TOWARDS AN OPEN-SOURCE INTEGRATED DEVELOPMENT ENVIRONMENT FOR SOFTWARE-DEFINED RADIO WORK

Stephen H. Edwards (Virginia Tech, Blacksburg, VA; edwards@cs.vt.edu);
Jason Snyder (Virginia Tech, Blacksburg, VA; snyder84@cs.vt.edu);
Carl Dietrich (Virginia Tech, Blacksburg, VA; cdietric@vt.edu)

ABSTRACT

Developing software-defined radio components and waveforms based on the Software Communications Architecture can be tedious. Commercial SDR tools provide a wide range of capabilities to address this. Meanwhile, existing open source work has provided some graphical tools for designing such assets, but full support across the development process is lacking. This paper describes work to combine graphical tools from the Open Source SCA Implementation::Embedded with a modern, extensible, integrated development environment: Eclipse.

1. INTRODUCTION

The Software Communications Architecture (SCA) defines a standard for software-defined radio components and waveforms. Developing SCA-based software can be an involved process, however.

The OSSIE (Open Source SCA Implementation::Embedded) project [1][2] provides a handful of tools to help address this problem. As an open-source implementation based on the SCA, OSSIE provides a platform that allows developers to run software-defined radio components and waveforms. OSSIE also provides several python tools to aid designers in the creation of new components and waveforms. While these tools simplify the process of designing new SDR components and waveform applications, generating necessary XML descriptions, and even generating skeleton program code, they still address only a small slice of the development cycle.

To strengthen the support that OSSIE provides for software-defined radio development activities, this paper describes efforts to combine existing OSSIE prototyping tools with an industrial-strength integrated development environment: Eclipse. By integrating OSSIE tools into Eclipse, developers can leverage the broad base of support that such tools provide for editing, compiling, debugging, change control, build management, and deployment.

2. EXISTING OSSIE TOOLS

The development tools provided by OSSIE include OWD (the OSSIE Waveform Developer) and ALF. OWD allows developers to create new components, or to link up several existing components to create a new waveform. When creating a new component, OWD provides a graphical user interface (GUI) for the developer to define the component's ports, properties, and other information. OWD can then automatically generate the XML profiles that describe the component, together with skeleton implementation code in either C++ or Python. When creating a new waveform, OWD provides a GUI allowing the developer to click and select components and add them to the waveform, specify the nodes that exist on the target platform, identify the various devices available on each node, and allocate components to devices. OWD generates the XML profiles that describe the waveform and the platform deployment strategy.

While OWD simplifies development tasks, ALF is a GUI-based tool for monitoring execution. It allows developers to run waveforms, display waveform block diagrams, and debug waveforms in real time using either provided or user-developed plug-in tools.

3. EXISTING ECLIPSE CAPABILITIES

Eclipse [3] is a popular, industrial-strength, open-source integrated development environment (IDE). It is most well known as the dominant professional IDE for Java development. Eclipse is based on a flexible plug-in architecture that allows it to be extended in a variety of ways easily. As a result, development support for many programming languages other than Java is now available, including C++, Python, Perl, PHP, and many more. In particular, the C/C++ Development Tooling (CDT) project provides comprehensive development support for C and C++ applications within Eclipse, including support for a variety of compilers and cross-compilers. It is particularly popular among g++ developers.

Because of the maturity of Eclipse, a wide range of professional-quality development features are readily

available through plug-ins published on the web. Some of these features, including those from the CDT, are:

- Project management and incremental build management for supported languages, including makefile-based or ANT-based control of C/C++ projects.
- Seamless integration with popular version control software, including CVS and Subversion.
- Full GUI integration of gdb for debugging.
- Editor features such as syntax highlighting, command completion, file cross-referencing, check-as-you-type syntax error detection, template support for recurring constructs, and personal snippet library support.
- XML editing and checking support.
- Availability of add-ons for creating and running automated XUnit-style unit tests [4].

These features were a major factor in choosing Eclipse as the foundation for an OSSIE-based IDE for software-defined radio work. Most importantly, Eclipse is extensible and allowed us to create a plug-in to recreate the functionality of OWD within Eclipse.

4. INTEGRATION

Previously, the workflow for creating a new component consisted of several steps. The first was to use OWD to generate the XML profiles and skeleton implementation code. Then the developer used a separate editor to fill in the skeleton code. Finally, the component was built and installed from the command line using the make utility. Similarly, waveforms were configured with OWD and then built and installed from the command line.

To simplify this process, the Eclipse IDE was used to create a unified workflow and increase developer productivity. We extended Eclipse by writing an OSSIE plug-in to embed OWD and ALF directly.

Figure 1 shows the new OWD waveform editor, which is modeled after the original Python version. It provides a list of available resources, including nodes, devices, and components. The designer can then add components to the waveform application under design, configure their properties and port connections, and allocate them to devices on the nodes in the current platform model. By right clicking on a component in the waveform editor, the developer can edit the component's properties, as shown in Figure 2.

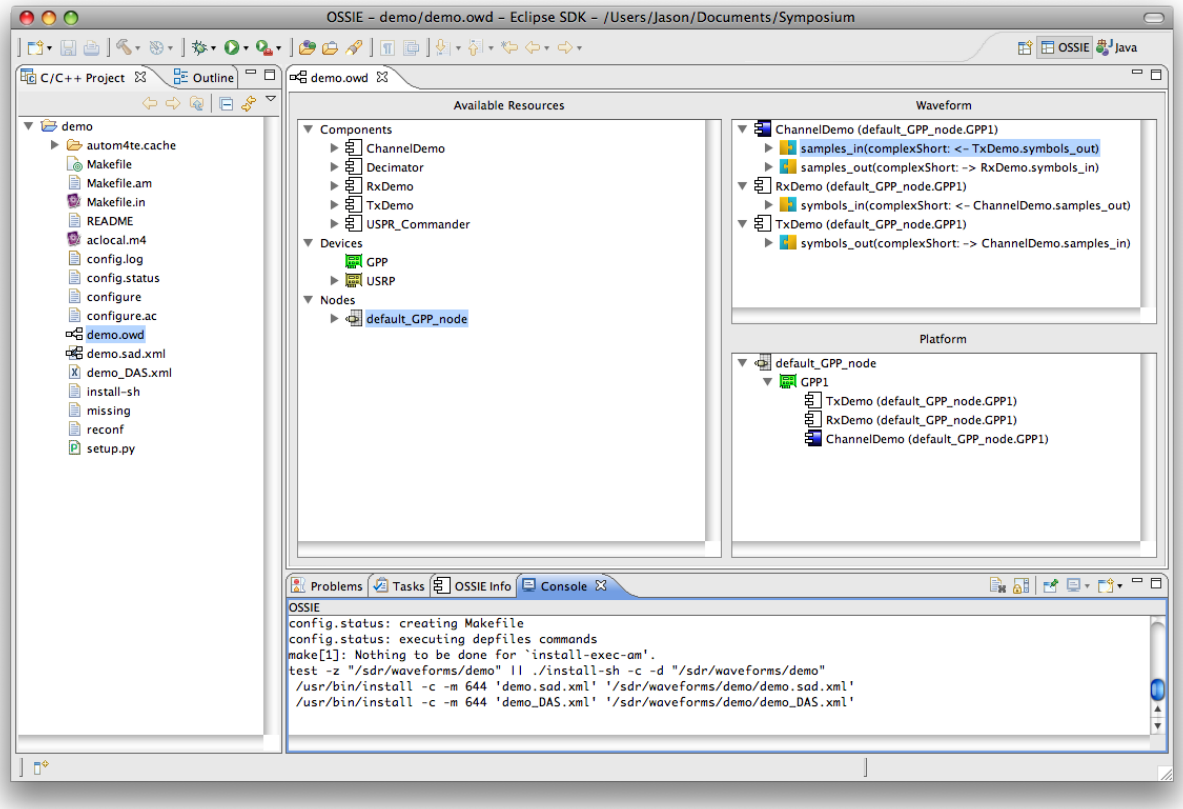

Figure 1: The Eclipse version of the OWD waveform editor.

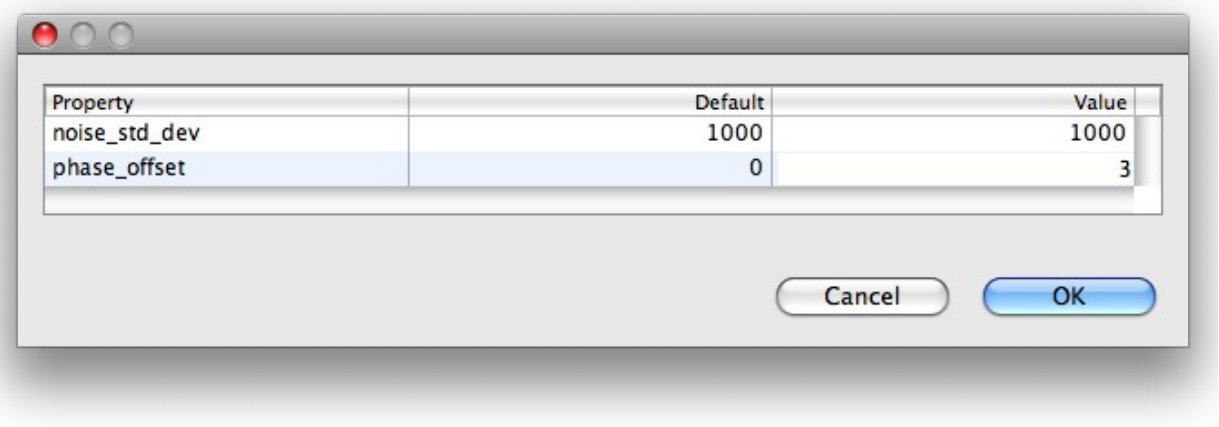

Figure 2: Editing component properties in the Eclipse version of OWD.

When creating new components from scratch, a developer used the component editor shown in Figure 3. Like the waveform editor, the component editor was also based on the original Python implementation. However, we were able to simplify the interface by moving functionality relating to component deployment into the waveform editor. Rather than using the component editor to allocate components to devices, the waveform editor allows the user to simply drag and drop components onto devices to configure their deployment settings.

As component or waveform designs are saved, the corresponding XML files are automatically generated from the design information. Later editing triggers regeneration, so design decisions can be changed at any time.

Eclipse plug-ins are written in Java, with metadata and configuration details in XML. The preexisting OWD code was written in Python and could not be used directly in the plug-in. To bridge this gap, we used Jython, a python interpreter written in Java.

In order to minimize the amount of "glue" code needed between Java and Python, we first determined which parts of OWD to re-implement in Java and which parts to leave in Python. We decided to re-implement the interface using the

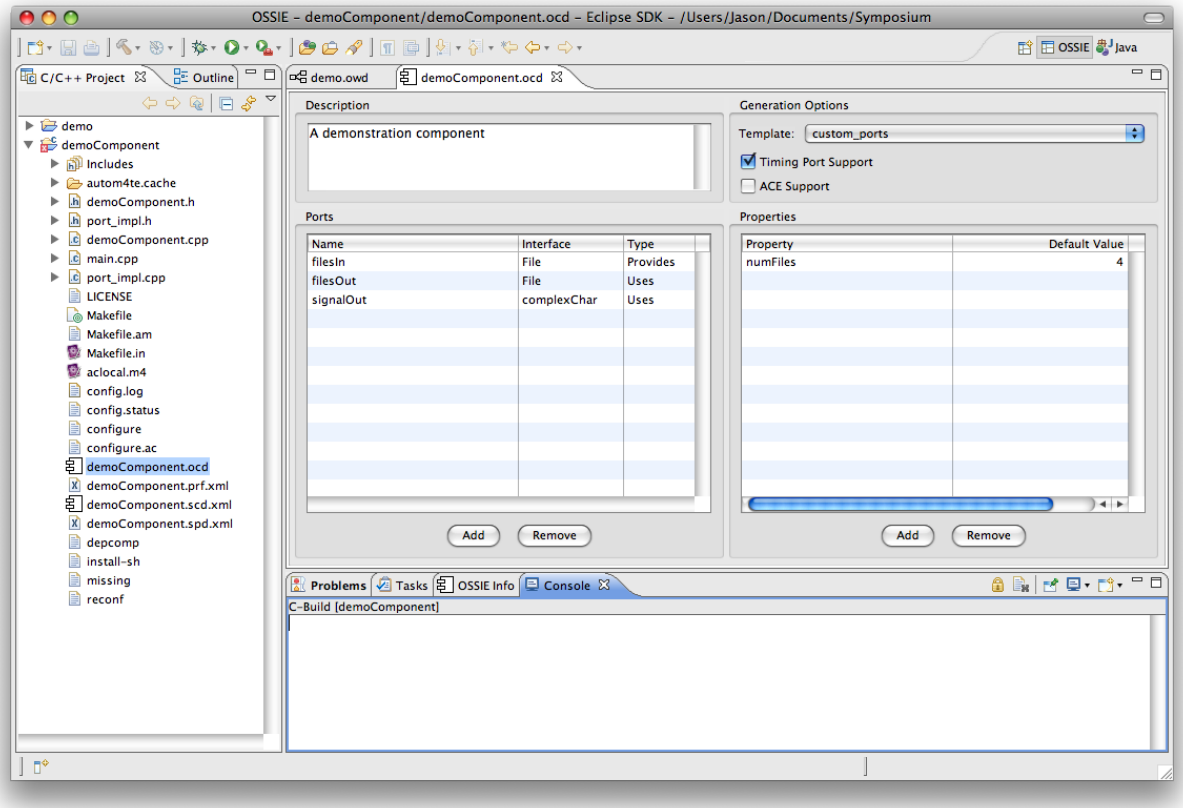

Figure 3: The Eclipse version of the OWD component editor.

Standard Widget Toolkit (SWT). SWT is the standard widget toolkit used by Eclipse plug-ins. We also rewrote much of the functionality for saving and loading waveforms and components.

The most important part of OWD, and the part we wished to leave in Python, was the code and XML generation. For each new component, OWD generates a skeleton implementation in either C++ or Python. This ensures the component will work correctly as a part of an OSSIE waveform. OWD also generates XML descriptors for components. Finally, OWD generates the necessary configure and make files to build and deploy the component. For waveforms, OWD generates the XML files that describe which components make up the waveforms, and how they connect with one another. The waveform can also specify properties for the components that are also described in the XML files. Like it does for components, OWD generates the necessary configure and make files for building and deploying the waveform.

We wished to leave this portion of OWD's functionality in Python for two reasons. First, it is the most complicated part of OWD and porting it to Java would have been difficult and would have taken longer than writing the infrastructure necessary to use the Python code within Java. Second, OWD was initially developed in 2006 by DePriest [5] and has been continually used and improved by the OSSIE group at Virginia Tech and is known to be very stable and reliable.

Finally, we adopted the notion of Eclipse projects. Previously, OWD could be used to create new components or to create new waveforms from existing components. The line between the two was somewhat blurred however, as new components could be made while making a new waveform. Our plug-in distinctly separates the two functions. Components and waveforms are now created as entirely separate projects within Eclipse, each contained entirely within its own directory.

5. CONCLUSIONS AND FUTURE WORK

Eclipse provides a powerful and extensible base for software development. The OSSIE plug-in described here, along with built-in and third party features, allow a developer to create, configure, edit, build, and deploy software-defined radio components and waveforms, all from within Eclipse. Future plans include creating a plug-in to integrate the functionality of the ALF tool within Eclipse.

Reusing the generation code and rewriting the interface allowed us to quickly build our Eclipse plug-in. It allows developers to quickly and easily build SCA-based components and waveforms as OWD did, and also provides several more improvements as a part of a full featured IDE. The first such improvement is project management. As stated previously, components and waveforms are now separate projects contained within directories. This makes the distribution of components and waveforms much easier. Using our plug-in simplifies the process of generating implementation code and XML. This had to be done manually with OWD. Our plug-in automatically does all of the necessary generation whenever a component or waveform is saved. The process of building and deploying components and waveforms has also been greatly simplified. After using OWD to generate all the necessary files, users previously had to use the command line to run several different tools. The user now has the option of building and deploying with a simple menu command, or he may opt to have the component or waveform built and deployed automatically every time it is saved.

Along with improvements gained by using our plug-in, several other plug-ins can be used to further aid developers. The C Development Tooling (CDT) plug-in provides code highlighting and debugging for C and C++ files. The PyDev plug-in provides code highlighting along with code completion for Python files. Finally, the SubClipse plug-in allows developers to store their work in a SVN repository. SVN is a version control tool that provides backups and versioning along with allowing multiple developers to work on the same project simultaneously.

10. REFERENCES

[1] James Neel, Carlos Aguayo, Jeff Reed, "Automated Waveform Partitioning and Optimization for SCA Waveforms," SDR Forum Technical Conference 2006, Orlando, FL, November, 2006.

[2] "OSSIE: SCA-Based, Open Source Software Defined Radio," <http://ossie.wireless.vt.edu/>.

[3] "Eclipse.org Home," <http://www.eclipse.org/>.

[4] Anthony Allowatt and Stephen H. Edwards. "IDE Support for Test-driven Development and Automated Grading in Both Java and C++," In *Proceedings of the 2005 OOPSLA Workshop on Eclipse Technology Exchange* (San Diego, California, October 16-17, 2005). ACM Press, New York, NY, pp. 100-104.

[5] Jacob A. DePriest, *A Practical Approach to Rapid Prototyping of SCA Waveforms*, M.S. Thesis, Virginia Tech, Blacksburg, VA, April 25, 2006.

ACKNOWLEDGMENT

This work was supported by U.S. ARMY CERDEC. Matthew Carrick, Shereef Sayed, and Philp Balister provided additional testing and feedback on the OSSIE Eclipse Feature.