White Noise
Software Requirements Specification
ORBIT Interference Subsystem Controller
Version <1.0>

Revision History

Date	Version	Description	Author
<20/09/05>	<1.0>	Initial Specification	Ed White, Kishore Ramachandran, Larry Sasso.

Table of Contents

Software Requirements Specification (SRS)

1. Introduction

1.1 Purpose

This is the SRS for the White Noise Project. It describes the purpose and functionality of the software tool, as requested by the Wireless Information Networks Laboratory (WINLAB) at Rutgers University. It aims to serve as a guideline for developers as well as for future use and maintenance.

1.2 Scope

The objective of this project is to create a web-based software tool that is part of a suite of applications enabling the efficient usage of an indoor wireless experimental facility. The purpose of this tool is to enable the generation and visualization of radio-level signals in wireless experiments run by end-users. These signals could range from simple hardware supported waveforms to complex combinations of arbitrary waveforms provided by the end-user. We aim to provide sufficient documentation to facilitate the work for future teams on this tool.

1.3 Definitions, Acronyms and Abbreviations

Testbed – Experimental Research Facility
Instrument – A system capable of generating radio-level signals or observing them
Node – A computer used in the testbed as part of wireless experiments
WINLAB – Wireless Information Networks Laboratory
ORBIT – Open Access Research Testbed for Next Generation Wireless Networks
HTTP – Hypertext Transfer Protocol
AWGN – Additive White Gaussian Noise
UI – User Interface

1.4 References

- Ivan Seskar, Director of IT, WINLAB, Rutgers University
- ORBIT Indoor Testbed
- NodeHandler
- Agilent Vector Signal Analyzer
- Agilent ESG Vector Signal Generator

1.5 Overview

The rest of this document describes the software requirements for the White Noise Project. In section 2, we describe the end users and their usage patterns, tool perspective, general constraints, and functional data requirements for implementing this project. Section 3 describes the detailed functional requirements for this tool. The remainder of the document mainly discusses reliability requirements and quality attributes.

2. Overall Description

2.1 User Personas and Characteristics

There are two basic user personas, each of which has specific set of access levels and privileges. We differentiate our user personas according to their interactions with the system. *NOTE*: There is no overlap between the personas, since only one user is to be entertained by the system at any point of time.

- Administrator: An end-user with root/administrator privileges with regards to
    - Controlling all applications which includes starting or stopping them as well as changing their configurations.
    - Modifying any file or directory; this includes adding new ones and modifying or deleting existing ones.
    - Changing network interface properties like IP addresses and routing tables.
    - Changing authentication settings like passwords.

  From our tool's perspective, Administrators will be responsible for configuration and quality control – they will review tool configuration settings before they can used. We expect this user to be an expert with regards to the underlying hardware systems with at least an intermediate level of expertise in system administration.

- NodeHandler: A software application that provides a script-based interface to run experiments on the indoor wireless testbed. One aspect of an experiment is provided by our tool, namely the controlled injection of signals and their visualization. NodeHandler will access the functions provided by our tool on behalf of an experimenter.

  From our tool's perspective, we expect this user to know nothing with regards to the hardware systems in use. We also do not expect this user to be anything more than a novice with regards to using software applications.

2.2 Product Perspective

- The tool should provide a standard interface for remote access via the Internet;

2.3 Overview of Functional Requirements

- The primary requirement of the tool is the generation and visualization of radio-level signals in experiments run by the end-users. These experiments are carried out on an indoor wireless testbed that is part of ORBIT.

- The tool will be responsible for interacting with and controlling a hardware subsystem consisting of,

    1. Multiple *instruments*, each of which is responsible for either signal generation or visualization. Each instrument maintains state and has certain rules that should be followed; these rules are dependent, among other things, on the state of this

system.

2. Multiple antennae, which will either radiate signals generated by the respective instruments or be in a position to observe them

3. An RF switching and gain control system that connects the instruments and the antennae. The client has asked us to assume that this switching system can support every possible switching/routing combination. In addition, this system also provides the ability to control each individual antenna's gain.

The tool could either use software libraries to interface with the entire system or use libraries to interface with each hardware component separately. Figure 1 illustrates this concept.

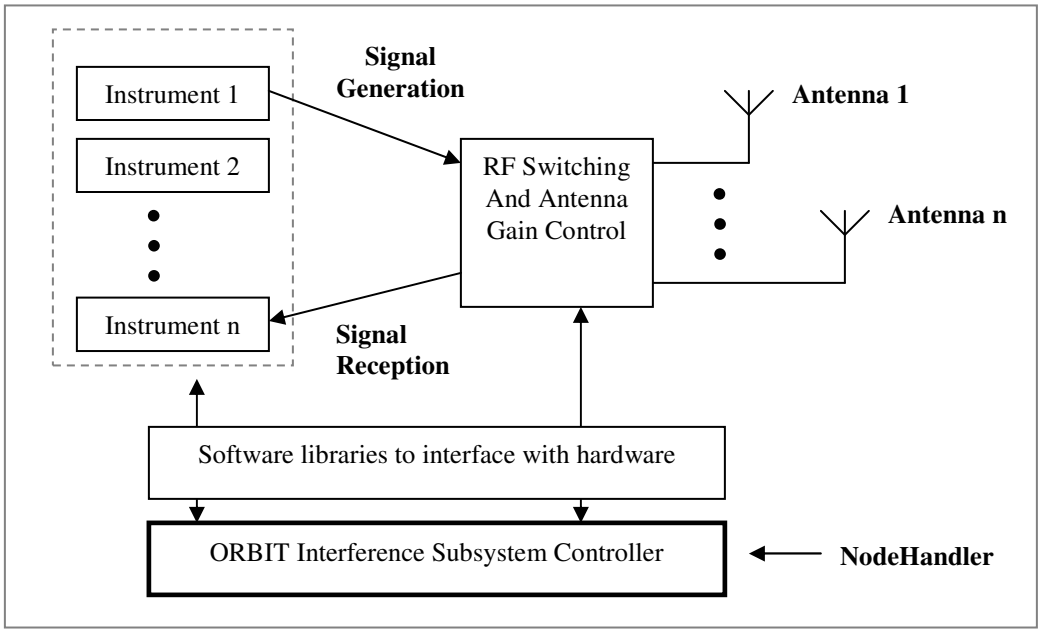


Figure 1. System-level overview of the ORBIT Interference Subsystem Controller

- The tool should support a fine granularity of control for users who wish to specify the actual antennae and their respective functions (transmission or reception) in the experiment. It should also support users who wish to specify only the presence or absence of specific radio signals without regard to the specific antennae used.

- With regards to the interaction between NodeHandler and the signal generating instruments, the tool should support the following:

    1. Functions with a "one-to-one" mapping between the user request and what the instrument supports. For e.g. user requests generation of an in-built signal on a specific channel on all antennae.

    2. Functions, which correspond to "canned" scenarios and could result in the combination of multiple instruments and/or the execution of multiple instrument

commands. For e.g. user requests the presence of a signal that resembles radiation from a microwave oven.

3. Functions, which allow the user to specify arbitrary signals using some form of input and the subsequent generation of the same. The form of this input should be well-defined.

4. Functions, which are a combination of all or any of the three scenarios mentioned above.

5. The tool should provide for storage and display of radio-level visualization data. The storage formats should be well-defined.

2.4 Overview of Data Requirements

Storage and well-defined formats will be required for hardware architecture data provided by the administrative user.

Storage, well-defined formats and storage policies for the size of data will be required for user data describing the waveforms that are to be generated.

Storage, well-defined formats and storage policies for the size of data will be required for the radio-level visualization data as well.

2.5 General Constraints, Assumptions, Dependencies, Guidelines

- It is assumed that the end-user has access to the Internet.
- It is assumed that only one user will access this tool at any point of time. Further, this prevention of multi-user access is provided by the ORBIT hardware and software infrastructure.
- Our tool does not address user authentication either and depends on the underlying ORBIT infrastructure for the same.

3. Specific Requirements

3.1 Functionality

A customizable, hardware systems architecture is defined by an administrative user, consisting of:

- A set of antennae and constraints upon them
- A set of available instruments
- The RF switching and gain control system between the instruments and the antennae

Our tool uses this architecture specification to provide a software interface that

1. Controls signal generation and visualization,
2. Responds to remote requests for service,
3. Supports script-based execution of all functions supported,
4. Provides sanity checks before relaying user requests to the hardware, for e.g. in response to a signal generation request, making sure that at least one signal generator instrument is available and the requested numbers of transmitting antenna are available before request execution.
5. Uses existing, standard Internet protocols to communicate.

Thus, http has been specified as the underlying protocol upon which this interface must be built. Actual implementation may consist of html forms, scripts, and/or web services.

3.1.1 User types and separation of access

Users of the system fall into two types: 'Administrators' and 'NodeHandler'

Administrators: Administrative users are responsible for maintaining and modifying the hardware architecture (modifications could include the addition or removal of instruments and antennae). They are also responsible for relaying the architecture changes to our tool using a well-defined format. It is assumed that these users will have elevated access and control over the complete subsystem including our tool.

In order to provide the administrator with an interface to update the hardware architecture, a couple of choices exist. One is that of input obtained from an online HTML form page during runtime. Information from this form is captured and used for future experiments. The other option is that of reading this information from a file or a database and requiring a restart after each change. A couple of reasons as to why we opt for the file interface are as follows:

(a) These architecture changes are expected to be rare events
(b) The administrator and the experimenter are expected not to access the tool simultaneously. Thus, the requirement is only for this change to take effect in the *next* experiment. Additionally, these changes may have to be tested before their introduction into main-stream usage.
(c) Ease of use and simplicity of implementation

A sample file format for hardware architecture information is shown in figure 2.

Entry No.	Instrument ID	Instrument Type	Name	Feature 1	Constraint 1
1	01	ESG	Some string	A	L
2	02	VSA	10.12.2.1	B	M
3	03	RF Switch		C	N

ESG: Signal Generator VSA: Signal Analyzer
Example Feature: Range of frequencies supported.
Example Constraint: maximum power levels instruments can generate.

Figure 2. Sample file format for Administrator to enter hardware architecture information.

A sample file format for antenna information is shown in figure 3.

Entry No.	Antenna Functionality	Approximate Position
1	Tx	(x1,y1)-(x2,y2)
2	Rx	(x3,y3)-(x4,y4)
3	Tx	(x5,y5)-(x6,y6)

Tx : Transmitting antenna Rx: Receiving antenna
(xi, yi) – (xj,yj): Position of antenna in the indoor testbed, with respect to the nodes.

These file formats are preliminary and we will need to understand the features provided by the software interfaces. The tool does not have to check for configuration changes during run-time.

NodeHandler: As specified earlier, this is a software application that interacts with our tool on behalf of an experimenter to perform signal generation and monitoring. We see no need to expose system configuration functionality to this end-user – we aim to provide an opaque interface to underlying hardware systems. The tool should also protect the hardware from malicious as well as ignorant users and return descriptive error statements where applicable. We expect inadvertent mistakes such as,

- Typographical errors in the NodeHandler script
- Errors in the NodeHandler script for functions that do not exist or are currently unsupported.

We also aim to provide protection from malicious attacks such as setting the transmit power on the signal generation antennas to a value that could damage the hardware. This could result in a denial-of-service to the next experimenter if a particular function is unavailable due to hardware malfunction.

3.1.2 Administrators can specify a set of antennae and their individual function (transmission or reception)

Administrators should have a mechanism whereby they can define the usable set of antennae along with a specification of whether each one transmits (‘T’) or receives (‘R’).

Name: Admin Antenna Change
Precondition: Administrator is logged in and is the only end-user using this tool.
Main flow of Events:

1. Log into the computer on which this tool is running using administrator privileges.

2. Stop this application.
3. Apply changes to the antenna configuration file in the specified format.
4. Our tool checks syntax and consistency of the configuration files.
5. Restart application
6. Run test suite of commands before allowing experimenters to use this tool.

Priority Level: 1 (To be considered for implementation this semester)

3.1.3 Administrators can update list of available instruments and their features.

Administrators can add/remove instruments from the available set for which access is allowed to experimenters. While adding instruments, they will include relevant constraints and features in the instrument configuration file.

Name: Admin Instrument Change
Precondition: Administrator is logged in and is the only end-user using this tool.
Main flow of Events:

1. Log into the computer on which this tool is running using administrator privileges.
2. Stop this application.
3. Apply changes to the instrument configuration file in the specified format and directory.
4. Our tool checks syntax and consistency of the configuration files.
5. Restart application.
6. Run test suite of commands before allowing experimenters to use this tool.

Priority Level: 1 (To be considered for implementation this semester)

3.1.4 NodeHandler can request to generate a signal, optionally specifying which antenna should be used.

NodeHandler can provide information about what type of signal to generate, along with which antenna(s) to connect this signal to for transmission. The front-end will validate this request in the context of the current system state, and interface with the system to connect an appropriately configured signal generation instrument to the requested antenna(s).

Seven sample signal types we are beginning to look into are sine, cosine, triangle, square, positive ramp, negative ramp and AWGN. NodeHandler will need to provide the amplitude, frequency, phase and duration of the signal. Any item determined to be out of range will trigger an error to be returned and no signal generation will occur.

Antenna requests will be validated based on the constraint information stored in the antenna configuration file.

Name: Signal Generate
Precondition: NodeHandler has requested signal generation
Main flow of events:

1. Process request from NodeHandler (antenna, waveform, gain, frequency, phase, duration)

2. Check specified information for proper value range

What is "proper" value range?

3. Check specified information relative to antenna configuration file
4. If specifications are within range, send commands to equipment to generate signals, otherwise return error to NodeHandler
5. Signal generation ends on the execution of the "stop generation" command from NodeHandler.

Priority Level: 1 (To be considered for implementation this semester)

3.1.5 NodeHandler can request to observe and store spectrum information, optionally specifying an antenna, during an experiment

NodeHandler can request to observe the signal being received on a given antenna. The front-end will validate this request in the context of the current system state, and interface with the system to connect an appropriately configured signal observation instrument to the requested antenna. Observed data will be transported to a database using the ORBIT measurement library (OML) framework.

We expect this each observation to be a single floating point value. We also expect to provide an interface where NodeHandler can specify both the number and type of measurement.

Name: Signal Observe
Precondition: NodeHandler has requested signal observation
Main flow of events:

1. Process request from NodeHandler (antenna to observe signal on, type of observation, number of observations)
2. Check whether an instrument that supports the type of observation is available.
3. If instrument is available, send a "trigger" message asking the instrument to observe (for a particular duration).
4. Once duration elapses, ask instrument for recorded observation values.
5. Recorded observation values are then sent to a database using the OML framework.
6. Signal observation ends on the reception of the "stop observation" command from NodeHandler.

Priority Level: 2 (To be considered for implementation only if other higher priority tasks are completed)

3.1.6 NodeHandler can reset the state of hardware systems

NodeHandler can reset the state of all instruments in use. A reset can be performed at any time during an experiment.

Name: Reset
Precondition: Experiment is about to begin or a requested experiment is currently running
Main flow of events:

1. Instruct all instruments to stop.
2. Store any data collected prior to the execution of this command (if NodeHandler had

made such a request earlier).
3. Instruct all instruments to reset.

Priority Level: 1 (To be considered for implementation this semester)

3.1.7 NodeHandler can generate signals from "canned" scenarios

The NodeHandler can request a "canned" scenario, which may require the participation of multiple instruments simultaneously and multiple antennae.

Name: Canned Signal Generate
Precondition: NodeHandler has requested canned signal generation
Main flow of events:
1. Receive canned scenario specification from NodeHandler
2. Use multiple instances of Signal Generate to run experiment
3. Proceed as described in 3.1.4

Priority Level: 2 (To be considered for implementation only if higher priority tasks are completed)

3.1.8 NodeHandler can request generation of user-defined signals

The NodeHandler can request a "user-specific" scenario, which may require the participation of multiple instruments simultaneously and multiple antennae. Formats need to be specified for user input.

How does user define the signal???

Name: User-defined Signal Generate
Precondition: NodeHandler has requested user-defined signal generate
Main flow of events:
1. Receive scenario information from NodeHandler
2. Verify input for format inconsistencies and checks to determine whether available instruments can support it
3. Generate signal using Signal Generate
4. Proceed as described in 3.1.4

Priority Level: 2 (To be considered for implementation only if higher priority tasks are completed)

3.2 Usability

3.2.1 Administrative Users

Administrative users are likely to be more familiar with the hardware systems, our tool and ORBIT. A functionally correct front-end interface can be achieved through the use of configuration files. These files will be text-based with a specified format.

3.2.2 NodeHandler

The end-users in this category may not be familiar with ORBIT and our tool and are to be shielded from the instruments and related hardware. The functionality required should be abstracted to a user-interface which takes high-level requests, and hides the details of how those requests are executed.

3.3 Reliability

We expect our tool to provide a robust, 24/7/365 online service, which is brought down only during maintenance and architecture upgrades.

3.4 Performance

No specific performance requirements are specified.

3.5 Supportability

Upon delivery, our tool should be documented in such a way as to be fully supported by Administrative users, as the original developers will not be available for future enhancements. See also 3.7.

3.6 Design Constraints

3.6.1 Possible use of Visual Studio .NET for interfacing with Agilent libraries

The Agilent libraries allow simplified control of the hardware and are available on the Microsoft .NET platform. If the Agilent libraries are to be used, programming must be done using Visual Studio .NET. The languages available in Visual Studio .NET include Visual Basic and C#.

3.7 On-line User Documentation and Help System Requirements

Our tool should be documented for future administrative users, developers and NodeHandler. The current team of developers will not be available after initial implementation and therefore all aspects, including the requirements specification and system design should be documented as well.

Documentation for NodeHandler should be written without any assumptions of previous knowledge of ORBIT or this tool.

3.8 Purchased Components

Our tool must be implemented without any additional purchased components.

3.9 Interfaces

3.9.1 User Interfaces

The user interface must offer remote access to RF signal generation and observation. In addition, we expect the common mode of access to be via nodeHandler, which implies that, our user

interface, be "scriptable". Given these requirements, we are leaning towards a web-services based interface, which would leverage the universal nature of HTTP for remote access. Additionally, this would enable remote procedure calls in an extensible manner. SOAP/XML requests and responses will be sent over an HTTP link and we could provide support for this interface in many programming and scripting languages.

For any language, we expect to provide the following methods:

```
public string Generate(String[] names, String[] values)
public string Observe(String[] names, String[] values)
public string GenerateCanned(String cannedname)
public string Reset()
public string StopGenerate(String request_id)
public string StopObserve(String request_id)
public string ReturnSupportedSignals()
```

3.9.2 Hardware Interfaces

The hardware involved is controllable directly by socket connections or by the Agilent libraries. If the software libraries are not used, specific interface information will be needed to interface with the hardware using GPIB and other direct command protocols. In this case an expansion card would be required to provide connectivity with the hardware.

3.9.3 Software Interfaces

The Agilent .NET libraries could be used. In this case programming must be done using Visual Studio .NET. Otherwise any development environment and language could be used.

The program must interface with a registered user's time-slot list that is created externally by another piece of software.

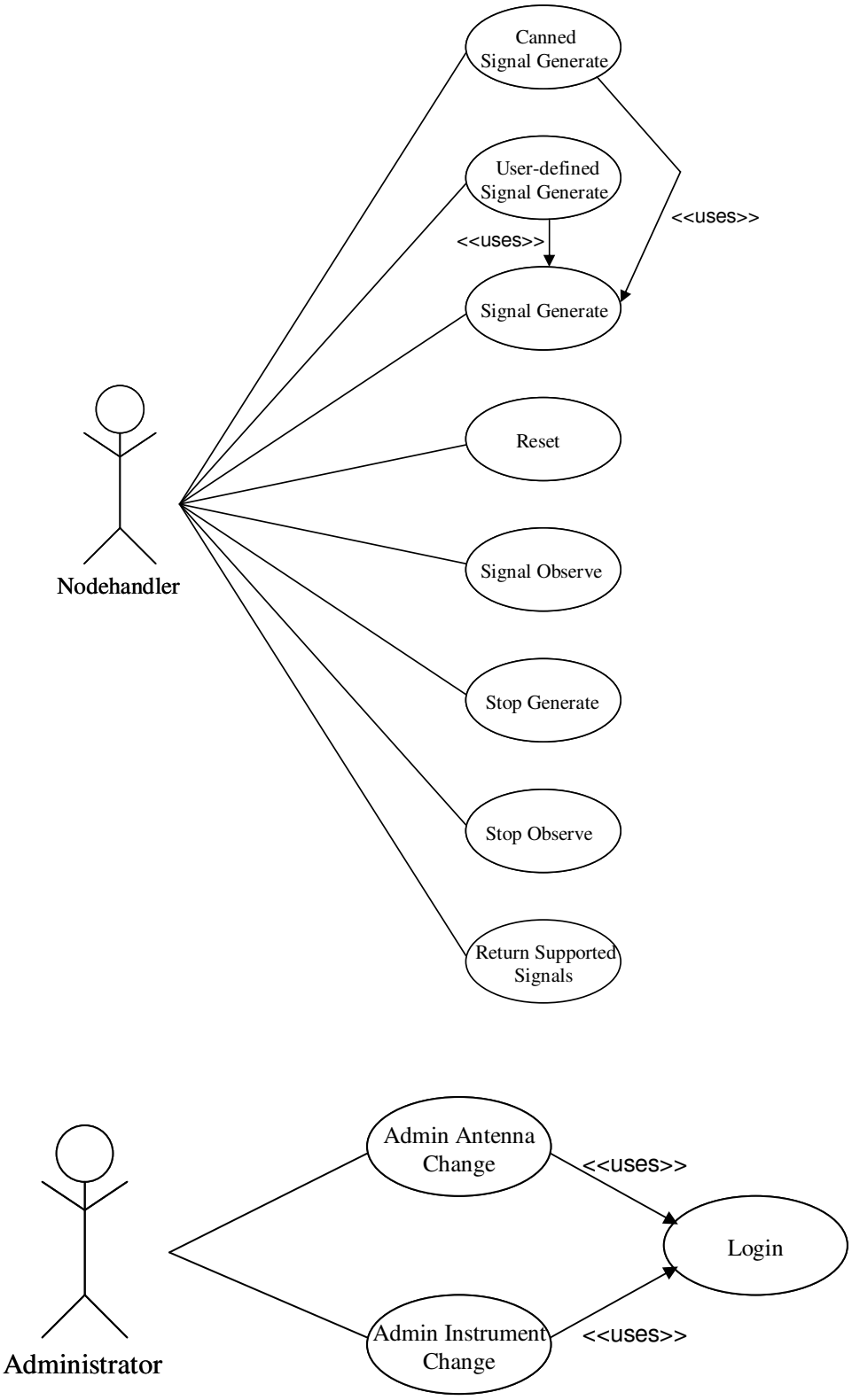
3.9.4 Communications Interfaces

Various communications schemes are used by the equipment including GPIB and proprietary protocols. Communication could be simplified by using the Agilent .NET libraries to control communication.

3.10 Licensing Requirements

Any additional components used aside from the Agilent libraries must not incur any intrusive licensing costs or other constraints.

Use case diagrams:


Canned Signal Generate
User-defined Signal Generate
<<uses>>
<<uses>>
Signal Generate
Reset
Signal Observe
Stop Generate
Stop Observe
Return Supported Signals
Nodehandler
Admin Antenna Change
<<uses>>
Login
Admin Instrument Change
<<uses>>
Administrator