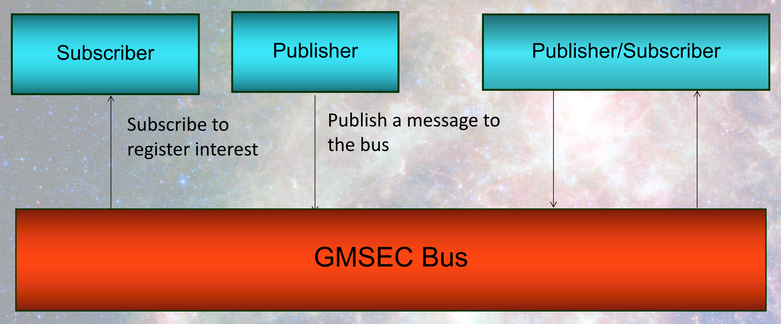
Upgrading the GMSEC Gradle Build Plugin

Nikhil Mittu[1], Faith Cheung[2], Rhea Mortam[3], Theresa Beech[3]
[1]University of Maryland, College Park, MD USA, [2] Richard Montgomery High School, Rockville, MD USA, [3]NASA Goddard Space Flight Center, Greenbelt, MD USA

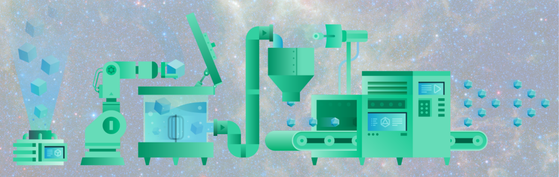
Background

The Goddard Mission Services Evolution Center (GMSEC) is a ground system architecture used broadly by many NASA and other US government space missions. It provides a scalable and extensible backbone for satellite mission operations systems which reduces ground system development and maintenance costs. There are a numerous GMSEC components that can also be selected based upon the mission requirements and easily changed as the mission evolves.



Gradle

- The GMSEC components are built using the Gradle build system.
- Most of the build tasks that the Components use are common across all of the GMSEC components so a custom Gradle plugin was created that contains all of these Gradle tasks.
- This Gradle plugin is then added as a buildscript plugin to the build.gradle files for each component.
- This means that the code for the tasks is just in one location and changes only have to be made in one location.

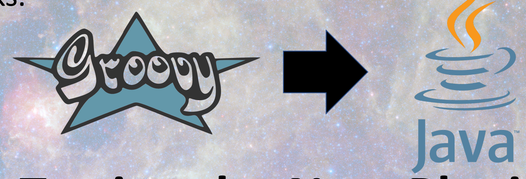


Learning Curve

Built Tool	OS Support	Language Support	Environment Discovery	Dependency Management
Autotools	NO	NO (java is painful)	YES	Only Checks, does not install.
CMake	YES	NO (java is painful)	YES	Only Checks, does not install.
Ant	YES	NO (c++ is painful)	NO	NO (Needs IVY)
Maven	YES	YES	NO	YES (for Java)
Gradle	YES	YES	YES	YES (for Java)

Upgrading the Gradle Plugin

When upgrading the GMSEC Gradle Plugin we moved from using Groovy as the programming language to Java. The move from Groovy to Java enhanced our ability to perform unit and functional testing of the plugin itself. Additionally the move to java will make the builds using the plugin faster. The Java version of the plugin also has increased maintainability. This is because all of the tasks and actions in the plugin have been separated out into their own classes. In addition to re-writing the plugin in Java we also made improvements and enhancements to the functionality of the plugin and its tasks.



Testing the New Plugin

Because the new plugin is written in Java we are able to test the plugin using the Mockito library to write unit tests for all of the actions in the new Gradle plugin. The new Gradle plugin also has a functional test. The functional test runs a Gradle build against a sample project to ensure the overall functionality of the plugin works, and to test all of the tasks.

Izpack 5 Upgrade

The GMSEC plugin uses Izpack to package the built components into an installer. Our work involved upgrading from Izpack 4 to Izpack 5. GSS, a separate GMSEC component, was already using Izpack 5 so now, with version 2 of the plugin, all GMSEC components will be using Izpack version 5. When upgrading to version 5 of Izpack the following things had to be done:

- Rewrite the Izpack task from scratch because the third-party plugin we were using previously, only supported Izpack 4.
- Update the Izpack configuration files in the sample project to work with Izpack 5.
- Document all of the changes that need to be made to the configuration files in the GMSEC developer's guide.

IzPack
Package once. Deploy everywhere.

Migrating the Robot Test Framework

The original GMSEC Gradle Build Plugin utilized acceptance test driven development (ATDD) through Robot Framework for the purpose of testing code. The process of upgrading the Gradle Plugin in turn, required the migration of the robot test framework to the new Gradle Plugin. Essentially, this meant converting the Groovy code of the original robot test framework to Java to correctly integrate it in the updated Plugin. Along with migrating the robot test framework, the rebot task also had to be migrated. The job of the rebot framework task is to format the results of the robot framework tests. Integrating the robot test framework into the new plugin required the following changes:

- Updating the rebot task framework to delete previously outputted logs of past tests before starting up.
- Managing input and output dependencies using Properties.
- Update the robot framework task to fall under the test category.
- Document all changes in the GMSEC developer's guide.

Conclusion

The new Gradle plugin is written in a modular way such that tasks and actions in the plugin are separated into their own classes. This makes the plugin more maintainable into the future. The fact that the plugin is written in Java means that GMSEC team members will not need to learn a new language. This will be very useful in the future if new members of the GMSEC team have to maintain the plugin. The new plugin also has unit tests and functional tests. This gives us an automated way to test the plugin before it is deployed to production to ensure there are no bugs.

Acknowledgments

We would like to thank our mentors Rhea Mortam and Theresa Beech for their guidance and help.