

# THE WAVEFORM DASHBOARD: AN INTERACTIVELY CONFIGURABLE GUI FOR PROTOTYPE SCA-BASED SDR WAVEFORMS

Deepan .N. Seeralan (Dept of Computer Science, Virginia Tech, Blacksburg, VA, USA; deepanns@vt.edu); Stephen Edwards (Dept of Computer Science, Virginia Tech, Blacksburg, VA, USA; edwards@cs.vt.edu); Carl Dietrich (Bradley Dept of Electrical and Computer Engineering, Virginia Tech, Blacksburg, VA, USA; cdietric@vt.edu)

## ABSTRACT

The Software Communications Architecture (SCA) is prominent in development of software defined radio (SDR) waveforms for wireless communication systems. Prototyping of SDR waveforms is an interactive process that benefits from user friendly tools. In this paper, we present one such tool, the Waveform Dashboard. This tool was developed to enable interactive configuration of component properties of SDR waveforms, helping to streamline the rapid prototyping and testing process. The SCA standards define the Common Object Request Broker Architecture (CORBA) interfaces to configure component properties and a means to identify the component properties via the PRF files that enables creation of a generic GUI to work with SDR waveforms. The core framework and tools of the Open-Source SCA Implementation Embedded (OSSIE) developed by Wireless @ Virginia Tech provide a convenient environment for rapid prototyping of SCA-based waveforms. Our tool abstracts the OSSIE core framework and presents an interactive and customizable interface to configure the component properties of SDR waveforms. This paper describes the design and implementation of the Waveform Dashboard and its applications to SDR waveforms.

## 1. INTRODUCTION

Interactive control of software defined radio (SDR) waveform application parameters at run time enhances SDR education, research, and rapid prototyping, and is essential in many practical SDR implementations. Unlike end users, however students, researchers, and developers may desire the capability to control all or a large subset of parameters of an SDR waveform application. Moreover, it may not be efficient to invest significant time in developing custom user interfaces for waveforms that will not be employed by end users. Thus, there is a natural demand for rapid generation of easily customizable graphical user interfaces that provide control of such waveform applications. Such a capability frees communication engineers and students from developing GUIs, and frees user interface (UI) developers to

concentrate their efforts on UIs for production SDR platforms.

For SDR waveforms based on the Software Communications Architecture (SCA) [1], methods and profiles that are integral to the core framework and application software can be leveraged to provide the desired capability. This paper describes the Waveform Dashboard, a flexible GUI tool that was developed and tested using open-source SCA-based SDR software from the OSSIE project [2] and is now distributed with OSSIE.

## 2. THE WAVEFORM DASHBOARD

The Waveform Dashboard (WaveDash) is an interactive and configurable GUI tool that acts as a one point control panel to work with multiple SDR waveforms and their components. This tool abstracts the SCA interfaces and implementation of OSSIE core framework and provides a direct manipulation interface to control and configure SDR waveforms interactively. Users can install, start, stop and uninstall a waveform and also configure the components' properties in real time. Further it allows the user to customize the interface on a per-waveform per-component basis. Users can choose the type of widget they prefer to represent a particular property of a component and can also hide/unhide the components and its properties they wish to.

We developed this tool in Python to provide better platform independence, and interoperability with other tools in the Waveform Workshop (OSSIE Tools suite). Other packages we used in developing WaveDash are:

- *wxPython* [3]– an open-source toolkit for GUI programming in Python
- *omniORB* [4]– a free CORBA ORB for Python
- *xml.dom.minidom* [5] – an open-source lightweight XML parser

### 3. DESIGN

WaveDash is based on a simple Model-View-Controller (MVC) architecture. The model acts as a database of waveforms and applications running on the node and the user customizations of the GUI. The view simply renders a graphic representation of the information contained in the model.

The main modules of WaveDash are:

- *Wavedash Model* – comprises three sub-models, one each to represent Waveform, Component and Properties
- *Wavedash View* – combines the view and controller logic and renders the visual representation of the model.
- *Wavedash Utils* – utility functions to access OSSIE core framework, parse eXtensible Markup Language (XML) files using xml.dom.minidom and Widget Container which maps properties to GUI widgets.

A simplified class diagram of the WaveDash model is shown in Fig 1.

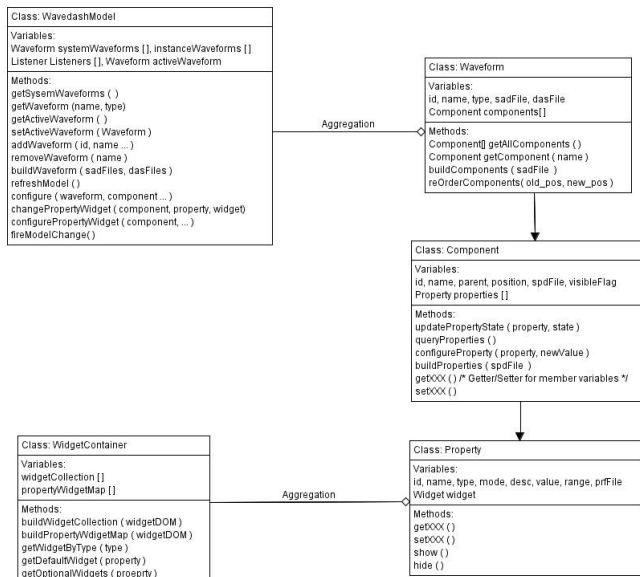


Fig. 1 Simplified class diagram of WaveDash Model

#### 3.1. Use Cases

The lifecycle of an interaction in WaveDash comprises of installing, starting and stopping, configuring and uninstalling a waveform (shown in Fig 2.). It also includes several use cases during this interaction lifecycle. The user can do any of the following:

- Install and start a waveform or just install a waveform from the SDR waveforms listed in the Waveforms menu.
- Start and stop a waveform multiple times
- Work on the recently installed waveform or choose another waveform from the list of installed waveform instances. User can then start, configure, stop or uninstall a waveform.
- Preview a waveform before installation.
- Reorder the layout of a waveform's GUI by moving a component panel up or down.
- Change the value of one or more property of one or more component of the active waveform. An error message should be shown if the value is not compatible with the data type of the property being modified. For e.g., entering a float value for a property which takes only integer values.
- Change the GUI widget of a property using the context menu of a property. For e.g., properties with integer values are mapped to a Text Box by default. User can change to a spin box or a slider when required.
- Hide or unhide one or multiple properties of a component and hide or unhide one or more components of a waveform as well.
- Maintain changes to a waveform layout or component properties when switching between multiple waveforms.
- Uninstall a waveform.

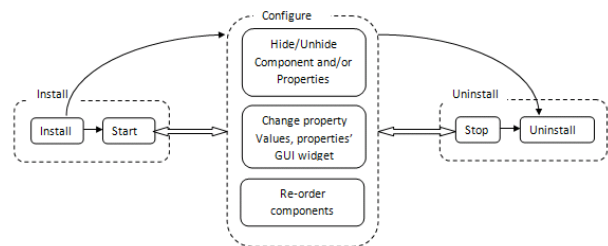


Fig. 2 A typical interaction lifecycle in WaveDash

### 4. IMPLEMENTATION

#### 4.1. Widget Container

The Widget Container addresses one of the main objectives of WaveDash to directly manipulate the GUI widgets of properties and customize the interface to their preferences. The SCA standards specify the data type (e.g. *float*, *char*) and XML type (e.g. *simple*, *simplesequence*) for properties of each component. WaveDash maintains two set of widgets for each of the XML type and data type tuple: one acts as a default widget and other acts as optional set of widgets. Users can change the widget of a property to another widget only from the list of optional widgets. This prevents the user

from choosing an incompatible widget for a property. The mappings are maintained in a configuration file and are loaded during WaveDash startup. Table 1 shows the mapping for some of the property types.

Table 1. Property-Widget mappings

Property Type (XML Type,DataType)	Default Widget	Optional Widgets
( simple, long )	Text Box	Spin box, Slider
( simple, integer )	Spin Box	Text box, Slider
( simple, double )	Text Box	Slider
(simple, float )	Text Box	Slider
( simple, short )	Spin box	Text box, Slider
( simple, char )	Text Box	None
(simplesequence, short )	Text Box	None
(simplesequence, string)	Text Box	None
(simplesequence, float)	Text Box	None

## 4.2. SCA Domain Profiles

On startup, WaveDash looks for waveforms in a preconfigured directory (e.g./sdr/waveforms) and builds its model structures. In the SCA, the Domain Profile includes a set of XML files that describes the set of interfaces, interdependencies and interconnections between components, and component properties and their values, of the waveforms in the domain [1]. These files are:

- *Software Assembly Descriptor* (SAD) – used to describe a waveform. It includes the list of components, their logical relationships and connections. It is through this file WAVEDASH explores the components of a waveform.
- *Software Package Descriptor* (SPD) – describe the components and their implementations. The location of SPD file is obtained from the SAD file and the SPD file provides the location of the PRF files.
- *Properties Descriptor* (PRF) – Specifies the data type, value and description for each of the properties of a component.

## 4.3. Initialization

During initialization, the SAD file of each of the waveform is parsed to learn its components and the location of their SPD files. The SPD file of a component is then parsed to find the properties and their PRF files. The PRF provides the details of every property of a component. The property type which is a tuple of (xml type, data type) for each property is looked up in the Widget Container to find the right GUI widget for it. This procedure is repeated for all the SAD files found in the configured directory. The waveform names will be populated in the *Waveforms* menu of WaveDash. The system found waveforms are listed under *Waveforms* menu (Figure 3) and the waveform applications are shown in a tabbed fashion. The *Components* menu will be refreshed with the names of the components of the currently selected waveform application (Figure 4) as determined by the active tab.

The initialization process also looks for currently installed waveform applications in the domain through the Application Factory interfaces. It also looks for single components installed as a waveform application through ALF. If any waveform application is found running, the Application Factory returns an instance of the application with which the properties of a component can be queried or configured. The property values are queried and model is then updated with the new waveform application. Hence a waveform which installed from command line or ALF can also be controlled from WaveDash.

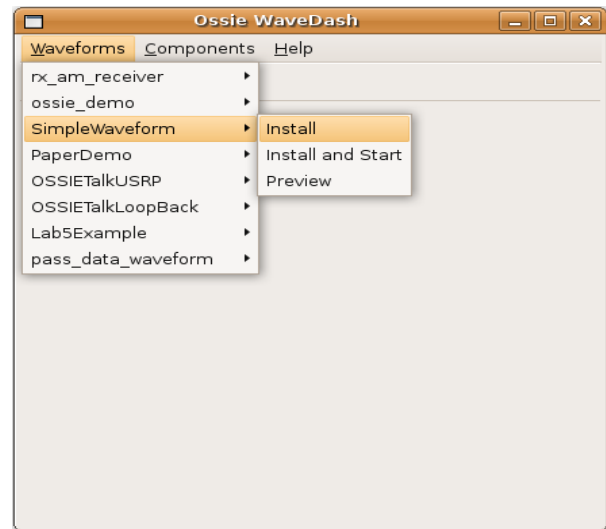


Fig. 3 Waveform Menu Listing

## 4.4. Operation

The WaveDash GUI represents a waveform application as one single panel which in turn consists of multiple panels, one for each component. Tabs are added as and when a new waveform application is installed. Each component panel

contains a label and a default widget for each of its properties. A context menu is pinned to each of the property widget and the component panel to directly manipulate them. The component context menu is used to move a component panel up or down and hide/unhide the properties (Figure 5).

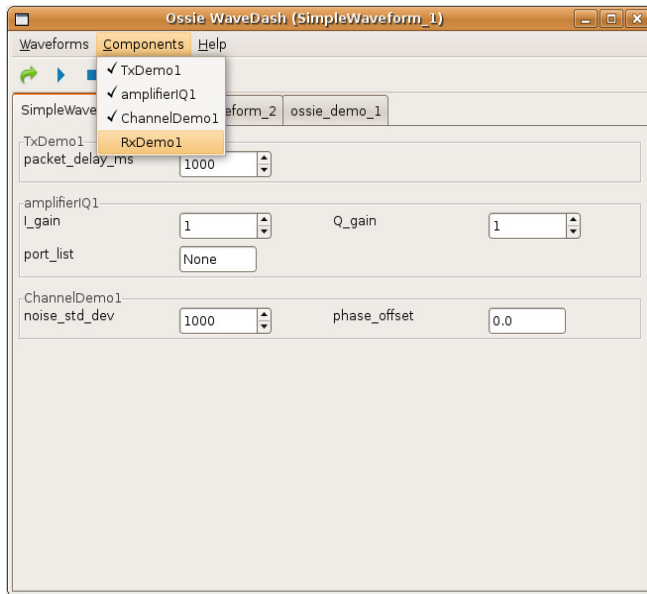


Fig. 4 Component Menu Listing

The property context menu lists the default widget and the optional widgets as obtained from the Widget Container for the user to change from one widget to another (Figure 6). Certain widgets like Spin Box and Slider have minimum and maximum values that can also be configured from the options in the context menu.

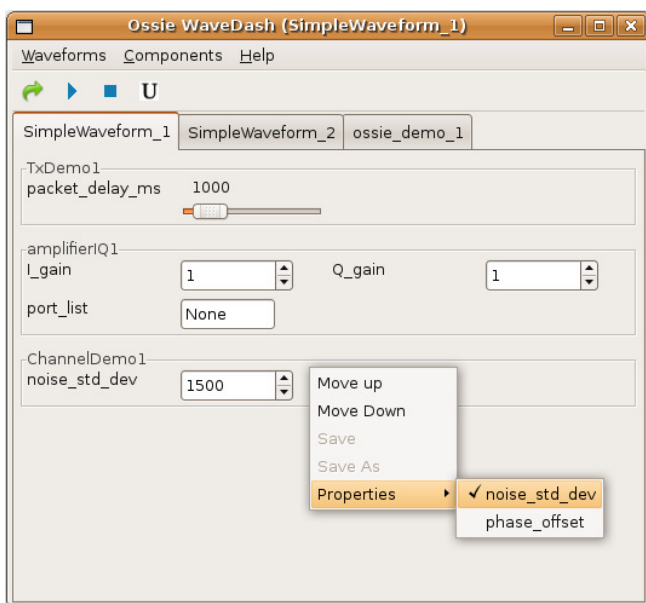


Fig. 5 Component Context Menu

When the user changes the value of a property in the GUI, the value is read and validated for type compatibility before configuring the component. If the value is valid, the *configure()* [1] method is invoked on the application instance obtained from the Application Factory. On success, the GUI is refreshed with the new value and on failure the old value of the property is retained.

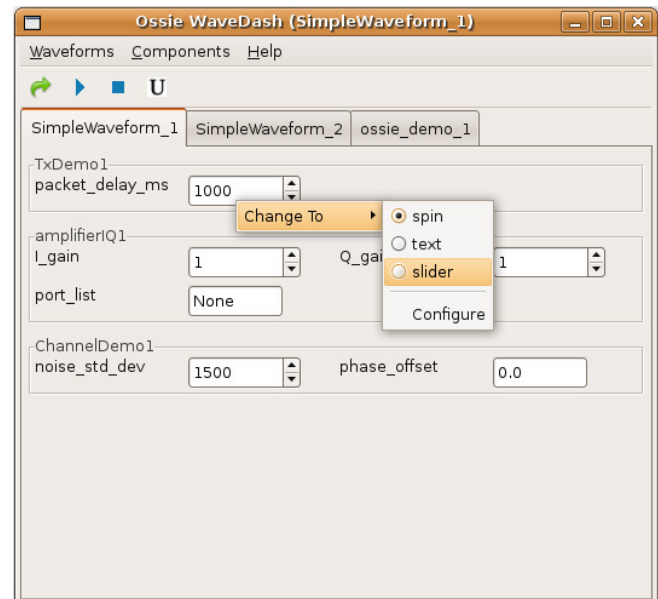


Fig. 6 Property Context Menu

## 5. ENHANCEMENTS

The current implementation of the Waveform Dashboard allows users to configure the component properties of a waveform and customize the GUI to their preferences. While changes to waveforms will persist as long as the waveform is running on the device, changes to GUI will not. In the future, WaveDash will be extended to allow the users to save their GUI preferences on a per-waveform and per-component basis. On startup, the saved preferences will then be applied to the matching waveforms. The GUI will also be updated to incorporate the components' ports and interfaces and the connection information as well. Integrating WaveDash with ALF (visualization and debugging environment for OSSIE waveforms) will significantly improve the debugging experience.

## 6. CONCLUSION

The Waveform Dashboard (WaveDash) provides an interactive and configurable interface to configure component properties of waveforms that makes the SDR prototyping in OSSIE more efficient than before, and is expected to prove useful for SDR education and research as well. Using WaveDash, component properties of a

waveform can be queried and configured in real time, which facilitates rapid development and debugging of SDR waveforms. Moreover, WaveDash demonstrates a capability that can be achieved using the SCA or similar SDR architectures.

## 7. REFERENCES

- [1] Software Communications Architecture, Available at <http://sca.jpeojtrs.mil/>
- [2] Open-Source SCA Implementation Embedded, Available at <http://ossie.wireless.vt.edu/>
- [3] wxPython - a GUI Toolkit for Python, Available at <http://wxpython.org/>
- [4] omniORB - a CORBA ORB for Python, Available at <http://omniorb.sourceforge.net/>
- [5] xml.dom.minidom, a light-weight implementation of Document Object Model Interface, Available at <http://docs.python.org/library/xml.dom.minidom.html>
- [6] N.Rapping and R.Dunn, *wxPython in Action*, Manning Publications, March 2006