XML META-DATA EXPERIMENTS

Gilles K/Bidy
L-3 Communications, Telemetry-West

ABSTRACT

As part of the technology upgrades driven by the iNet initiative, there is a need to establish a meta-data standard to describe configuration information for the system under test. The technology identified for such a standard is XML and XSD schemas. This paper presents findings from various experiments to import and export existing telemetry configuration information to XML based on the new Meta-data model. In addition, this paper will discuss the possible conversions to and from the existing IRIG TMATS standard.

INTRODUCTION

Over the past 10 years, the eXtensible Markup Language (XML) has revolutionized information sharing over networks. It has become a robust and mainstream technology for internet applications but also for data sharing across heterogeneous platforms. Users now have at their disposal multiple ways to ingest, parse and validate XML documents. The technology is particularly useful to define custom information in a structured hierarchical manner and provides excellent means to validate syntax and content of documents.

TELEMETRY APPLICATIONS

As a result, multiple initiatives are underway in our Telemetry user base to introduce XML as a standard technology to specify and transmit meta-data information. There is now a new version of TMATS based on an XML standard. There is also an ongoing effort to define meta-data information in XML form as part of the iNet project. These initiatives have already proven successful by gaining user buy-in and focusing on verifiable content, something that legacy tools could not do easily or at all.

This paper focuses on basic experiments to support the iNet Meta-Data task. The goal of this task is to define a generic solution to describe a telemetry system. As the standard matures, it is important to validate its feasibility with real telemetry systems in use today. To do so, various example XML instance documents and corresponding XSD documents (XML Schema Definition) have been provided to various vendors for test and integration with existing products. Additionally, this is also an opportunity for solution-providers and users to share critical feedback early, while the standard is being developed.

EXPERIMENT REQUIREMENTS

To successfully complete the initial experiment, we had to be able to accept XML instance files defining a basic telemetry system configuration. The initial focus was on PCM measurements and simple hardware configuration. A key part of the experiment consisted of validating the content of the instance documents using the supplied XSD document. Additionally, it was requested that round-trip XML be considered to produce a generic report of the process.

SOLUTION

The environment chosen for this experiment targeted the Vista product which is developed in Java™. Therefore, we looked for a solution where we could easily integrate the XML parser/validator with our existing Java™ software.

Java™ offers many options to process XML files, streams or objects. For instance, there are standard Application Programming Interface (API) classes for DOM, SAX and XSLT. But looking at the arsenal of tools available to us, we decided to use JAXB (Java XML Binding). JAXB is a technology that provides marshalling and unmarshalling of XML structures to Java objects. In other words, it provides easy manipulation of XML elements using Java objects. To do so, it is necessary to first parse the XML schema definition (XSD file) and create Java code on the fly to map the defined XML constructs to Java classes. This is done by running a schema translation command. It is then necessary to compile the resulting class files to compiled java objects (byte code binaries).

Our tests were focused on XML data ingest from a Ground Station perspective. Using the JAXB methodology described above, we translated the complete schema definition and built the necessary software to process the XML instance documents.

Then, we had to write code to traverse the XML instance document for PCM re-construction. In particular, the following constructs had to be handled:

- Measurements
- Sampled Measurements
- Transducers
- Units…

The resulting Java objects had to be mapped to native L-3 objects and were eventually stored in persistent storage (Vista SQL database). The imported database was further tested with actual modules to make sure that no invalid settings existed.

The following diagram provides an overview of this XML/XSD to Java™ translation process:

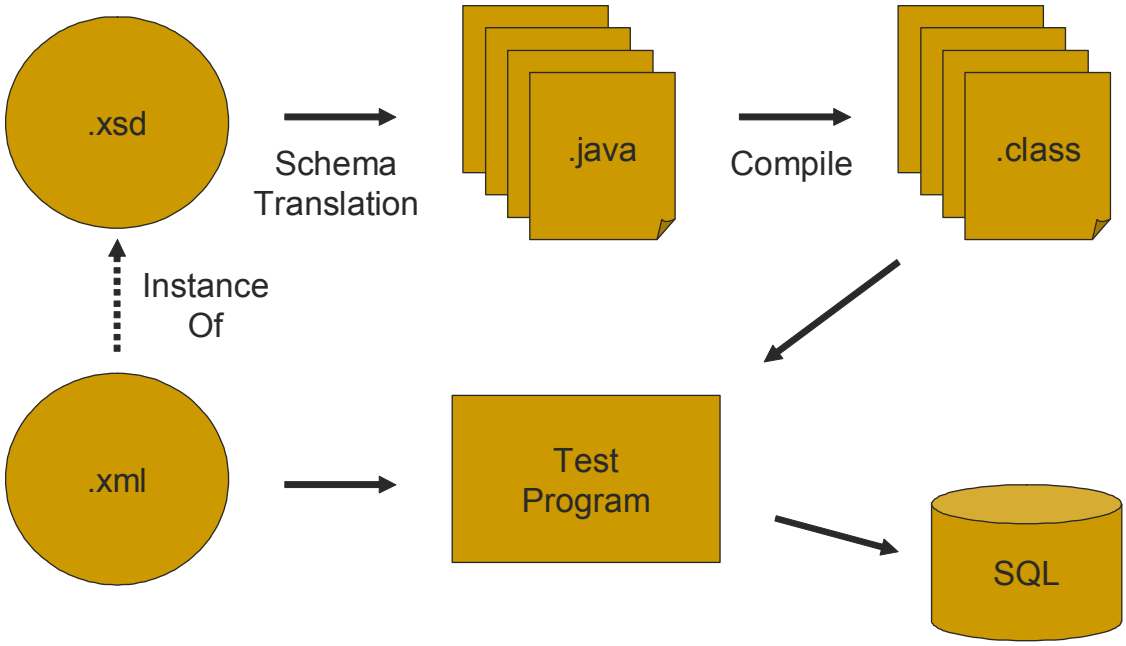

Figure 1. XML/XSD Translation Process

RESULTS

Using the JAXB approach proved very effective. We were able to complete the assignment within a few days.

In terms of technical issues with the XML/XSD tools, we really did not encounter any major problem. We used an ANT file to automate the various build steps (schema translation to Java, Java compile, and example code compile) and were up and running in short order.

In terms of content, we identified a lot of common meta-data descriptions between the generic files and the L-3 proprietary formats (XML and database schemas). It was easy to map PCM Measurement definitions, frame metrics, etc. We identified a few issues related to bit rate, frame sync word location etc but nothing significant.

The only significant drawback we encountered had to do with data organizational structure. It quickly became problematic to try to build information for a particular object that required information from XML constructs located at very different locations. To do so, we had to navigate through the XML hierarchy to pull the necessary information and this felt very artificial and convoluted at times.

EVOLUTION

A possible future evolution to the XML Meta-Data standard could be to fill the missing validation function for “business rules”.

Unfortunately, there is no standard or emergent solution today to apply business rule validation to an XML document. In practice, there is often a need to validate content beyond structure and datatypes. For example, it is often necessary to perform cross-field validation, i.e. if field x has this value, then field y can only have this other value. A good example of this would be a PCM frame defined with a 16-bit Sync Pattern length but the Sync Pattern field containing a 32-bit value. There is no way for the schema validation process to catch this. Both fields can be valid but incompatible in terms of intended use. Therefore, there needs to be another layer of validation where users can define business rules for this type of constraint on data content.

As part of the XML Meta-Data effort, it seems that we should approach this problem in layers and start addressing the issue of providing two levels of XML validation:

- Schema validation to catch structural and field definition errors
- Business rule validation to catch problems crossing the boundaries of individual fields.

CONCLUSION

Overall, the big lessons learned from the initial experiment were twofold. First, it is very important to keep the XML structure as lean as possible. Having a deep hierarchical structure greatly complicates the parsing process. Secondly, we should address the issue of business rule validation to avoid the proliferation of XML documents that may be correct in form but could contain invalid content.

This is an evolving standard and broad user input greatly helps define a mature, easy-to-use and generic approach. There is little doubt that XML was the right technological choice for this project.