Development Canada
Recherche et développement pour la défense Canada

DEFENCE R&D DÉFENSE

Simulation network exploitation

SEDReC station: specifications document

Bobby Chawla and Oliver Schoenborn

Defence R&D Canada – Ottawa
Canada
CONTRACT REPORT
DRDC Ottawa CR 2008-314
April 2009

Simulation network exploitation

SEDReC station: specifications document

Bobby Chawla
Oliver Schoenborn
CAE Professional Services

Prepared By:
CAE Professonal Services
1135 Innovation Dr., Suite 300
Ottawa, ON K2K 3G7
Contract Project Manager: Leo Roberts, 613-293-8993
W8475-06BM04
CSA: Nacer Abdellaoui, Defence Scientist, 613-998-4582

Defence R&D Canada – Ottawa

Contract Report
DRDC Ottawa CR 2008-314
April 2009

Contract Scientific Authority

Original signed by Nacer Abdellaoui

Nacer Abdellaoui

Defence Scientist

Approved by

Original signed by Julie Tremblay-Luther

Julie Tremblay-Luther

H/CARDS

Approved for release by

Original signed by Pierre Lavoie

Pierre Lavoie

DRP Chair

Abstract

The Synthetic Environment Distributed Resources (SEDR) Tool Suite is a set of three applications to facilitate knowledge management, accessibility, and repeatability of simulation exercises. The SEDReC Station is the application that automates the start-up, execution and shutdown of a complex, distributed set of simulation resources that form a simulation system. This manual starts with overview and background information on the application. In the remaining chapters the Functional and Non-Functional requirements are given.

Résumé

Le kit d’outils des ressources distribuées des environnements synthétiques (SEDR) est un ensemble de trois applications misant a faciliter la gestion des connaissances, l'accessibilité, et la répétabilité des exercices de simulation. La Station SEDReC est l'application qui automatise le démarrage, l'exécution et la fermeture d'un système complexe et distribué de ressources de simulation. Ce manuel offre une vue d'ensemble de l’application et des informations de base, suivi par les exigences fonctionnels et non fonctionnels du système.

This page intentionally left blank.

Executive summary

Simulation network exploitation: SEDReC station: specifications document

Bobby Chawla, Oliver Schoenborn; DRDC Ottawa CR 2008-314; Defence R&D Canada – Ottawa; April 2009.

Introduction or background: The Synthetic Environment Distributed Resources (SEDR) Tool Suite is a set of three applications to facilitate knowledge management, accessibility, and repeatability of simulation exercises. The SEDReC Station is the Control Station in the SEDR Tool Suite.

Results: This application provides a mechanism for automating the start-up, execution and shutdown of a complex, distributed set of simulation resources that form a simulation system. This allows end-users of the simulation system, as well as non-expert users, to repeat the system execution, find what computers no longer support the system components, and to understand what steps are necessary to execute it. Moreover, the steps can be grouped into a hierarchy to help the user see how different steps, or actions, relate to one another.

This manual starts with overview and background information on the application. In the remaining chapters the Functional and Non-Functional requirements are given. This manual assumes some familiarity with simulation systems, their development, and system requirements in general.

Sommaire

Simulation network exploitation: SEDReC station: specifications document

Bobby Chawla, Oliver Schoenborn; DRDC Ottawa CR 2008-314; R & D pour la défense Canada – Ottawa; Avril 2009.

Introduction ou contexte: Le kit d'outils des ressources distribuées des environnements synthétiques (SEDR) est un ensemble de trois applications développées pour faciliter la gestion des connaissances, l'accessibilité, et la répétabilité des exercices de simulation. Le Station SEDReC est la composante de contrôle du kit SEDR. Cette application fournit un mécanisme permettant d'automatiser le démarrage, l'exécution et la fermeture d'un système de simulation complexe composé par un ensemble de ressources de simulations distribuées de simulation.

Resultats: Cela permet aux utilisateurs finaux, ainsi qu'aux utilisateurs non-experts, de facilement ré-exécuter le système de simulation, d'identifier les composantes qui ne sont plus disponibles au sein du système, et de comprendre les étapes et mesures nécessaires pour correctement exécuter ce system de simulations. En outre, ces mesures sont regroupées hiérarchiquement pour aider l'utilisateur à voir clairement les séquences et les interrelations des différentes mesures, étapes, ou actions, sont inter-liées.

Ce manuel commence avec une vue d'ensemble de l'application ainsi que les informations pré-requises. Dans le reste des chapitres les exigences fonctionnels et non fonctionnels nécessaires sont étalées. Ce manuel assume que le lecteur possède une bonne familiarité avec les systèmes de simulation et leur développement, ainsi que les exigences du système en général.

Table of contents

This page intentionally left blank.

1 Introduction

The Synthetic Environment Distributed Resources (SEDR) Tool Suite is a set of three applications to facilitate knowledge management, accessibility, and repeatability of simulation exercises. The SEDReC Station is the Control Station in the SEDR Tool Suite. This application provides a mechanism for automating the start-up, execution and shutdown of a complex, distributed set of simulation resources that form a simulation system. This allows end-users of the simulation system, as well as non-expert users, to repeat the system execution, find what computers no longer support the system components, and to understand what steps are necessary to execute it. Moreover, the steps can be grouped into a hierarchy to help the user see how different steps, or actions, relate to one another.

This manual starts with an overview and background information on the application. In the remaining chapters the Functional and Non-Functional requirements are given. This manual assumes some familiarity with simulation systems, their development, and system requirements in general.

1.1 Problem Statement

Simulation systems typically consist of many programs, running on a heterogeneous network and communicating using a variety of methods, e.g. TCP/IP, shared memory, pipes etc. They must be started in the correct sequence, with the right amount of delays between them, etc. Similarly each system must be shutdown according to specific rules. The steps are difficult to express clearly in documentation, and do not use a language that is understood by computers. This requires that the reader not only interpret the instructions, but execute what they understand, as well as infer steps that were missed in the documentation, understand error messages, determine if some software has been upgraded and is no longer compatible with the simulation system to be run, etc.

The system must therefore enable the “system expert” to encode the system setup, start-up, execution and shutdown of a large variety of systems distributed over many computers, while enabling the non-expert to read the steps, have the system execute them, and receive feedback on failures.

1.2 Types of Users

System Experts: the people involved in the development of a simulation system.

System Users: the people wanting to use the system for data generation and analysis. They can be the same as the experts, or may be the operators needed to steer the simulation once executing, or may be the end-users interested in the analysis results. Often, a combination thereof.

Network Administrators: the people in charge of maintaining the infrastructure needed to support the system’s function, e.g. the installation of client/server software, etc.

1.3 Operational Setting

Currently expert users develop systems, write a technical report or article, and the project is closed. Documentation about which machines were used, the specifics of the software used (version etc), as well as the sequence of steps to start the system, is often lacking.

The new system will provide a client program and a server package. The server package can be installed easily on any host that supports Python, whereas the client can be installed on any host to be used by the expert users. The servers should *not* be reachable from the Internet due to security risk (even if OpenSSL is used). The new system interfaces with existing systems only in terms of setup/startup/shutdown of programs and transfer of files across systems.

The system components are therefore:

1. **Servers**:
    a. Hosts on which one or more simulation components will be run
    b. Hosts able to execute commands received from a remote (client) machine
2. **Network**: A high-speed TCP/IP network
3. **Client**:
    a. A client machine running the control station software
    b. The machine can send commands to remote servers for controlling the system setup/startup/shutdown

1.4 Impact Analysis

For the client machines, the impact is minimal: they need only display an execution “recipe” in a GUI, send commands to remote hosts, monitor the success/failure of those commands, and provide feedback to the user via the GUI. This imposes a typical desktop program load on the machine. On the positive side, the client user does not need to be physically located at all the server machine in order to run programs on the servers.

For the network, the impact is also minimal: some packaged data, representing commands and status information, is moving back and forth between the client and servers. The amount of data is minimal (e.g. 1-2 kB) over the span of minutes.

For the servers, the impact can be considerable: the servers are often used for many different simulation programs, which implies:

- One or more other users may be using the system at the time the execution commands are issued. Their programs could be impacted by the increased load due to the extra simulation components running on their system. Their desktop could be impacted if the simulation programs open GUI’s on the desktop. Those users could inadvertently install new,

incompatible or less stable versions of simulation components or libraries used by those components, leading those simulation components to crash, behave unpredictably or incorrectly, or fail start-up.

- It is important that some configuration management take place in parallel to the creation of "system control recipes".

1.5 Related Systems

Many software tools exist that provide one or more aspects of the described functionality but they are either deficient one some aspects, are not customizable, or are limited to HLA. Two examples are *MARCI* which is part of JSAF by BMH Associates Inc. [1], and *HLA Commander* by Pitch Technologies [2].

1.6 Limitations

The simulation system control station is architecture-agnostic, meaning that it does **not** replace architecture-specific software tools designed to setup/start/stop simulation systems within a particular architecture, such as the High Level Architecture (HLA). The control station can however start/stop such components for the purpose of delegating some of the startup/stop sequence to such tool.

The control station will not know until it executes the control "actions" whether an action is still supported. E.g. an action may indicate that some executable should run on host A. There is no way of knowing until the "recipe"[*] of actions is executed whether host A still has the executable, whether the executable can run with the given parameters, and whether the files, data, libraries, socket ports, etc that it depends on are available.

Basic assumption: system startup can be decomposed into groups of actions, where each action in the group is an individual action, or a sequence of actions, or a group of actions in parallel.

[*] A structured description of the precise sequence of commands necessary to launch a simulation system distributed over several networked computers.

2 Functional Requirements

2.1 User Interface Overview

The User Interface will show the following views:

- An action tree view of the hierarchy of control actions of the “recipe”.
- A hosts view of all the computers mentioned in the recipe, showing which actions from the tree take place on which host, whether a host is available (as of last hosts discovery), and what is the status of an action being executed (see below).
- An action details view, showing the details of the action selected from the tree. Action details are e.g. which host the action runs on, what type of action (run, upload file, etc), which files to upload, etc.
- An editable action details view, where the attributes can be changed.
- A failures view, showing the reason that an action failed.
- Some command buttons related to the recipe: execute, stop, refresh list of hosts.

Selecting an action in one view should select the action in the other views. E.g. selecting action A in the tree should highlight action A in the hosts view, show the details in the action details view, and highlight the failure message (if any) related to the action’s last execution.

2.2 System-Specific Requirements

The user starts the control station from the command line, specifying a “control recipe” to read at start-up. This file is also called an “action recipe” file. This is an XML file whose elements contain a mixture of text, attributes and action elements satisfying a schema described via Relax NG. The text forms part of the documentation (together with elements and attributes), while the elements and attributes per se form the executable portion of the document. This XML file is edited by user with tool of their choice (e.g., xmloperator).

2.2.1 Action tree view

1. The user can see a hierarchical representation of the action tree described in the control recipe.

2. The user can select any node of the tree to get further details on the action, shown in the Action Details view. The selected node is highlighted.

3. The user can expand and collapse tree nodes and scroll the panel vertically if the tree goes beyond the bottom of the tree panel.

2.2.2 Hosts view

1. The user can see which hosts are mentioned in the recipe.

2. The user can see which actions are to take place on which hosts via host action buttons. This does not apply to action groups since groups involve actions over (typically) more than one host.

3. The hosts are displayed on an unordered grid as "square" as possible. E.g. if only one, two or three hosts are used, then the grid should be 1x3 (one row by 3 columns) or 2x2 (prototyping required to determine which is best). If four hosts are used, the grid should be 2x2.

4. The user can see which of the displayed hosts responded to the last discovery request. A discovery request is sent out over the network when the application has read the control recipe.

5. The user can refresh the availability of hosts at any time, except while a recipe is executing.

6. The user can click on a host action and see the corresponding action in the Action tree view highlighted.

7. Once the recipe is being executed, the host action buttons change color to identify the state of the action execution as described in Annex A.

2.2.3 Action details view

1. The user can easily identify what is the type of action, where it will take place, what is its position in the action tree, and any other details related to the specific action.

2.2.4 Editable action details view

1. The user can change the attributes of the action and add texture documentation for any of the attributes or the action itself.

2.2.5 Failures view

1. The user can see each failure as soon as it is discovered by the client application. Therefore failure from a child action further from the action tree root will appear first in the list of failures. The root action failure will always appear last.

2. The user can select a failure and see where in the action tree this action is located.

3. The user can see any failure message that accompanies the failure.

2.2.6 Action execution

1. The user can trigger the execution of the loaded control recipe by clicking on a button directly accessible on GUI.

2. Once the execution has started, the user can abort the execution by clicking on a button directly accessible on GUI, causing each action to already started to be aborted.

3. The user cannot trigger the execution if already executing, and cannot stop execution if not started.

2.2.7 Control Recipe

1. The run and start actions must allow for the remote execution to take place in a console window if the user so desires, rather than in the default simulation host *server* console window, in case several simulation components are run on the same host.

2. The user can define individual actions:

    a. Run: execute a process on a remote host and wait for successful exit. The mission is successful if remote process exits without error code, and fails if process exits with error or crashes. Action is done when remote process exits.

    b. Start: execute a process on a remote host. Mission is successful if process can be started, fails otherwise. Action is done when remote process exits.

    c. Upload: transfer a file from local host (client) to a remote host.

    d. Download: transfer a file from a remote host to the local host (client).

3. The user can define a group of actions to be executed in sequence. The allowed actions within that group are: other action groups, and individual actions. An action within the group is started only when its predecessor has reached its goal.

4. The user can define a group of actions to be executed in parallel. The allowed actions within that group are: other action groups, and individual actions. All actions within the group are started simultaneously, in unspecified order.

5. The mission of an action group (sequence or parallel) is to succeed the mission of all actions within the group, hence the action group's mission fails if any action within the group fails its mission. The group action is done when all actions within the group are done.

3 Non-Functional Requirements

3.1 System-Related Non-Functional Requirements

3.1.1 Operational environment

- **Hardware platforms**: any modern PC (x86, etc)
- **Software platform**: Operating Systems supporting Python, wxPython, lxml, Rpyc, OpenSSL and the Python subprocess module.
- **External software interoperability**: System is standalone but should have a standard Python subprocess module so it can control other processes.

3.1.2 Standards conformance

- **XML**: w3c at www.w3c.org.
- **Relax NG**: OASIS at www.oasis-open.org/committees/relax-ng.

3.1.3 General characteristics

- **Reliability**: client should run reliably on Windows and Linux machines and interact reliably with servers on intranet
- **Performance**: should be able to support 10 – 50 actions without noticeable slowness
- **Correctness**: action attributes can only be checked for correctness via execution
- **Security**: the client and servers should support TLS-based client authentication to gain access to servers, on a per-server basis
- **Privacy**: as supported by TLS-based encryption
- **Safety**: remote execution of programs presents the risk for serious injury if proper procedures are not in place. E.g. a user could cause motion of a robot arm controlled by an application available on host. However this is not a concern at FFSE.
- **Portability**: the client and server software should be portable to mainstream operating systems such as Windows and Linux.
- **Modifiability and extensibility**: the client software should use sound architectural principles such as object-oriented design, layering and decoupling.

3.2 Process-Related Non-Functional Requirements

Installation will require that the user follow instructions and run a sequence of executables on both the client and server hosts (one or more of each). Once all executables have been run and data copied as per instructions, the system will be ready for use.

3.3 Personnel-Related Non-Functional Requirements

Expert users should be familiar with the types of actions encountered in the context of setup, startup, and shutdown of distributed simulation systems.

End users should be familiar with how to start a program, use mouse and keyboard, and click on GUI items.

References

[1] About Joint Semi Automated Forces (JSAF) (online), www.bmh.com (Access date: 10 Mar. 2008)

[2] Pitch Commander (online), http://www.pitch.se/products/commander/monitor-and-control.html (Access date: 12 Nov. 2008)

This page intentionally left blank.

Annex A Action States

Actions defined within a control recipe have the following states and transitions:

- ***Started***: the action has been started, but its mission status is not yet known. This state is shown as blinking yellow. When the mission status becomes known, a transition takes place to *Mission Failed* or *Mission Succeeded*.
- ***Mission Failed***: the action's mission has failed, but the action is still executing. This state is shown as blinking red. Once the action is done, a transition takes place to *Ended Unsuccessfully*.
- ***Mission Succeeded***: the action's mission has succeeded, but the action is still executing. This state is shown as blinking green. Once the action is done, a transition takes place to *Ended Successfully*.
- ***Ended Successfully***: the action has ended (it is done), but the mission failed. This state is shown as steady red. This is a final state.
- ***Ended Unsuccessfully***: the action has ended (it is done), and the mission succeeded. This state is shown as steady green. This is a final state.

This page intentionally left blank.

List of symbols/abbreviations/acronyms/initialisms

DND	Department of National Defence
DRDC	Defence Research & Development Canada
DRDKIM	Director Research and Development Knowledge and Information Management
R&D	Research & Development
SE	Synthetic Environments
SEDR	Synthetic Environment Distributed Resources
SEDRA	SEDR Access
SEDReC	SEDR Control
SEDReDE	SEDR Data Entry
SNE	Simulation Network Exploitation

This page is intentionally left blank

DOCUMENT CONTROL DATA

(Security classification of title, body of abstract and indexing annotation must be entered when the overall document is classified)

1. ORIGINATOR (The name and address of the organization preparing the document. Organizations for whom the document was prepared, e.g. Centre sponsoring a contractor's report, or tasking agency, are entered in section 8.)

CAE Professional Services
1135 Innovation Dr., Suite 300
Ottawa, ON K2K 3G7

2. SECURITY CLASSIFICATION (Overall security classification of the document including special warning terms if applicable.)

UNCLASSIFIED

3. TITLE (The complete document title as indicated on the title page. Its classification should be indicated by the appropriate abbreviation (S, C or U) in parentheses after the title.)

Simulation network exploitation: SEDReC station: specifications document

4. AUTHORS (last name, followed by initials – ranks, titles, etc. not to be used)

Chawla, B., Schoenborn, O.

5. DATE OF PUBLICATION (Month and year of publication of document.)

April 2009

6a. NO. OF PAGES (Total containing information, including Annexes, Appendices, etc.)

24

6b. NO. OF REFS (Total cited in document.)

0

7. DESCRIPTIVE NOTES (The category of the document, e.g. technical report, technical note or memorandum. If appropriate, enter the type of report, e.g. interim, progress, summary, annual or final. Give the inclusive dates when a specific reporting period is covered.)

Contract Report

8. SPONSORING ACTIVITY (The name of the department project office or laboratory sponsoring the research and development – include address.)

Defence R&D Canada – Ottawa
3701 Carling Avenue
Ottawa, Ontario K1A 0Z4

9a. PROJECT OR GRANT NO. (If appropriate, the applicable research and development project or grant number under which the document was written. Please specify whether project or grant.)

13jb

9b. CONTRACT NO. (If appropriate, the applicable number under which the document was written.)

10a. ORIGINATOR'S DOCUMENT NUMBER (The official document number by which the document is identified by the originating activity. This number must be unique to this document.)

10b. OTHER DOCUMENT NO(s). (Any other numbers which may be assigned this document either by the originator or by the sponsor.)

DRDC Ottawa CR 2008-314

11. DOCUMENT AVAILABILITY (Any limitations on further dissemination of the document, other than those imposed by security classification.)

Unlimited

12. DOCUMENT ANNOUNCEMENT (Any limitation to the bibliographic announcement of this document. This will normally correspond to the Document Availability (11). However, where further distribution (beyond the audience specified in (11) is possible, a wider announcement audience may be selected.))

Unlimited

13. ABSTRACT (A brief and factual summary of the document. It may also appear elsewhere in the body of the document itself. It is highly desirable that the abstract of classified documents be unclassified. Each paragraph of the abstract shall begin with an indication of the security classification of the information in the paragraph (unless the document itself is unclassified) represented as (S), (C), (R), or (U). It is not necessary to include here abstracts in both official languages unless the text is bilingual.)

The Synthetic Environment Distributed Resources (SEDR) Tool Suite is a set of three applications to facilitate knowledge management, accessibility, and repeatability of simulation exercises. The SEDReC Station is the application that automates the start-up, execution and shutdown of a complex, distributed set of simulation resources that form a simulation system. This manual starts with overview and background information on the application. In the remaining chapters the Functional and Non-Functional requirements are given.

Le kit d'outils des ressources distribuées des environnements synthétiques (SEDR) est un ensemble de trois applications misant a faciliter la gestion des connaissances, l'accessibilité, et la répétabilité des exercices de simulation. La Station SEDReC est l'application qui automatise le démarrage, l'exécution et la fermeture d'un système complexe et distribué de ressources de simulation. Ce manuel offre une vue d'ensemble de l'application et des informations de base, suivi par les exigences fonctionnels et non fonctionnels du système.

14. KEYWORDS, DESCRIPTORS or IDENTIFIERS (Technically meaningful terms or short phrases that characterize a document and could be helpful in cataloguing the document. They should be selected so that no security classification is required. Identifiers, such as equipment model designation, trade name, military project code name, geographic location may also be included. If possible keywords should be selected from a published thesaurus, e.g. Thesaurus of Engineering and Scientific Terms (TEST) and that thesaurus identified. If it is not possible to select indexing terms which are Unclassified, the classification of each should be indicated as with the title.)

Modeling & Simulation, Synthetic Environments, Distributed Simulation, Data Management, Knowledge Management, Remote Access, Software

Defence R&D Canada

Canada's leader in Defence
and National Security
Science and Technology

R & D pour la défense Canada

Chef de file au Canada en matière
de science et de technologie pour
la défense et la sécurité nationale

www.drdc-rddc.gc.ca