P2.2 OVERVIEW OF THE NEW AWIPS SOA

John D. Lawson, Frank P. Griffith*
Raytheon Company

Jason P. Tuell, Ronla K. Henry
NOAA/National Weather Service

1. INTRODUCTION

The National Weather Service (NWS) has been an early adopter in the use of Open Source software in AWIPS by converting to the Linux Operating System and more recently moving from Informix to PostgreSQL. Open Source projects have proliferated and matured and are now viable alternatives for many applications.

Raytheon Company is currently developing a new AWIPS SOA that utilizes Open Source software extensively. The system will minimize the use of COTS and proprietary software.

This paper provides an overview of the specific Open Source components and how they are used in the new AWIPS SOA. The paper also discusses the benefits of using Open Source as well as associated risks and mitigation approaches.

2. WHY OPEN SOURCE?

A *fundamental* driver for the new architecture is the National Weather Service's desire to utilize *Open Source* software instead of COTS or proprietary software. With this approach, the NWS will realize significant savings on license fees and the administrative costs of negotiating and administering software licensing and distribution versus using commercial software (e.g., COTS). Moreover, the NWS will benefit from substantial code reuse and the ability to incorporate new Open Source software and enhancements as they become available. The extensive use of Open Source on this project has enabled a very rapid development cycle.

As demonstrated by Linux, Open Source software has become a viable alternative to COTS or in house developed software, In fact, open source can be found in commercial code and some open source code has commercial roots. For example, the Eclipse Rich Client Platform was developed by IBM. The NWS desire to utilize open source code makes Java the natural choice for the primary language of the new system due to the extensive amount of Java based open source offerings.

By utilizing Java-based Open Source software, NWS can achieve a significantly lower Total Cost of Ownership (TCO) and improved programmer productivity.

3. APPROACH

There are prerequisites, however, to effectively using open source extensively on a large scale project such as AWIPS, which currently weighs in at ~4.5 million SLOC. First, a framework must be designed for incorporating the code.[1] Second, the development team needs first hand experience with the Open Source Projects being considered and used. The Raytheon

* *Corresponding author address*: Frank P. Griffith, Raytheon Company; email: Frank_P_Griffith@raytheon.com

[1] See the companion paper, "Overview of the NEW AWIPS SOA", paper number 6.2 of these proceedings.

team has the experience with Object Oriented (OO) design, Java programming, Open Source Projects, and developing high performance Java based systems needed for success. Several best of breed open source projects have been integrated with a set of advanced enterprise patterns to create a highly extendable framework. So far, 15 major open source projects have been incorporated. The open source is used for the core infrastructure of the new AWIPS and is standards compliant. The open source libraries are controlled by putting them in the CM compile library and deploying them to the runtime environment. The open source codes are packaged together into the AWIPS Development Environment (ADE) which contains everything from the source code repository to the execution environment including operator clients. Note since the ADE includes the execution environment, applications are migrated *into* the environment. As a result, the system environment that exists at the end of the migration will include end-user applications *and* the development tools needed to develop new applications and/or extend the environment. This eventual system is referred to as AWIPS II. The 15 open source projects currently included in the ADE are shown in Figure 1 and represent approximately 1 million SLOC[2]. The amount of code developed to integrate these projects is an order of magnitude less.

The URLs for these projects are easily located using an internet search engine (e.g. "Google").

4. RISKS

Nothing is without risk and one must factor it into *any* approach (e.g. COTS, custom code). Typical questions that arise regarding the use of open source include: what if the open source is not robust enough or doesn't provide the required performance; what if you need a bug fixed; what if the open source project goes dormant; what if the open source project is superseded by a superior technology; what if it goes commercial? Let's discuss these.

"What if the open source is not robust enough or doesn't provide the required performance?" This is one reason for needing hands-on experience. Risk Reduction Demos are conducted to evaluate viability *early* in the life cycle.

"What if you need a bug fixed?" Support for open source projects is generally good, and is reported by some to be possibly better than commercial. This is another reason for needing first-hand experience with the project. Remember that you have the source code and in the worst case scenario can fix the bug yourself if it is critical and then submit for incorporation.

"What if the open source project goes dormant?" There are two answers here. First, this one very good reason to have a well designed framework and loosely coupled components. Since the details of each project are insulated and abstracted from the rest, it is relatively easy to replace it. In some cases the function is stable and the code is solid and so there may be no reason to replace it. And remember, you have the source code.

"What if the open source project is superseded by a superior technology?" See the previous paragraph on insulating components. We expect that we will replace some of these projects with new improved superior projects as time passes.

"What if it goes commercial?" If it is an open source project your options are to replace it, pay for it or … remember you have the source. It is unlikely that an open source project could collect fees for software already deployed. Since you have the source you aren't forced into immediate action and could replace it. However, if Java goes commercial that is a different story, we would most likely

[2] Although we all know SLOC isn't a good benchmark (similar to the MIPS in hardware being a Meaningless Indicator of Processor Speed) it does give an idea of the amount of code.

OS Project	Functionality	ADE Role
ANT	Build scripting	Build system
Mule + Spring	Enterprise Service Bus + Container	Run Environment
ActiveMQ	Java Messaging System Broker	JMS Broker
Lucene	Reverse Document Index	metadata index engine
Jibx	XML to Object Serialization	Canonical XML Message
GeoTools + JTS	GIS capabilites	GIS primitives
Tomcat	Web Server	Test Client Server
Jython	Python Scripting Engine	uEngine Python Script Engine
Baltik	Scalable Vector Graphics Tools	SVG tools
Common-digester	XML event driven parser	uEngine Script Parser
Ehcache	Event Driven Clusterable Cache	Cache Framework
Log4j	Java Logging API	Log manager
Jogl	Java API to OpenGL	CAVE rendering interface
Eclipse RCP	GUI plugin based framework	CAVE framework
Eclipse IDE	Java Integrated Development	Development environment
MC4J Console	JMX Management Console	Remote management console

Figure 1 – Open Source Projects Used in the AWIPS Development Environment

opt to pay for it and rationalize that we are better off than if we started with a commercial product.

5. SUMMARY

Open source has matured and proliferated which is testimony to its viability. Java has enabled the long standing OO promises of reuse. The savings to be had through reuse of open source software in developing systems is substantial both in terms of labor and time to system realization. The licensing fees and maintenance cost savings for a nationally distributed system such as AWIPS are *substantial*. The savings in administrative cost of managing the licenses Is larger than we want to admit. With open source projects brainpower from a wide area with expertise focused on a particular function is brought to bear, much more than your organization and more than a single commercial company. The rapid advancement of the internet is testimony to that. The brain power being leveraged is staggering.

To reap these benefits requires careful planning and solid engineering, and it is not without risk. However, we have decided that the promise is worth the risks and that the risks can be reasonably well mitigated. Apparently, we aren't alone. Open source usage in large scale projects seems to be growing.