Using Component Technology to Facilitate External Software Reuse in Ground-based Planning Systems

Adam Chase - Jet Propulsion Laboratory, Caltech Institute of Technology

06/12/2003

Abstract

APGEN (Activity Plan GENerator - 314), a multi-mission planning tool, must interface with external software to best serve its users. APGEN's original method for incorporating external software, the User-Defined library mechanism, has been very successful in allowing APGEN users access to external software functionality. A few drawbacks to this method, however, have been identified: lack of library reuse and difficulty in library implementation, installation, configuration, portability and maintenance. A new method, the Adapter-Defined Component mechanism, is proposed to augment and someday even replace the User-Defined Library facility. The new method uses JCOM (JPL COM - Laverne Hall et al, 369) technology (a Unix-based implementation of Microsoft's Component Object Model) to create and dynamically load objects in a structured way. Because the component interfaces and implementations are carefully controlled, encapsulation and re-usability are promoted. The registration database can control which versions of which components are loaded at runtime. To allow JCOM components to be used from within APGEN, a SEQDispatch interface was created which allows APGEN to locate and invoke methods without access to component signature information. A command-line tool will use templates to generate all of the boilerplate necessary for creating a component that can be used from within APGEN, leaving only the implementation of the methods themselves to the component developers. This strategy should reduce the amount of work missions do by eliminating boilerplate, fostering reuse, and facilitating configuration management.

Contents

1 Introduction **2**
1.1 Mission Planning 2
1.2 APGEN 2
1.3 Challenges 2

2 Existing System **3**

3 ADef System **3**
3.1 JCOM 3
3.2 libADefVariant.so 3
3.3 SEQDispatch 4
3.4 adeftemplates 4
3.5 Adaptation 4

4 Conclusion and Future Work **4**

1 Introduction

1.1 Mission Planning

Planning spacecraft and rover missions is difficult. In the beginning of the process, the requirements are so loosely defined, it is hard to know where to begin. At the end of the process, there are so many constraints and variables that getting the most science out of your hardware can be daunting. Planning software must support an iterative process that takes our models of spacecraft and mission from something very abstract to something exceedingly detailed and provide answers to important questions along the way. The software must provide the planner with access to a varied array of tools and a means visualize and manipulate a plan easily.

1.2 APGEN

APGEN (Activity Plan Generator) is a GUI application from JPL's Sequence group that supports a version of user-driven, iterative planning. This is achieved through the modeling of scenarios on a timeline. An APGEN adapter defines the types of activities supported by her plan, how they will affect viewable state variables in her plan (resources), and any constraints that may exist on those activities and resources. The APGEN planner then assembles a scenario (plan) of activity instances and models it. As the spacecraft model becomes more complex, so does the APGEN plan and adaptation. Frequently, in order to make the model detailed enough, it is necessary to interface APGEN with highly specialized software (mmpat, cspice, mission specific). APGEN has been interfaced to AI-planners, databases, and mission-specific software.

1.3 Challenges

Mission planning software faces a number of challenges in the years ahead. There will be more missions and those missions will be more complex. There will be less time to spend adapting multi-mission software. There will be less time to do mission planning in general. Mission planning software must meet these schedule and complexity challenges while also providing planners with access to new planning tools and techniques.

2 Existing System

In order to give planners access to important features and functionality not provided by APGEN, APGEN adopted a User Defined Library. APGEN is linked to the User-Defined Library dynamically, and APGEN adapters can write their own User-Defined Libraries. A function to register functions that are callable from APGEN's adaptation language must be defined in the user-defined library. Adapters can link their User-Defined library to whatever C++ libraries they wish to use in their adaptation. This method has been used by a number of missions including MER, Cassini and Deep Impact. Though interaction with the external software is restricted to the base types supported by the APGEN adaptation language, there is no limit to the types of software that can be interfaced. There are a couple of drawbacks to this strategy. There is little reuse of User-Defined Libraries across missions or across Sequence tools because libraries become specific to APGEN and their mission. This prevents the adaptation that uses the functionality defined in those libraries from being reusable. Since there can only be one library, library writers cannot break the external software they want to use into modules. User-Defined Libraries must frequently be recompiled for each version of APGEN.

3 ADef System

3.1 JCOM

JCOM stands for JPL COM. It was developed by Laverne Hall et al in JPL's section 369. It is a C++ Unix-based implementation of a subset of Microsoft's Component Object Model. JCOM allows a user to instantiate a C++ object that implements an interface. The implementation of that interface is fully encapsulated. A registration database controls which implementations are instantiated at runtime. In JCOM, the components that are instantiated are always in the same process as the program that creates them. JCOMGen, a program designed to facilitate the creation of components, generates a number of boilerplate component source files and build files by applying the method signature data to a directory of WebMacro files.

3.2 libADefVariant.so

libADefVariant.so contains C++ classes that represent APGEN's fundamental data types. The classes were created with no dependencies on APGEN. The intent of this was that we could create components that acted on types defined in libADefVariant.so and then APGEN could create these types from its own internal types. Changes then to APGEN's core classes would not break APGEN's interoperability with JCOM Components.

3.3 SEQDispatch

Typical usage of a JCOM component would require that the user of the component have a header file corresponding to the interface that the component implements. In our case of components being called from our adaptation, the APGEN core doesn't know at compile time the interfaces that its adapter's components will have. The solution was to create a SEQDispatch interface that all components being used by APGEN must implement. There are a number of methods in the SEQDispatch interface, but the important one is Invoke. If we pass the name of the components method to Invoke along with an array of arguments, the Invoke interface will call the correct method.

3.4 adeftemplates

A set of JCOMGen templates for use with APGEN were created. The templates create a set of empty methods for each of the signatures specified in the file. They also implement the SEQDispatch interface in terms of the other interface. Using these templates, creating components that can be used from APGEN becomes a lot easier.

3.5 Adaptation

A standard function `ADef` was added to the suite of internally defined APGEN functions. ADef's first parameter is `''progID::methodName''`, where progID is a string that corresponds to a registered JCOM Component. The rest of the arguments are the arguments to the method in question. APGEN loads the component that corresponds to the progID, converts the internal data class arguments to the ones defined in libADefVariant.so, and calls 'invoke' on the component with the methodName. If the method succeeds, the returned value classes are translated back into APGEN's internal data types and the modeling continues.

4 Conclusion and Future Work

With this system, high-quality reusable software components can be easily made available to users of APGEN. Hopefully, this will give mission planners and adapters more time and tools to better meet their demanding schedules. User Defined Libraries that are in use for current missions are being examined and we are creating components that encapsulate that functionality. There is some desire to see if the libADefVariant.so types can be used with some of the other Sequence tools. There is also an effort to start capturing adaptation patterns for reuse in assembling new missions.

Acknowledgement

This work was performed at the Jet Propulsion Laboratory, California Institute of Technology, under a contract with the National Aeronautics and Space Administration.