REUSE OF THE JPL CFDP SOFTWARE IN THE JHU/APL MESSENGER GROUND SYSTEM

William C. Stratton
Constantine M. Frangos
Joseph J. Harrison
Douglas B. Holland

Ground Applications Group
Information Systems Branch
Space Department
The Johns Hopkins University Applied Physics Laboratory
Laurel, MD 20723-6099
William.Stratton@jhuapl.edu

ABSTRACT

JHU/APL is using CCSDS CFDP in support of MESSENGER, a NASA Discovery mission to study the planet Mercury. JHU/APL added CFDP capabilities to their common ground software assets by reusing core CFDP software components provided by NASA JPL. C++ classes were developed to encapsulate the JPL CFDP software, and a reusable, mission independent user program was implemented that conforms to JHU/APL's common GSE control and monitoring interface to the ISI EPOCH 2000 system. In developing the MESSENGER CFDP ground software, JHU/APL achieved an 87% software reuse level.

1 Introduction

CCSDS (Consultative Committee for Space Data Systems) has developed a standard protocol for transferring files to and from a spacecraft mass memory called CFDP (CCSDS File Delivery Protocol)[i]. Due to the unique requirements of space communications systems, CFDP is quite complicated; by definition, any complete implementation will be large and complex.

JHU/APL (The Johns Hopkins University Applied Physics Laboratory) is using CFDP in support of the MESSENGER (MErcury Surface, Space ENvironment, GEochemistry, and Ranging), a NASA Discovery mission to study the planet Mercury[ii]. CFDP transmits files of MESSENGER instrument and spacecraft housekeeping and science data from space to ground. The MESSENGER flight software has tight memory and processing constraints, and only the file downlink features of CFDP are needed, so a simple custom implementation of the required subset of CFDP has been developed. The MESSENGER ground software runs on powerful Sun workstations, so reuse of a general-purpose CFDP implementation is feasible on the ground.

Five such implementations were developed in the course of defining and validating the protocol[iii] (see Table 1-1). The development efforts were entirely independent, performed by different centers of different national space agencies for different platforms, but extensive testing has determined that they are interoperable. As a result, flight projects interested in using CFDP for mission operations are in an unusually advantageous position; the variety of available implementations increases the likelihood of being able to enjoy the advantages of this new technology in a given technical environment without the expense of developing it from the ground up.

Table 1-1 Current CFDP Implementations[iv]

Agency, Center	Language	Target Processor(s)	Target OS(s)
BNSC, Qinetiq	C	Force Sparc 3CE	VxWorks
ESA, ESTEC	Object Pascal, Delphi	Intel x86	Windows
NASA, GSFC	C	Intel x86	Linux
NASA, JPL	C	Sun Sparc PowerPC Intel x86	Solaris VxWorks Linux
NASDA, NEC	C++	Intel x86	Windows 2000

JHU/APL has a proprietary ground software architecture, common to all supported NASA space missions, including MESSENGER. The architecture follows the "pipes and filters" pattern[v]. A large body of software assets has accumulated that works with JHU/APL's architecture, and this software is reused by each new mission. The existing software assets are implemented in Solaris/C++ and are compatible with the ISI (Integral Systems, Inc.) EPOCH 2000 satellite ground system[vi].

The JPL (Jet Propulsion Laboratory) CFDP implementation also follows the "pipes and filters" pattern, and it can be built for Solaris[vii]. This makes it compatible with JHU/APL's common ground software architecture at a high level; JHU/APL licensed the software to implement the ground side of MESSENGER's CFDP requirements.

JPL CFDP is maturing rapidly. Currently it is being integrated into the Deep Impact Flight Software. JPL CFDP is also being integrated into JPL's AMMOS (Advanced Multi-Mission Operations System) and the Mission Data Systems Framework. MRO (Mars Reconnaissance Orbiter) and MSL (Mars Smart Lander) are considering using the JPL CFDP software in their software systems.[viii]

To integrate JPL's CFDP software into the existing JHU/APL common ground system, some "glue" software was developed. The MESSENGER CFDP transport layer consists of CCSDS telemetry transfer frames and command packets; hooks into the telemetry and command processing were provided. The JHU/APL software supports a generic GSE (Ground Support Equipment) interface, so a UNIX process was created for controlling and monitoring the CFDP software that follows the GSE protocol, making JPL CFDP look like a GSE to the rest of the JHU/APL system. C++ classes were created to simplify the development of a JPL CFDP "user program" for controlling and monitoring the CFDP software. These classes completely encapsulate the JPL software, isolating the JHU/APL software from JPL implementation specifics.

Much less effort was required to implement the "glue" than to develop a CFDP implementation from scratch; JHU/APL achieved an 87% reuse level, saving an estimated 6.8 developer-years of effort and allowing a working system to be deployed in a timely fashion for spacecraft I&T (Integration and Test). MESSENGER I&T is currently being supported by the integrated system described in this paper.

2 JHU/APL Ground Software Architecture

Figure 2-1 illustrates the JHU/APL ground software architecture, common to all of JHU/APL's NASA space missions, including MESSENGER. The common ground system is comprised of four functional areas: telemetry, commanding, planning and assessment. The commanding and telemetry functional areas work with the EPOCH 2000 system to provide control, monitoring and display capabilities for the spacecraft and for the GSE. The planning area provides offline commanding and spacecraft management planning. The assessment area is responsible for data archiving and production of data products used by the operations personnel for spacecraft assessment and by other data users.

Reuse of the command and telemetry systems requires maintaining standard interfaces to the spacecraft. The common ground system must also provide capabilities for control and monitoring of GSE. The system converts data from these external hardware systems into ground source telemetry data, which are handled by the same interfaces as the spacecraft telemetry. Likewise, the system uses portions of the existing command system to provide a standard method for GSE control.

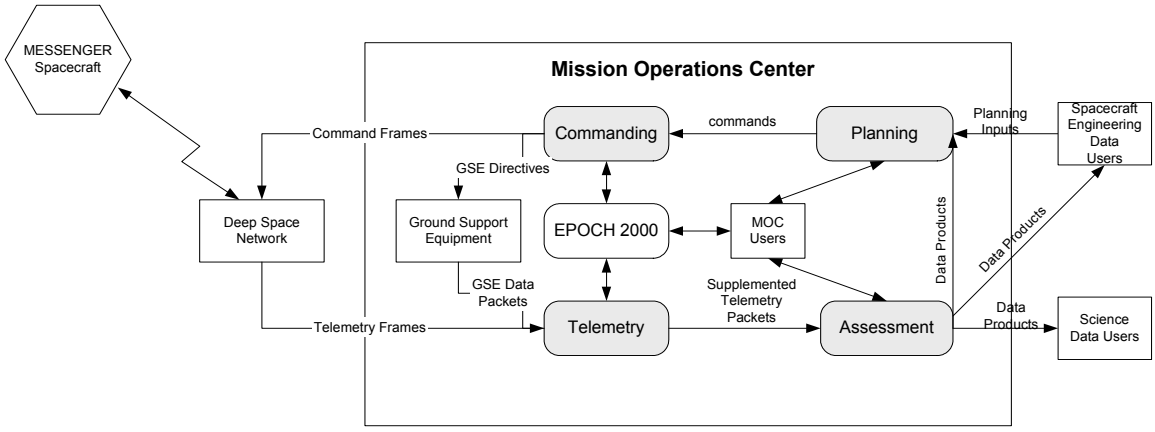

Figure 2-1 JHU/APL's common ground software architecture is used by all of JHU/APL's NASA missions.

The architecture is designed to be reused and extended as the mission moves through development stages. Initially the ground system supports flight software and subsystem development. The system is then used to support spacecraft I&T. The I&T configuration is the most complex, supporting a variety of GSE and spacecraft communication configurations. The system undergoes a reconfiguration to support flight operations. Adaptation of the command and telemetry interfaces is required for data coming from the Deep Space Network. This reuse across mission phases allows this system to develop and specialize as the mission progresses and allows major system architecture changes, such as the addition of CFDP, to be thoroughly tested before final deployment.

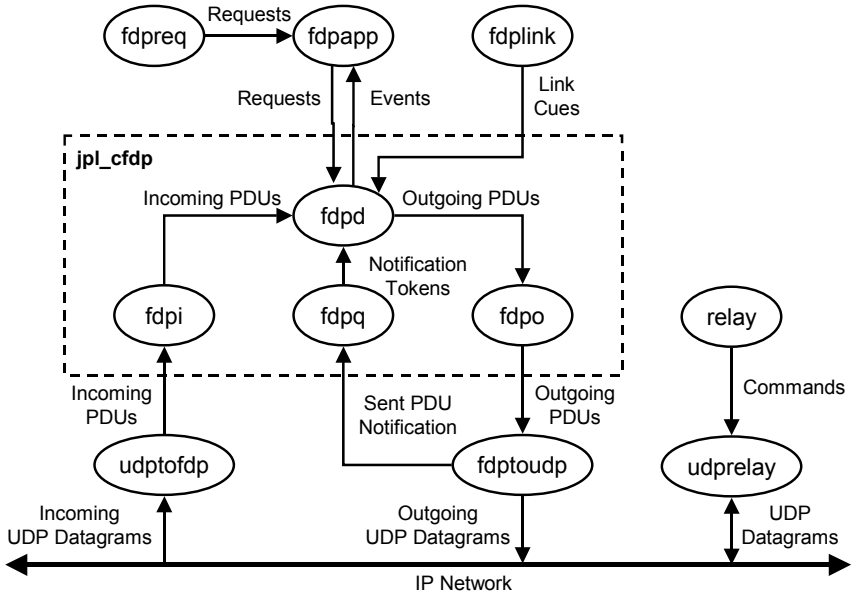

Figure 3-1 The core JPL CFDP processes (fdpd, fdpi, fdpq, fdpo) are reused in MESSENGER's ground system.

3 JPL CFDP Software Architecture

The core of the MESSENGER ground system CFDP implementation is provided by JPL. This C-language implementation, illustrated in Figure 3-1, is fully compliant with version 1 of the CFDP blue-book. JPL provides C libraries so that user applications can interface with the software. JPL also provides a fully functional UDP (User Datagram Protocol) implementation and a test harness so that the implementation can be verified. The test harness includes a UDP-based link simulator, which allows PDUs (Protocol Data Units) to be dropped, duplicated, delayed or reordered.

The MESSENGER ground system utilizes five of the JPL CFDP processes. These processes, which form the core of the JPL implementation, are *fdpd*, *fdpi*, *fdpo*, *fdpq* and *fdpsdrd*. The *fdpd* process is primarily responsible for implementing the CFDP protocol, including handling transmission and retransmission of PDUs. The *fdpi* and *fdpo* processes are responsible for receiving and sending PDUs from the Unitdata Transfer (UT) layer, respectively. The *fdpq* process notifies *fdpd* when an acknowledgement has been sent, so that *fdpd* can initiate a timer. The last process, *fdpsdrd*, manages the Simple Data Recorder (SDR). The SDR is used for persistent storage of data, providing the CFDP implementation with a consistent, abstract means of interacting with non-volatile memory on a variety of platforms.

The JPL implementation includes a number of processes that allow PDUs to be transmitted and received over UDP. A sample user program, *fdpapp*, allows the user to send requests from a command line program, *fdpreq*, to the CFDP daemon. The *fdplink* process is used to submit link cue commands to the CFDP daemon. The *udptofdp* process receives UDP datagrams, extracts PDUs from them, and passes them to *fdpi*. The *fdptoudp* process receives PDUs from *fdpo*, encapsulates them inside of UDP datagrams, and transmits them. Only one PDU is contained in each datagram. The *udprelay* and *relay* processes are the UDP link simulator and its command line interface, respectively.

The operation of CFDP is relatively straightforward. Incoming PDUs are received by *fdpi* via the UT layer and passed along to *fdpd*. *Fdpd*, in turn, will process the incoming PDU in accordance with the CFDP protocol. Whenever a PDU needs to be sent, *fdpd* will build the PDU and pass the data to *fdpo* for transmission to the UT layer. The user interacts with the JPL implementation through direct interaction with the *fdpd* daemon. Interactions, including requests for file transfers, link cues or changes to the configuration of CFDP, can be passed from the user application to *fdpd*. Whenever an asynchronous event occurs, such as the completion of a file transfer, *fdpd* will notify the user application through what is called an indication. From these indications, the user application can to perform any additional actions that may be necessary as a result of the event.

The JPL UT interface library, *libfdput*, is very easy to use. The three major functions this interface library performs are to allow user processes to initialize the interface,

forward PDU data received from other entities to CFDP and receive PDU data from CFDP for transmission to other entities. These functions allow the software developer to implement CFDP over a variety of transport layer protocols.

4 Integration of JPL CFDP

To integrate JPL's CFDP software into the existing JHU/APL ground system, software was developed to transport PDUs to and from the spacecraft, to control and monitor the ongoing CFDP processing and to process downlinked files as the transactions complete. Section 4.1 provides a description of the overall design of the new software.

As described above, the transport interface is simple and easy to use. However, the interface for controlling and monitoring the JPL CFDP software is not as well defined. The *fdpreq*, *fdpapp* and *fdplink* programs are provided as examples, but control and monitoring of the underlying SDR is not demonstrated. Fortunately, the JPL developers have provided timely support via email, and the test harness provided by JPL makes it possible to fully test a new user program. The use of the test harness in support of JHU/APL's new software development is described in Section 4.2.

The JPL CFDP software is delivered as a complete, self-contained system. This allows the system to be built and tested in a stand-alone configuration. For the operational MESSENGER ground system, JHU/APL is using the core functions of the JPL CFDP software, not the test harness. A good part of the integration effort turned out to be setting up the scripts and configuration files to launch just the selected core processes in the desired operational configuration for MESSENGER, as described in Section 4.3.

Software reuse was a primary design goal in the MESSENGER CFDP ground software development. A significant level of reuse was achieved, and most of the new software developed to integrate JPL's software into JHU/APL's ground system for MESSENGER is reusable in future JHU/APL missions. This is discussed in more detail in Section 4.4.

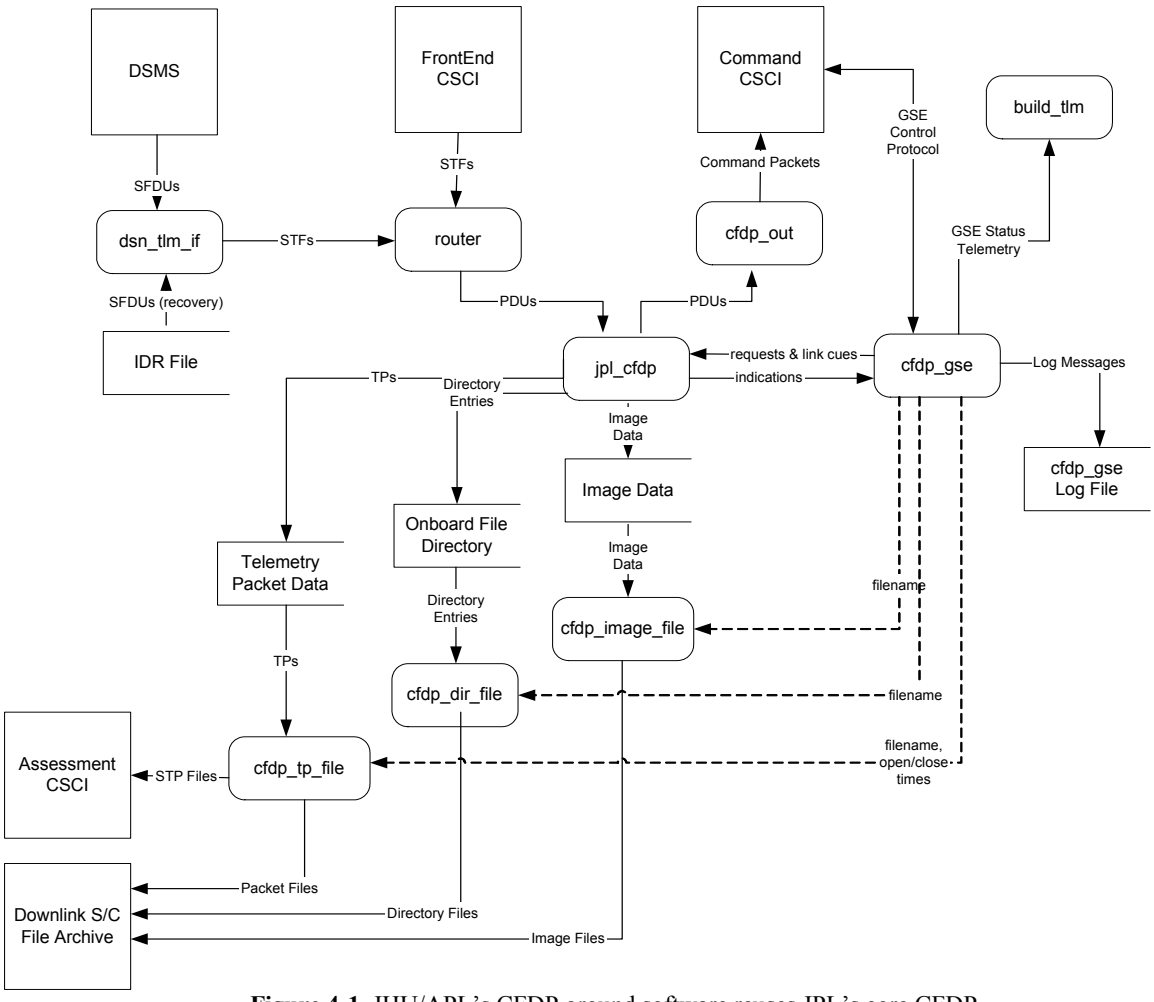

Figure 4-1 JHU/APL's CFDP ground software reuses JPL's core CFDP processes and JHU/APL's common ground software assets.

4.1 CFDP Ground Software Design

Figure 4-1 shows the CFDP ground software processing. The *router* and *cfdp_out* processes provide PDU downlink and uplink. The *cfdp_gse* process controls and monitors the JPL CFDP software. MESSENGER files are processed as soon as downlink completes: *cfdp_image_file* handles image files, *cfdp_dir_file* handles on-board file system directory files and *cfdp_tp_file* handles telemetry packet files.

4.1.1 PDU Transport

The CFDP protocol leaves the choice of transport layer up to the mission. In MESSENGER, PDUs are uplinked in CCSDS command packets and downlinked in CCSDS telemetry transfer frames.

The *cfdp_out* process encapsulates the handling of uplink PDUs. The uplink PDUs are obtained from the CFDP software using *libfdput*. For each PDU received from the JPL CFDP software, *cfdp_out* builds a MESSENGER command packet and sends it to the JHU/APL Ground System Command CSCI (Computer Software Configuration Item) for uplink. The Command CSCI builds and transmits command frames, interleaving *cfdp_out*'s packets with real-time packets generated by EPOCH.

The existing *router* process performs CCSDS telemetry packet reconstruction from frame data fields; it was a simple matter to extend *router* to handle downlink PDU reconstruction as well. MESSENGER telemetry VC6 (Virtual Channel 6) frames are dedicated to transporting PDUs. The PDUs are not aligned in the transfer frame data areas, so the *router* software reconstructs PDUs from data segments spanning frames. As PDUs complete, they are submitted to the JPL CFDP software through calls to JPL's *libfdput*.

4.1.2 Control and Monitoring

JPL's CFDP implementation assumes the presence of a "user program" that issues directives and handles indications for CFDP control and monitoring purposes. JPL provides *fdpapp* as an example but fully expects missions to develop their own user programs specific to their needs. JHU/APL developed the *cfdp_gse* UNIX/C++ process to play the role of user program for the MESSENGER ground system.

The JHU/APL ground system provides remote control and monitoring capabilities for any GSE. The typical use of this capability is for integration of external ground hardware systems, but it can also be used to control and monitor internal software systems. JPL's CFDP runs continuously and autonomously; from the ground system's point of view, the CFDP software is just another GSE. The *cfdp_gse* process wraps JPL's CFDP software, implementing the JHU/APL-specific GSE communications protocol and making JPL CFDP look like a GSE to the rest of the JHU/APL ground system. Figure 4-2 shows the static class structure of the *cfdp_gse* software.

The *Fdpd* class wraps the JPL CFDP control and monitoring software, adding many useful user program features:

- Application-level control directives
- Link-level control directives
- Event transmission and logging
- Periodic status of all active transactions
- Ability to cancel all active transactions
- File post-processing invocation

By wrapping JPL CFDP in a class, namespace collisions resulting from global names and C-preprocessor definitions are eliminated.

The *CfdpGseCmdClient* class implements the GSE control protocol. The parent class, *GseCmdClient*, supports the TCP/IP socket control communications required of all GSEs, while the *CfdpGseCmdClient* child class encapsulates the controls specific to JPL's CFDP software. The child's main job is to parse and validate the EPOCH STOL (Spacecraft Test and Operations Language) directives destined for JPL's CFDP.

The *CfdpGseTlmClient* class implements the GSE monitoring protocol. The *GseTlmClient* parent class provides the TCP/IP socket monitoring communications required of all GSEs. The *CfdpGseTlmClient* child class encapsulates the monitoring specific to JPL's CFDP software. In particular, the child constructs periodic

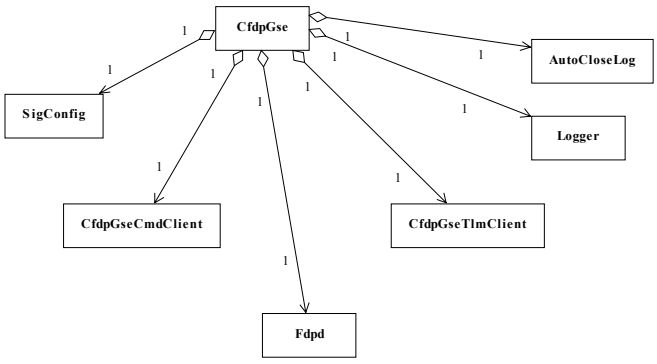

Figure 4-2 The cfdp_gse user program is layered on classes that encapsulate the JPL CFDP software and the JHU/APL GSE control and monitoring protocol.

cfdp_gse and active transaction status messages for transmission to the ground system.

The *cfdp_gse* process configuration is specified in XML (eXtensible Markup Language). SigConfig parses the configuration file using Apache Xerces C++. The configuration file syntax is validated using an XML Schema.

Events are logged to an ASCII file. The *Logger* class provides standard file naming and event message formatting for multi-threaded applications. To prevent log files from growing indefinitely, the *AutoCloseLog* class periodically closes out the current log file and opens a new one.

4.1.3 Downlink File Processing

The MESSENGER ground system must perform post-processing of three different types of files downlinked via CFDP: files of telemetry packets, files containing directory information regarding the spacecraft's file system, and critical image files. The processing of these files is handled by *cfdp_tp_file*, *cfdp_dir_file* and *cfdp_image_file*, respectively. The file types are distinguished by special naming conventions. Post-processing begins after an indication is received from the JPL software, notifying *cfdp_gse* that a file was downlinked. *Cfdp_gse* will initiate the post-processing of the file. For files of telemetry packets, additional headers are added so that other ground software tools can process them at a later time. Directory files are simply formatted and displayed in a window on the screen. Image files that are needed immediately for other processing in the MOC are decompressed. After the files are processed, they are distributed to the appropriate location.

4.2 JHU/APL User Program Testing

The *cfdp_gse* process is driven by STOL directives, CFDP indications and periodic timers. This makes the availability of test drivers essential for stand-alone testing. Two simple Perl scripts, *cmd_if_sim.pl* and *build_tlm_sim.pl*, were written to play the role of the MOC (Mission Operations Center), but no driver development was required to simulate the MESSENGER CFDP system; the JPL CFDP test harness was reused for that purpose. The test harness requires two UNIX workstations, one to play the role of spacecraft, and one to play the role of ground system. Figure 4-3 illustrates the complete *cfdp_gse* test configuration.

JPL provides an extensive set of test scripts, simulating a variety of CFDP usage scenarios. The only change required for stand-alone *cfdp_gse* testing was a small modification to the start-up script allowing the tester to select *cfdp_gse* rather than *fdpapp*.

4.3 JPL CFDP Software Configuration

To integrate the core JPL CFDP processes into the MESSENGER ground system, scripts and configuration files had to be developed. The JPL test harness provided source-level resources for this effort, but the scripts and configuration files had to be modified for the operational environment.

A script was written to generate the operational configuration files; it calls JPL's *fdp_config* script with the parameters appropriate to MESSENGER. JPL's *fdp_start* script was modified to run just the core JPL CFDP processes without the test scaffolding: *fdp_sdrdefine*, *fdpd* (which runs *fdpo*), *fdpi* and *fdpq*. The *cfdpconfig.pvl* file was tailored to select the

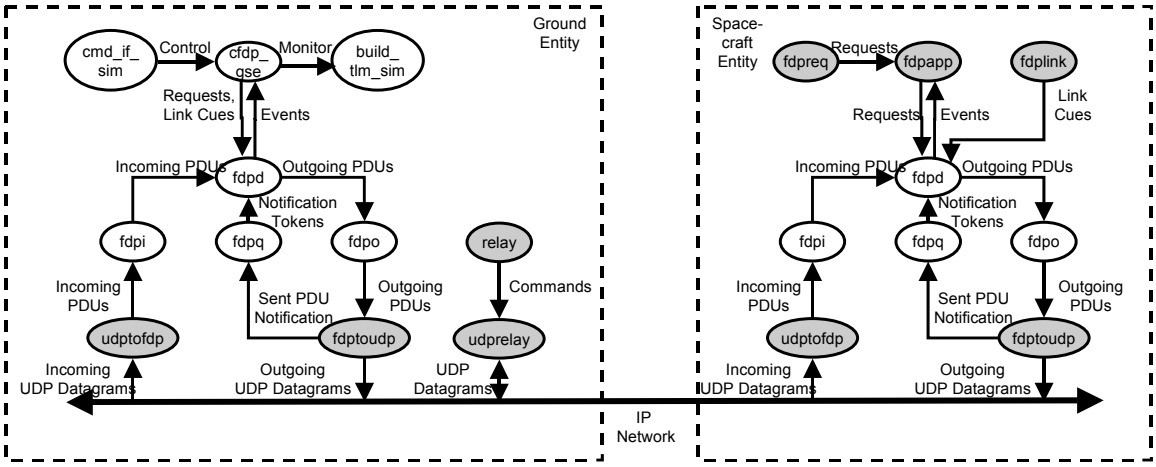

Figure 4-3 The JPL CFDP test harness provided a useful environment for developing JHU/APL's cfdp_gse user program (test harness processes are shaded in the diagram).

MESSENGER CFDP options.

Configuration of the SDR is not obvious. These parameters have to be tuned based on usage patterns. JHU/APL added statistics to the periodic status information produced by *cfdp_gse* to get visibility into SDR usage, but SDR tuning will be an on-going concern.

4.4 Software Reuse

Ivar Jacobson defines reuse level as "the ratio of size of workproducts derived from reusable component systems to total application size"[ix]. For the MESSENGER CFDP ground software implementation, the JPL CFDP is the reusable component system; the C-language portion of JPL CFDP is 39,030 SLOCs (Source Lines of Code) according to linecnt.pl[x]. Table 4-1 shows that 5,765 new C++ SLOCs were developed to integrate JPL CFDP. So the reuse level is R = 39,030 / (39,030 + 5,765) = 87%.

Table 4-1 SLOCs written to integrate JPL CFDP

Process	Total new SLOCs	% reusable in future missions
router	315	61%
cfdp_out	1,007	100%
cfdp_gse	4,443	98%
Total:	5,765	96%

To estimate how much effort was saved, assume that JHU/APL had to develop the JPL CFDP from scratch, and assume that the development rate for the JPL CFDP is the same as the development rate for the new C++ code written to integrate JPL CFDP. It took about one developer-year to write the new C++ code, so JHU/APL saved = 39,030 x (1 / 5,765) = 6.8 developer-years.

The new C++ code was developed by JHU/APL to be a reusable component in future systems. We estimate only 4% of the new code is MESSENGER-specific; 96% of the new code is reusable in future JHU/APL missions requiring CFDP support.

5 Conclusion

The reuse of JPL's CFDP software implementation in the JHU/APL MESSENGER Ground System has worked well. The JPL CFDP software has saved significant development effort on MESSENGER, allowing a complete CFDP ground system to be deployed in time for early spacecraft I&T activities.

The integration has not been totally seamless. To meet the MESSENGER Ground System requirements, the capability of reporting the status of all active transactions had to be added to the JPL CFDP software. MESSENGER was not the only user to request this enhancement, so while we had to wait for the feature to be implemented, we received a robust product, reviewed and tested by multiple missions.

Some change requests made by APL were not accepted for implementation in the JPL CFDP software because they were not considered to be of general utility; in those cases, APL's cfdp_gse user program was extended to provide the required functionality.

Additional enhancements to the JPL CFDP software would help future integration efforts go more smoothly:

- Encapsulation of the user program interface.
- Clarification of the SDR control, monitoring, and tuning.

These enhancement requests have been submitted to the JPL CFDP software developers and may become available in a future release.

Integration work remains to be done at JHU/APL for MESSENGER. The JPL CFDP software continues to evolve rapidly; in six months, the system grew by 15% (5,125 SLOCs), and the official release of the version we are using won't be available for another two months. JHU/APL will need to integrate the next official release for the system to be supportable. Finally, the implementation of the file-based, persistent SDR is just becoming available and remains to be integration tested. This feature will become increasingly important to MESSENGER as it progresses through the later stages of development in preparation for launch.

Overall, the reuse of JPL CFDP in the JHU/APL MESSENGER ground system has been successful. The CFDP ground software system was delivered on schedule for MESSENGER I&T. Reuse of the JPL CFDP software has kept costs low. Furthermore, the approach has resulted in a reusable JHU/APL CFDP ground system, providing CFDP capabilities to other missions if desired, with little additional software development effort.

ACKNOWLEDGEMENTS

The authors would like to acknowledge the extensive technical support and encouragement provided by Scott C. Burleigh and Kathy B. Rundstrom at JPL. The JHU/APL CFDP ground software implementation described in this paper would not have been possible without their rapid responses to our questions and their timely software deliveries.

REFERENCES

[i] *CCSDS File Delivery Protocol (CFDP).* Recommendation for Space Data System Standards, CCSDS 727.0-B-1. Blue Book. Issue 1. Washington, D.C.: CCSDS, January 2002. http://www.ccsds.org/documents/727x0b1.pdf

[ii] C. J. Krupiarz, S. C. Burleigh, C. M. Frangos, D. B. Holland, K. M. Lyons, W. C. Stratton, "Use of CCSDS File Delivery Protocol on MESSENGER": Space Operations 2002 Conference. http://www.ccsds.org/documents/so2002/spaceops02_p_t5_35.pdf

[iii] *CCSDS File Delivery Protocol (CFDP).* Report Concerning Space Data System Standards, CCSDS 720.1-G-1 and CCSDS 720.2-G-1. Green Book. Issue 1. Washington, D.C.: CCSDS, January 2002. http://www.ccsds.org/documents/720x1g1.pdf http://www.ccsds.org/documents/720x2g1.pdf

[iv] Table developed by Scott C. Burleigh, JPL.

[v] F. Buschmann, R. Meunier, H. Rohnert, P. Sommerlad, and M. Stal, *Pattern-Oriented Software Architecture; A System of Patterns, Volume 1.* John Wiley & Sons Ltd., 1996. ISBN 0-471-95869-7.

[vi] http://www.integ.com/

[vii] S. C. Burleigh, "Operating CFDP in the Interplanetary Internet": Space Operations 2002 Conference. http://www.ccsds.org/documents/so2002/spaceops02_p_t5_22.pdf

[viii] Current JPL CFDP usage information provided by Kathy B. Rundstrom, JPL.

[ix] I. Jacobson, M. Griss, and P. Jonsson, *Software Reuse; Architecture, Process and Organization for Business Success*. ACM Press, 1997. ISBN 0-201-92476-5.

[x] Source Lines of Code estimates per Reasoning Systems, Inc. linecnt.pl Version 1.5 (http://www.reasoning.com/downloads/c_line_count_estimator.html).