REDUCING THE COST OF GROUND SYSTEM DEVELOPMENT AND MISSION OPERATIONS USING AUTOMATED XML TECHNOLOGIES

Colette Wilklow
Jet Propulsion Laboratory,
California Institute of Technology
4800 Oak Grove Drive,
MS 301-240, Pasadena, CA
91109-8099
phone + 1 818 354-4674
fax + 1 818 393-4100
email: colette.wilklow@jpl.nasa.gov

Jesse Wright
Jet Propulsion Laboratory,
California Institute of Technology
4800 Oak Grove Drive,
MS 264-214, Pasadena, CA
91109-8099
phone + 1 818 393-7971
fax + 1 818 393-5247
email: jesse.wright@jpl.nasa.gov

David Noble
Jet Propulsion Laboratory,
California Institute of Technology
4800 Oak Grove Drive,
MS 264-214, Pasadena, CA
91109-8099
phone + 1 818 393-2917
fax + 1 818 393-5247
email: david.noble@jpl.nasa.gov

Kathryn Sturdevant
Jet Propulsion Laboratory,
California Institute of Technology
4800 Oak Grove Drive,
MS 301-240, Pasadena, CA
91109-8099
phone + 1 818 354-1147
fax + 1 818 393-4100
email: kathryn.sturdevant@jpl.nasa.gov

Joseph Snyder
Jet Propulsion Laboratory,
California Institute of Technology
4800 Oak Grove Drive,
MS 264-522, Pasadena, CA
91109-8099
phone + 1 818 393-2341
fax + 1 818 393-2347
email: joseph.snyder@jpl.nasa.gov

ABSTRACT

Limited time and resources continue to challenge today's ground system development and mission operations personnel. As the missions we develop become increasingly complex and sophisticated, we are faced with the demands of reduced schedules and shrinking budgets. Additionally, we must meet the need to share knowledge, tools and personnel across a variety of projects. These factors require a system that is robust enough to handle these constraints, and elegant enough to ensure mission success.

To meet this challenge, JPL software developers have implemented a new system, which provides an efficient common interface between flight and ground software systems. This system is based in the application of automated XML (eXtensible Markup Language) technologies. In the past, flight to ground software interfaces have been tediously defined such that changes in flight software definitions could take two weeks or more to propagate into the ground system environment. Earlier missions with longer development schedules were somewhat equipped to handle delays in the flight to ground process, however as mission development resources and personnel continued to decrease, it became clear that this could no longer be an effective way of doing business. The demands of today's schedules and budgets require that flight and ground system development occur in parallel, thus the need for a well-structured interface.

This new system has reduced ground system turnaround time from two weeks or more to approximately four hours with minimal staffing. Command and telemetry definitions are described using XML. The ground system interfaces to these XML definitions generated by the flight software developers. This enables the ground software developers to create code that does not require modification for new flight software definitions. Telemetry definitions include sensor data (called channels for historical reasons), event messages (printed messages from flight software), and flight software data products or files. Spacecraft data products can be assembled from CCSDS (Consultative Committee for Space Data Systems) packets and converted to text and XML representations for use by a variety of tools, including but not limited to those written in Java, Perl and Python. Operations personnel will benefit from the automated generation of downlink reports provided by this system. XML related technologies, such as Extensible Stylesheet Language (XSL), have further extended the sophistication of the system via data

transformations to desirable formats such as HTML and PDF. This paper discusses our experiences using this XML intensive system and the improvements and tradeoffs of this emerging technology.

INTRODUCTION

The demands of today's compressed mission development schedules dictate that the development of spacecraft flight and ground systems occur in parallel. This introduces a significant challenge to ground software developers in that the ground system must be adaptable to design changes in the flight system as they occur. Also, ground system development frequently begins before operations personnel are brought onto a project, thus it is desirable to design a configurable ground system that will be flexible enough to meet future needs of mission operators.

In order to meet these challenges for the Mars Exploration Rovers (MER) project, we identified the need for well-structured interfaces between the flight, ground and mission operations systems. If such interfaces were available, updates to the ground system could be automated. Additionally, a well-defined interface to ground system tools would enable mission operations personnel to reconfigure the system to meet their needs without being required to possess advanced programming skills. Due to its rising acceptance as a robust and well-structured way to describe data, XML was a logical choice for defining these interfaces

BACKGROUND

XML is a meta-markup language with a look and feel similar to the familiar HyperText Markup Language (HTML). Like HTML, data is expressed within a series of tags (e.g. <html></html>). Tags used in HTML are pre-defined. Someone writing an HTML file would consult the latest standard to find out how to describe his or her HTML file. XML is referred to as a "meta" markup language because unlike HTML, the definition of allowable tags is left to the user. The definition of what is allowed in a specific XML document is referred to as the document schema.

Unlike HTML, XML is used only to describe data and does not in itself allow for descriptions of how data will be formatted. XSL transformations are frequently used to define the output formats. Additionally output formats can be established via XML processors written in Java, Perl, Python or any other language capable of processing text.

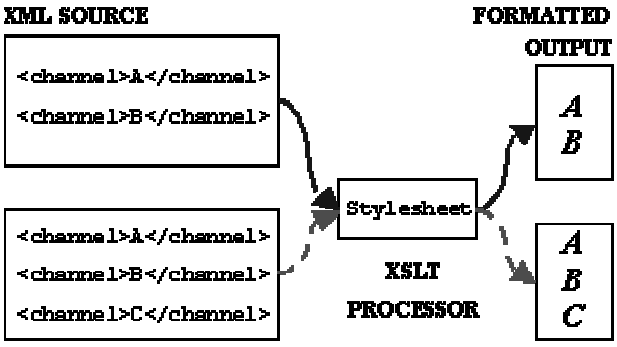

Figure 1: A single XML source file can be processed in a wide variety of ways resulting in the ability to create many different outputs with consistent content

DICTIONARY MANAGEMENT

Modification to the flight system command and telemetry dictionaries must be propagated through the ground system. In the past, changes to these dictionaries were costly, taking up to two weeks for a change to be reflected in affected telemetry and command processing and documentation because each change would require personnel to manually update affected software and documentation. On MER, we implemented a dictionary management system with the purpose of streamlining this process.

The dictionary management system provides software tools that allow widespread deployment of the MER command and telemetry dictionaries in a variety of formats. The flight software team provides an XML version of both dictionaries, as well as XML definitions for flight software data products and event reports.

For each XML delivery, the ground software development team generates the configuration files that are used by the ground software to process the telemetry and generate commands. Automation scripts are used to accelerate this process. Once built, the output products from this process are deployed to testbed and mission support areas as appropriate.

HTML and PDF formats of the dictionaries are also created and deployed. Running the XML file against a series of XSL stylesheets using an XSLT processor creates the HTML and PDF formats of the dictionaries.

Each command and telemetry XML file contains change log entries made by those who directly modify the command and telemetry XML files – these logs are captured and transformed into HTML

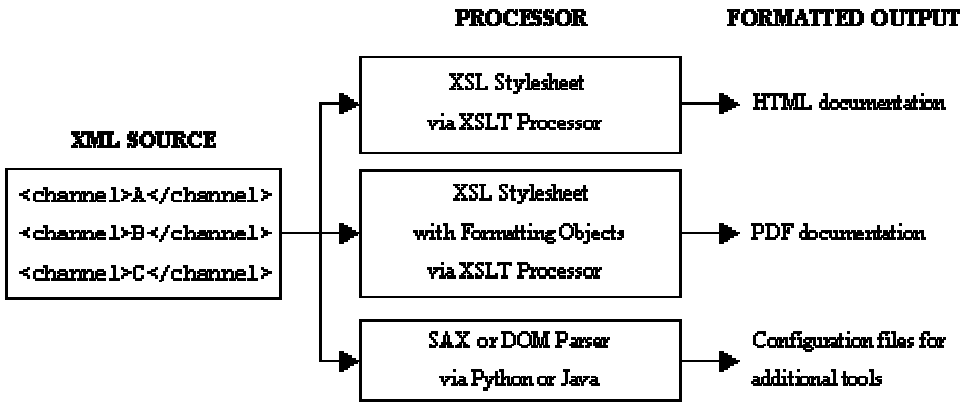

Figure 2: As long as the processing tools adhere to the XML document's schema, a change in content will not require a change to the processing tool

and PDF formats. Additionally, there exists a set of Java classes that capture changes automatically, looking for whether or not a command or channel has been added, deleted or was subject to selected changes.

The primary benefit of describing the data via XML is that processing a single source file can create all of the desired output formats. (Figure 1) For example the file may be processed using XSL Transformations, or by software using SAX (Simple API for XML) or DOM Document Object Model parsers for XML (where SAX is an event based parser allowing rapid processing and DOM is a tree based parser which loads the XML document into memory, allowing for random access of document elements). This reduces the complexity and errors that might be introduced because all changes only need to occur in one place. For example, one source XML file contains all of the telemetry channel definitions. This file is processed by various tools to produce documentation in HTML and PDF formats, as well as for producing configuration files for telemetry monitor and display. Since each of these tools can process the standard format of the source XML file, the addition, modification or deletion of a new telemetry definition will not require a software change in the processing tools (Figure 2).

During development, changes to mission command and telemetry dictionaries occur frequently. In the last year, the GDS team for MER has processed over one hundred new command and telemetry dictionary deployments. Fortunately, very few of these deployments have required a change to the GDS software used to process these dictionaries. The few times that code changes were needed were because of changes to the dictionary's schema as the definitions evolved.

The most notable result of the dictionary management system has been the significant reduction in time to propagate a flight software change through the ground system. Based on our experiences on the MER project, we estimate that on average it takes approximately one full time employee about four hours to complete this process per dictionary from the time that the XML source file is delivered from flight software. This is a significant reduction from the estimated two weeks or more this process took on other projects.

REPORT GENERATION

In developing ground system software, it is critical that the data acquired from the spacecraft be presented to the operations team as clearly, accurately, and concisely as possible. Mission engineers and scientists must be able to evaluate the data quickly to plan for the next uplink opportunity.

Early in mission ground software development it can be difficult to identify how best to express spacecraft data to operations personnel. There are many variables that may impact what data might be important to various operators. The software must be highly adaptable to changes and additions to the needs and desires of the operations team.

We facilitate this highly dynamic development environment through the MER report generator (Figure 3). MER Report Generation refers to a suite of XML definitions, Java and Python tools, and XSL transformations. The report generator can process

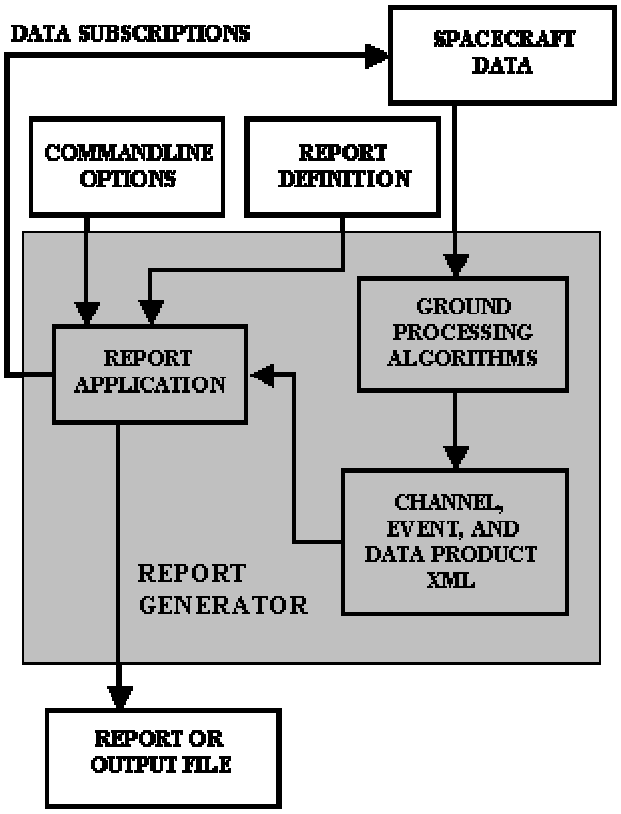

Figure 3: MER Report Generator

telemetry channel data, event reports and flight software data products and output this information in various formats including ASCII text, Encapsulated Comma-Separated Value (ECSV), Extensible Markup Language (XML), HyperText Markup Language (HTML), and Portable Document Format (PDF). This suite is used to process mission data for the purpose of automatically populating outputs that would otherwise be hand generated by operations personnel. These outputs are to be used by the spacecraft team to aide in the generation of the spacecraft downlink report, to use as input to various software tools, and to provide relevant mission information to operations personnel.

This system is highly configurable so it may be used to create a wide variety of reports. For each report type, there exists a definition describing the desired content (Figure 4). This definition file is expressed via XML where the data describes which telemetry, products and event reports to which the report generator should subscribe.

These subscriptions are designated with one of several capabilities that describe how the data should be collected. Currently we have capabilities for

```
<?xml version="1.0"?>
<definition version="1.0">
    <report title="Downlink Report">
        <section title="Downlink Summary">
            <capability type="LatestCh" title="Latest Channel Summaries">
                <xml_out_file>LatestChannels.xml</xml_out_file>
                <style>
                    <stylesheet>all_channels.xsl</stylesheet>
                    <formatted_out title="All Channels By sc_time">
                        all_channels_by_sc_time.html
                    </formatted_out>
                </style>
                <properties>
                    <property name="ch">1</property>
                    <property name="ch">2</property>
                    <property name="ch">3</property>
                    <property name="ch">4</property>
                    <property name="ch">5</property>
                </properties>
            </capability>
            <capability type="Event" title="Event Summary">
                <xml_out_file>AllEvents.xml</xml_out_file>
                <style>
                    <stylesheet>allevents.xsl</stylesheet>
                    <formatted_out title="All Events by Spacecraft Time">
                        all_events_by_sc_time.html
                    </formatted_out>
                </style>
                <properties>
                    <property name="orderedBy">sc_time</property>
                    <property name="id">all</property>
                </properties>
            </capability>
        </section>
    </report>
</definition>
```

Figure 4: An Example Report Definition Described in XML

subscribing to packet application identifier counts, event reports, latest telemetry channels, telemetry channels on change, all telemetry channels, new product availability and values to be plotted by an external plotting tool.

The subscription results in an output XML file which contains the collected data with associated ground processing algorithms having been applied. For example, the XML output may contain raw data values as well as ground-derived values. Each capability may also have associated properties. For example telemetry capabilities have properties for defining subsets of telemetry channels and which algorithms to apply to those channels. Event report capabilities have properties allowing for the output to be sorted in various ways.

In addition to defining subscriptions and their associated capabilities, the definition file allows specification of processing tools and associated stylesheets that should be applied to the XML output files containing the collected data. New reports can be created by producing a new definition file containing the desired capabilities and properties for that report.

At the heart of the report generator is the report application. The report application accepts command line parameters that define which report type is to be generated, which data set is to be reported on, the data source and other options for specifying spacecraft specific items (on MER we have two rovers) and output directories. The report application also parses the definition file and kicks off the subscriptions for the designated report type. Once the capability output XML files are generated, the report application runs the appropriate processing tools on those files and writes the outputs to the specified directory.

Once the reports are generated, they are made available for ingestion to other tools or for evaluation and annotation by the mission operations team for use in planning the next uplink session.

LESSONS LEARNED

We have found through our experiences using this XML intensive system, that it is worthwhile to spend time upfront carefully defining the document schema. Spending time defining the data to be expressed will save time in later development.

Also, it's important to consider using multiple XML files if some of the data you wish to express is orthogonal to the rest of the data. Attempting to define a schema with data that doesn't fit together nicely can make it cumbersome for others to compose and process XML files using the schema. Generally, XML is only as useful as the data it describes.

Lastly, it's important to have some understanding of the pros and cons of using a SAX or DOM parser for processing XML files. We began our telemetry channel processing for documentation using a DOM based parser. However, our source XML file eventually evolved into a >100,000 tag document. Processing a file this large with a tree parser can be extremely slow. We found that there are at least two alternatives that can speed up the processing. The first is to split the source XML file into several XML files and process the smaller files. The second is to write a SAX (event based) parser that does not store the entire document in memory. The main thing to keep in mind is that there are multiple ways to process XML files and its worthwhile to spend some time learning to understand the pros and cons of each.

FUTURE WORK

The described command and telemetry dictionary and report generation processing tools are currently mission specific to MER. We are in the process of converting these tools to become a standard toolkit for new missions.

Another area that we are investigating is in automatic generation of XSL stylesheets and report definitions. This would further enhance the speed and reliability of our tool set.

CONCLUSION

XML works! Using automated XML technologies, we have developed a system that provides a well-defined interface between the flight, ground and mission operations systems. This has allowed us to develop the ground system in parallel with the flight system and to produce a highly configurable system to meet the needs of the mission operations team. We have supported these efforts with minimal staffing and have designed a system that will be applied to future missions, thus reducing cost to this and future projects. Through our use of this system we have seen significant improvement in our ability to define and process information efficiently, and have demonstrated a significant reduction in system turnaround by using XML and related technologies.

ACKNOWLEDGEMENTS

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

REFERENCES

Elliotte, R. H. and Means, W. S., 2002, *XML in a Nutshell, 2nd Edition – A Desktop Quick Reference*, O'Reilly and Associates, Inc, New York City

Burke, E.M., September, 2001, *Java and XSLT*, O'Reilly and Associates, Inc, New York City

McLaughlin, B. September, 2001, *Java & XML, 2nd* Edition - Solutions *to Real-World Problems, 2nd Edition,* O'Reilly and Associates, Inc, New York City

Tidwell, D., August, 2001, *XSLT*, O'Reilly and Associates, Inc, New York City

Pawson, D., August, 2002, *XSL-FO – Making XML Look Good in Print*, O'Reilly and Associates, Inc, New York City

Beazley, D. M., 2001, *Python Essential Reference, 2nd Edition*, New Riders Publishing, Indianapolis

W3 Schools Home Page, Tutorials – XML, XSL, Schema, Xpath, DOM, 2003, http://www.w3schools.com/2003

The Apache XML Project Home Page, 2003, http://xml.apache.org

The SAX Project Home Page, 2003, http://www.saxproject.org

The World Wide Web Consortium (W3C) XML Homepage, 2003, http://www.w3.org/XML

The Java Home Page, 2003, http://java.sun.com

The Python Home Page, 2003, http://www.python.org