

HierarchThis: An Interactive Interface for Identifying Mission-Relevant Components of the Advanced Multi-Mission Operations System

Caltech Summer Internship Program 2012

Final Report

Krystof Litomisky

krystof@litomisky.com

Eleanor Basilio (mentor)

Eleanor.V.Basilio@jpl.nasa.gov

Jet Propulsion Laboratory

California Institute of Technology

Introduction

Even though NASA's space missions are many and varied, there are some tasks that are common to all of them. For example, all spacecraft need to communicate with other entities, and all spacecraft need to know where they are. These tasks use tools and services that can be inherited and reused between missions, reducing systems engineering effort and therefore reducing cost.

The Advanced Multi-Mission Operations System, or AMMOS, is a collection of multimission tools and services, whose development and maintenance are funded by NASA. I created HierarchThis, a plugin designed to provide an interactive interface to help customers identify mission-relevant tools and services. HierarchThis automatically creates diagrams of the AMMOS database, and then allows users to show/hide specific details through a graphical interface. Once customers identify tools and services they want for a specific mission, HierarchThis can automatically generate a contract between the Multimission Ground Systems and Services Office, which manages AMMOS, and the customer. The

document contains the selected AMMOS components, along with their capabilities and satisfied requirements. HierarchThis reduces the time needed for the process from service selections to having a mission-specific contract from the order of days to the order of minutes.

Background

The process of selecting which AMMOS tools and services can be used for a specific mission typically proceeds as follows:

1. The mission customer, together with a Mission Interface Engineer (MIE) from the Multimission Ground Systems and Services Office, identify AMMOS components that are relevant for the mission.
2. The MIE figures out exactly what each of the selected components does, using a central requirements database.
3. The MIE synthesizes the information from the customer and the requirements database into a Service Level Agreement (SLA), which is the contract between the customer and the Multimission Ground Systems and Services Office.

The Service Level Agreement includes, among other things, a detailed description of all the AMMOS tools and services selected for the mission, as well as detailed descriptions of each tool's and service's capabilities.

The current process involves multiple people doing a large amount of collecting, copying and aggregating information from different sources. This can be time-consuming and error-prone. Automating parts of this process has the potential to decrease the time required for the process as well as to reduce the possibility of errors.

Such automation is the driving force behind the creation of SSOFI, the Single Source of Information. SSOFI is being created as the database of all AMMOS tools and services, including information about the requirements that each tool and service satisfies. When complete, SSOFI will serve as the framework through which the process of identifying and selecting mission-relevant AMMOS components is done.

Objectives

My task this summer was to create an interactive interface to SSOFI that can:

- Automatically create diagrams of AMMOS tools and services as well as their capabilities
- Dynamically modify those diagrams to show only the information that is important to the user at any given point of time
- Allow the user to select tools and services that are relevant to a specific mission
- Automatically generate a Service Level Agreement that includes only the components that were selected for the mission

The person interacting with SSOFI in this way is typically a Mission Interface Engineer. Therefore, in this document, “user” can generally be understood as “Mission Interface Engineer”.

Technical Approach

SSOFI, the Single Source of Information, is stored as a database of SysML models. SysML is the Systems Modeling Language, an extension of the Universal Modeling Language (UML). SysML is designed for modeling various kinds of systems structurally as well as behaviorally. This database is created and maintained using MagicDraw modeling software.

MagicDraw is implemented in Java, and allows access to certain parts of the program and the model through its Open API. Such access is possible using any language that can run in a Java Virtual Machine, most notably Java and Jython. I chose to use Jython, the implementation of the Python programming language that runs in a Java Virtual Machine. Jython is popular at JPL, and as a scripting language, it allows rapid prototyping and development.

In order to generate the Service Level Agreements, I used DocGen. DocGen is a MagicDraw plugin developed at JPL, designed to generate XML documents from “document models” in MagicDraw. To generate mission-specific SLAs, I created a template SLA document model. When the user has made their selections and is ready to generate the SLA, HierarchThis runs a script that copies over the relevant parts of the template document. In addition, the script inserts and populates tables in the document with information pulled from the model, including only information pertaining to the components that the user selected.

HierarchThis

Manipulating Diagrams

HierarchThis uses diagrams to convey information about available tools and services as well as to make selections. Figure 1 shows an example diagram that was created by HierarchThis. The blue boxes are used to organize the tools, the orange boxes are the tools themselves, and the pink boxes are the requirements that each tool satisfies.

Note that HierarchThis generates the diagrams based on the underlying model. This means that when a new tool is added to the model, it is a matter of a few quick clicks to generate an updated diagram. This is significantly faster than if an engineer had to create the diagram manually.

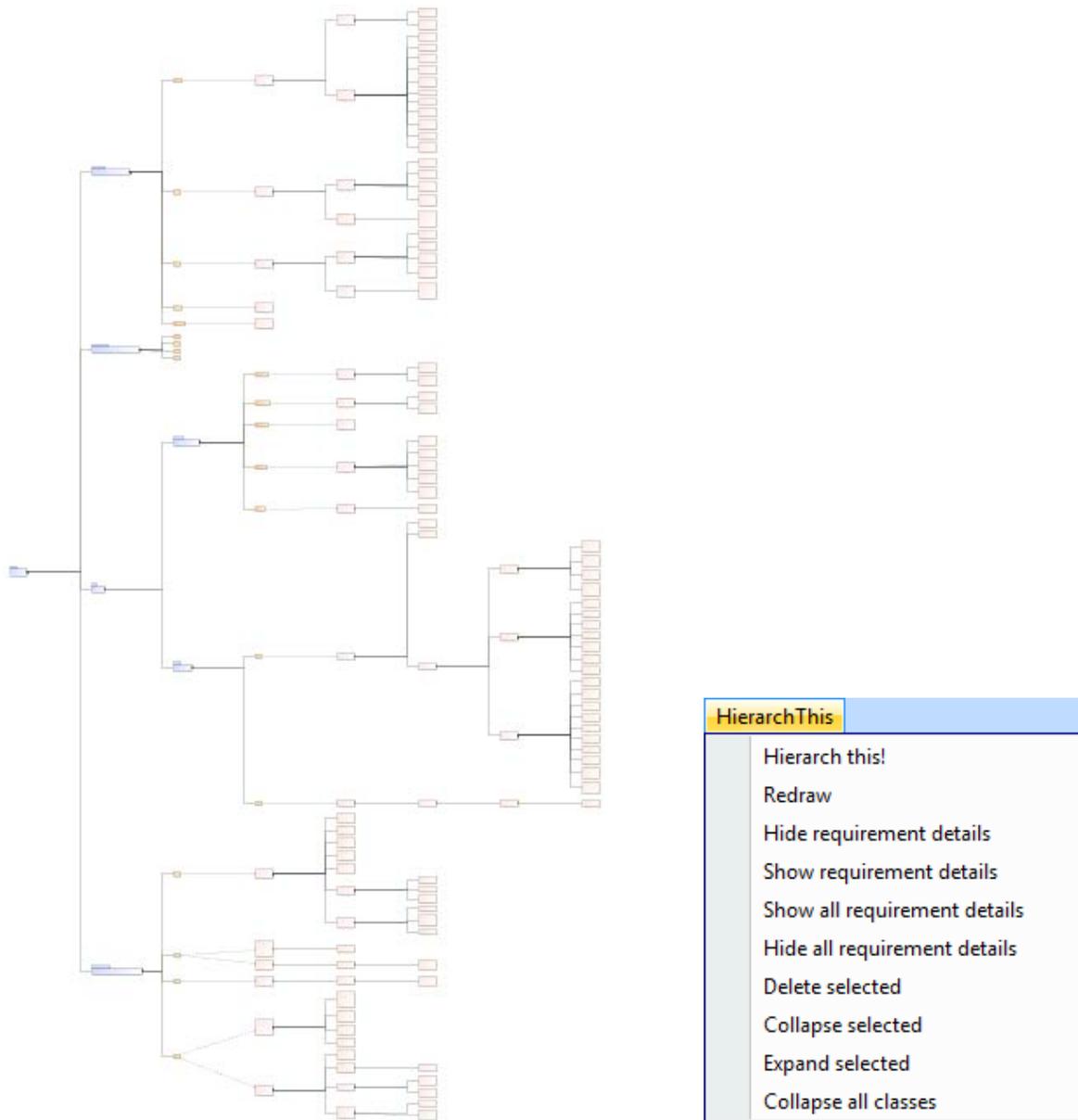


Figure 1. Left: diagram created by HierarchThis. Right: HierarchThis menu options.

Diagrams created using HierarchThis can be manipulated in certain ways through the HierarchThis menu. For example, if we just want to see the tools that are available without the requirements they satisfy, we can do so by selecting the “Collapse all classes” option. The resulting diagram is shown in Figure 2.

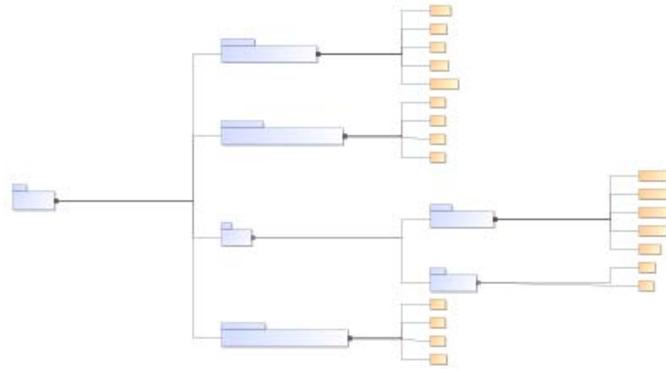


Figure 2: diagram with no requirements shown.

Conversely, if the user does want to find out more details about which requirements a tool or service satisfies, the user can do so by right-clicking on the tool/service in question, and using the “Expand selected” option. This shows all the satisfied requirements for that tool/service. The user can also show/hide details about requirements at will; figure 3 shows an example of what this looks like.

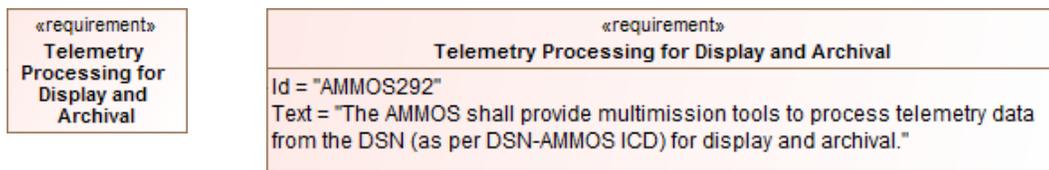


Figure 3: requirement details hidden (left), and shown (right).

Initially, HierarchThis creates a diagram of all of the available tools and services. As the user identifies components that they don’t need, the user can remove them from the diagram using the HierarchThis menu (“Delete selected”). This deletes the selected components, as well as any components hierarchically under them.

Generating Service Level Agreements

Once only the tools and services that are relevant for the mission are on the diagram, the Mission Interface Engineer can use HierarchThis to automatically generate a Service Level Agreement (SLA) that includes only those tools and services. This is easily done through the HierarchThis menu, as Figure 4 shows.



Figure 4: generating the SLA through HierarchThis.

This generates a document in the format used by DocGen, after which it is a matter of a few clicks to use DocGen to generate an XML version of the document. DocGen is a MagicDraw plugin developed internally at JPL to generate XML documents from “document models”.

The XML document can be easily edited through an XML editor; this experience is comparable to that of editing a MS Word document. At this point, the Mission Interface Engineer would enter additional information that is not (or cannot be) stored in the database, such as delivery dates or any mission-specific notes.

The XML document can then be exported to a PDF document, which would then be passed on to the customer as the contract.

Conclusions

During my time in the Summer Internship Program this summer, I developed HierarchThis, a MagicDraw plugin to streamline and automate steps of the process of identifying and selecting mission-relevant components of the Advanced Multi-Mission Operations System. To do this, HierarchThis automatically creates diagrams based on underlying models. These diagrams are interactive, allowing a Mission Interface Engineer to easily select components that are relevant for a particular mission. Once these selections are made, the Mission Interface Engineer can use HierarchThis to automatically generate a Service Level Agreement tailored to the mission. By doing all this, HierarchThis limits the potential for errors and reduces the time required for the process from days or weeks to minutes or hours.

Notes and Acknowledgements

This research was carried out at the Jet Propulsion Laboratory, California Institute of Technology, and was sponsored by Summer Internship Program and the National Aeronautics and Space Administration.

The development of HierarchThis led to a New Technology Report being filed and approved by Caltech. As a result, HierarchThis is copyrighted to the California Institute of Technology.

I want to thank my mentors, Eleanor Basilio and Sheldon Shen, for all their help and guidance, as well as for giving me the opportunity to work at JPL this summer. Other people who have helped me with the project include Greg Welz and Elyse Fosse.