Metawidget White Paper

Case Study: Global Navigation Satellite System

Richard Kennard

June 2013

http://metawidget.org

1. Introduction

This white paper presents a case study of using Metawidget to provide automatic User Interface (UI) generation for GALILEO, a next-generation global navigation satellite system.

2. Organisation and Product Overview

SCISYS is a leading developer of IT services operating in a wide range of markets and domains. They are based in the United Kingdom and Germany. GALILEO is a global navigation satellite system (GNSS) currently being built by the European Union (EU) and European Space Agency (ESA).

In the scope of GALILEO, SCISYS' Space Division are involved in the development of major ground segment assets like GACF, GNMF, CMCF.

3. Integration of Metawidget

SCISYS were looking for a UI generation solution as part of their product build. They had exacting requirements around data formats.

Heiko Müller, software engineer at SCISYS' Space Division, summarised their integration of Metawidget. “Signals between different applications inside a satellite ground segment are often encoded as XML structures based on central schema definitions (XSD). The schemas actually form a central part of the system’s interface definition and a lot of effort is spent on setting up their structure, declaring restrictions and attaching documentation notes”. It was important to SCISYS that this effort not be duplicated when it came to defining their UI.

“For the manual editing of the signals it is natural to think of the schemas as the core resource for an [UI] editor. In a pre-processing step we therefore use the Java Architecture for XML Binding (JAXB) to compile[1] the schema definitions into a Java library and add it as a project specific resource to the editor environment.”

“We then unmarshal the actual signal (i.e. XML file) and build up a customised `ObjectTreeModel` from the JAXB nodes. The `ObjectTreeModel` is used as the internal representation of the signal. It can be accessed in an *Explorer*-component (shown in Figure 1). Each node of the tree relates to a JAXB object of the unmarshalled signal.”

1Since the standard compiler ignores the definition of restrictions and documentations we implemented a plugin which adds this information to the generated code as annotations.

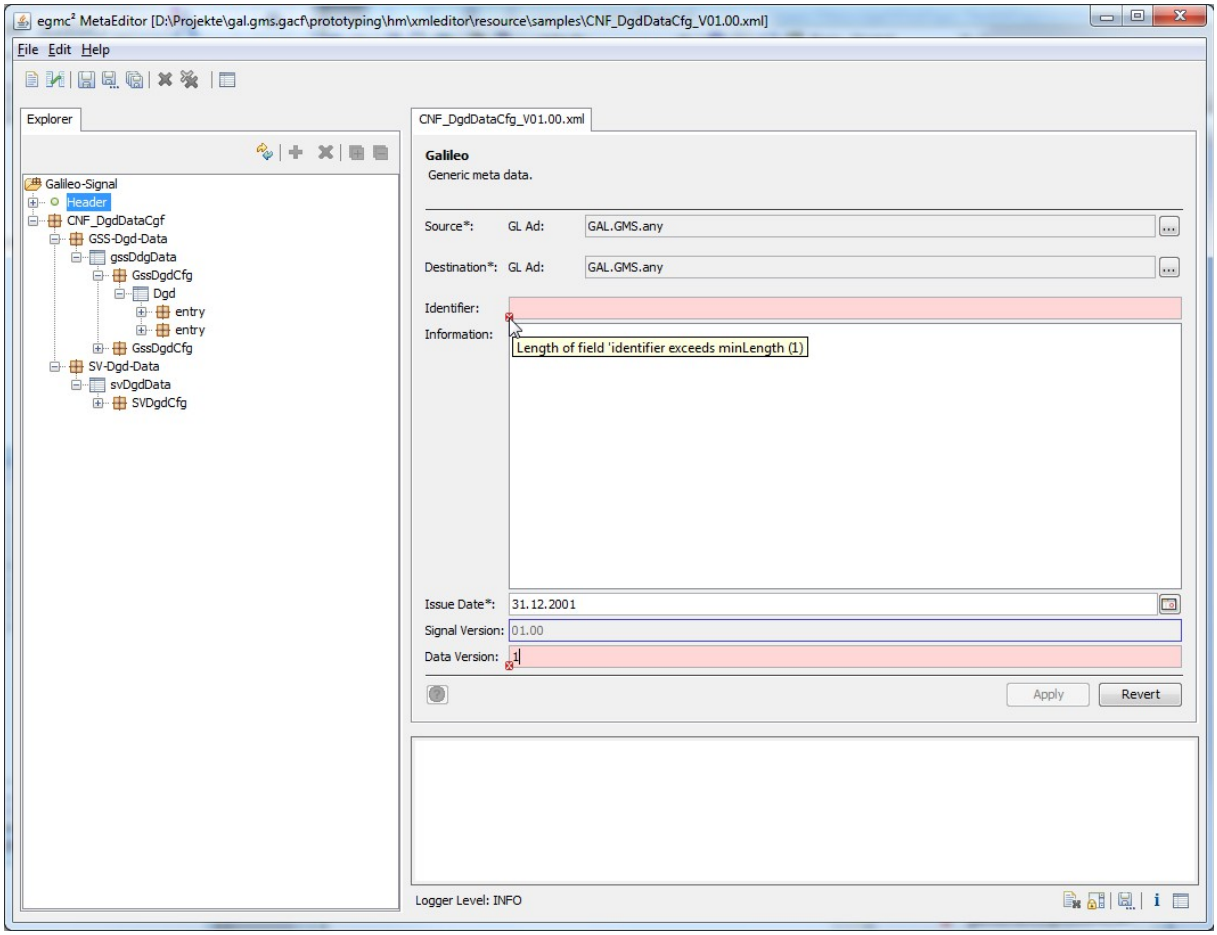

Figure 1: The editor showing some header information of a GALILEO ground segment signal. The right-hand section is assembled by a SwingMetawidget

The team immediately saw the value of integrating Metawidget into their editor. “Each time a node is selected in the Explorer we provide the related JAXB object to an instance of a `SwingMetawidget` and let it render the corresponding input form (seen on the right of Figure 1).

Heiko explained this approach avoids duplicating the effort put into the XML schemas. “The nice thing is that all the restrictions originally defined in the schema are automatically bound to the input fields. They are injected into the Metawidget pipeline by a `JAXBAnnotationInspector`”. The team were able to leverage Metawidget's pluggable widget processors to incorporate third-party libraries: “[the widgets] are evaluated using the JGoodies validation framework. Similarly documentation notes of the schema are injected as tooltips”.

The team were also able to leverage Metawidget's pluggable inspection architecture to envision scenarios beyond XML schema. “Of course the input for the editor is not restricted to XML files whose related schema has been pre-compiled via JAXB. You just need to provide what we call a `MetaContext` in order to support other input types. The standard `XMLDecoder` for example is used to decode serialized [Java]Beans. Also non-XML structures can be imported and visualised using the right `MetaContext` (Figure 2)”.

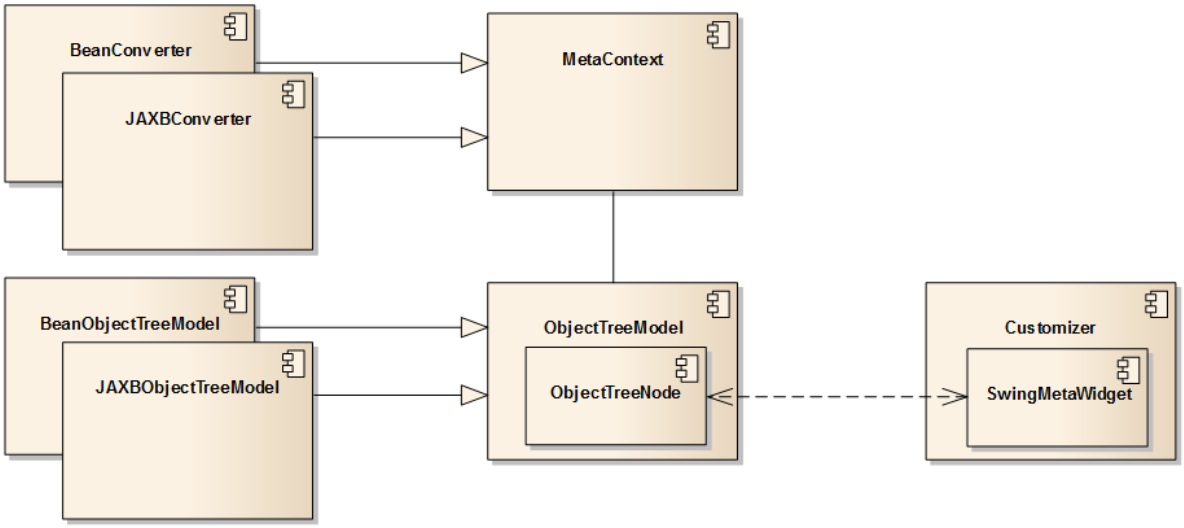

Figure 2: A MetaContext is responsible for decoding (e.g. unmarshalling) the input and providing an appropriate ObjectTreeModel. A SwingMetawidget is then used to display a selected ObjectTreeNode which is associated with the original object being edited. The variables of the object are bound to the UI via a modified BeansBindingProcessor

4. Conclusion

Finally, we asked Heiko to summarise the team's experience with Metawidget. “[It] was a good experience, both with respect to Open Source and Metawidget in particular. I believe our customer will be happy... Thanks again for your support”.

5. Resources

SCISYS: http://scisys.co.uk

GALILEO: http://esa.int/Our_Activities/Navigation/The_future_-_Galileo/What_is_Galileo

Metawidget: http://metawidget.org

This document has been produced under funding of the European Union. The views expressed herein can in no way be taken to reflect the official opinion of the European Union and/or ESA