Trade Space Exploration with Ptolemy II

Shanna-Shaye Forbes

Electrical Engineering and Computer Sciences
University of California at Berkeley

Technical Report No. UCB/EECS-2009-102
http://www.eecs.berkeley.edu/Pubs/TechRpts/2009/EECS-2009-102.html

July 21, 2009

Acknowledgement

This work was conducted with support from a Jenkins Predoctoral Fellowship Mini Grant Award.

Trade Space Exploration with Ptolemy II

Shanna-Shaye Forbes*
Dept. of Electrical Engineering and Computer Sciences
545 Cory Hall, University of California, Berkeley
Berkeley, CA 94720
sssf@eecs.berkeley.edu

July 21, 2009

Abstract

This work presents an extension of JPL's Trade Space Specification Tool used in early craft design. The extension explores Ptolemy II as a modeling and simulation environment for craft architecture combinations specified by the Trade Space Specification Tool. This extension creates a model generator that acts a bridge between JPL's Trade Space Specification Tool and the Ptolemy II environment to facilitate simulation of an architecture with the use of fuzzy logic.

1 Introduction

When designing a craft to explore outer space, numerous different architecture options need to be considered and one set of options selected for the construction of the craft. These decisions are generally made based on cost, level of risk, performance indicators, or many other constraints associated with a project. The exploration of these options, as well as the interconnections or linkages between the options referred to as Trade Space Exploration, is critical to fulfilling NASA's mission concepts. Trade Space Specification helps to determine which options are feasible and which should be considered after the initial brainstorming session. Since Trade space analysis is generally done as an ad-hoc process and is at times limited to the experience of experts in a field or to options that were used for prior projects, often times, a selection is made without properly exploring all possible craft architecture options.

A trade space specification tool is used to assist a spacecraft design team in the early design phase to conduct architecture trade studies through comprehensive trade space exploration based on a set of given rules. In the early design phase, there is no physical model to run simulation or to guide design trades. Instead, a set of rules helps to define feasible solutions and possible preferred paths. The selected architecture options can then link to physical models for simulation and point

*This work was conducted with support from a Jenkins Predoctoral Fellowship Mini Grant Award.

design trades in a later stage. A trade space specification tool that allows users to rapidly capture the trade space dimensions and options, capture the linkages among options, and analyze the feasible combinations is necessary and is currently being developed at JPL. This work extends JPL's Trade Space Specification Tool to automatically determine the combined cost, risk, and mass of components specified in an architecture. The extension is accomplished by the use of the Ptolemy II modeling and simulation framework[1].

2 Experimental Section

The objectives of the project are two-fold. The first is to use Ptolemy II for modeling of the various spacecraft architectures specified by the Trade Space Specification Tool. The model results will help determine performance, mass, costs, risk, and other trends based on architectures selected for the spacecraft. The second objective is to facilitate quick high-level actor development in the model environment in the initial stages of design.

To automatically determine the cost, risk, and mass of all the components within a craft architecture specification, one would need a model of each feasible architecture option in Ptolemy II. The Trade Space Specification Tool is able to output the options within an architecture but currently does not currently contain information about a model of the component/option itself. Since Ptolemy II uses actors for each component in a model, we realized we had two options to create an actor for each component. The first was to (1) create custom actors for each feasible option or (2) create one generic actor that can be customized for each architecture option.

Since the Trade Space Specification Tool focuses on the early design phase, we wanted there to be the use of a set of rules when there is no model and a need for rough order calculations for decisions. For this implementation, we used fuzzy logic. Our solution was to create a fuzzy logic actor within the Ptolemy II framework that uses Edward Sarzonov's Java Fuzzy Engine. Fuzzy Logic refers to reasoning that is approximate instead of precise. Common rules associated with fuzzy logic include: (1) if gas is expensive, then take the bus, and (2) if gas is cheap, then drive your car. Though the rules are approximate in terms of expensive and cheap the defuzzification of the terms produces a precise result based on the price of gas. It is also relevant to note that the specification of expensive and cheap can change without the need to change the fuzzy rules.

Each instance of the fuzzy logic actor expects rules governing its behavior to be specified in an XML file, whose name is the generic name of the component. In the case of a Solar Power component, the fuzzy logic actor expects to find its rules defined in the power.xml file. The expected format of the rules file is shown in Appendix A. The use of fuzzy logic allows quick, high-level actor development in the modeling environment.

To accomplish our second objective, we created the TSSTtoPTII model generator. The TSSTtoPTII model generator expects XML output from the TSST in the format shown in Appendix B.

[1]Ptolemy II [2] is an open source modeling and simulation framework being developed at the University of California-Berkeley that supports model-based design. Ptolemy II facilitates actor-oriented and object-oriented modeling. Actor-oriented modeling is an alternative to the established object-oriented methodology where objects are manipulated. Instead, actors allow actions to take place on the evolving data which flows through them [1]. Ptolemy II facilitates the modeling and simulation of the design of systems whose behavior is governed by directors implemented in the Ptolemy II framework.

TSSTtoPTII model generator, written in Java, reads the file filename.xml specified by the user and produces a MoML XML file filenameModel.xml to be run by Ptolemy II.

3 Results and Discussion

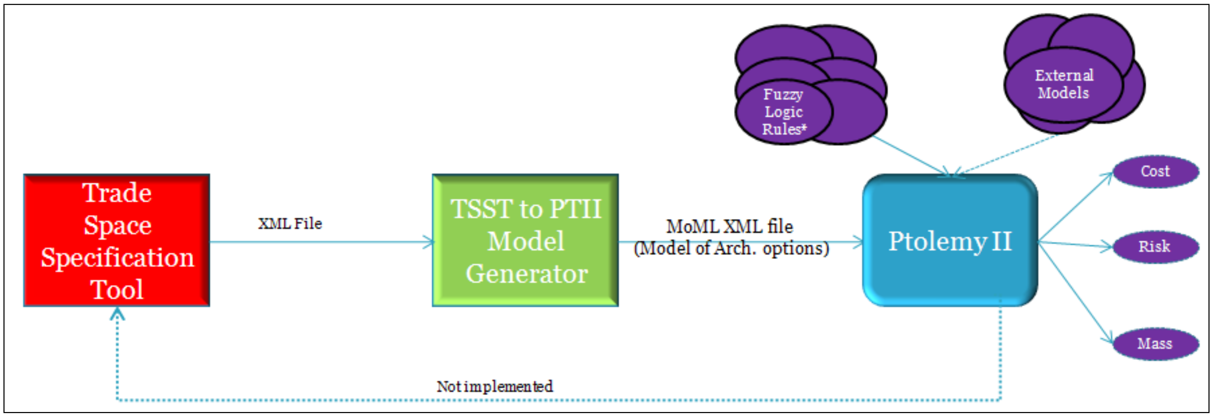


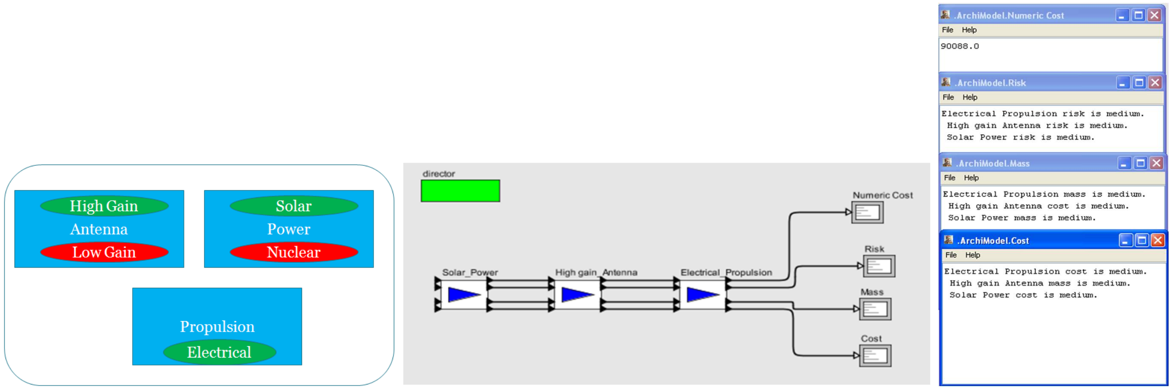
Figure 1: TSST to Ptolemy II Framework

We successfully created the initial TSSTtoPTII framework shown in Figure 1. Here the Trade Space Specification Tool produces an XML file that is sent to the TSSTtoPTII model generator. The TSSTtoPTII model generator produces a MoML XML file. Running Ptolemy II with the created MoML XML file determines the cost, risks, and mass of the model. Each fuzzy logic actor instantiated in the model has an associated XML file specifying its rules. These files contain fuzzy logic rules and relevant associated information for the model. Though shown in the framework diagram, it is important to note that we are not currently linking to external models for the simulation of components in this stage of the TSSTtoPTII framework.

In addition to creating the initial TSSTtoPTII framework, we successfully ran the initial framework on an architecture specified by the Trade Space Specification Tool. A visual representation of the architecture options in TSST is shown in Figure 2(a), the Ptolemy II model created from the specification is shown in Figure 2(b), and the output from the model is shown in Figure 2(c). Though fuzzy logic produces numeric outputs, in addition to a number representing the cost of a component, we also output information into low, medium, and high categories for the cost, risk, and mass of each component.

4 Conclusion

The Trade Space Specification Tool to Ptolemy II model generator created by this project demonstrates the effectiveness of using an existing modeling and simulation framework to determine the cost, risk, and performance indicators, such as mass of architecture options in the early phase of spacecraft design. We successfully demonstrated the use of the combined framework for a small



(a) Architecture Options (b) Model (c) Model Output

Figure 2: Ptolemy II model and output for a TSST architecture

number of architecture options and are considering numerous extension points for our first attempt at the combined framework.

References

[1] Edward A. Lee. Center for Hybrid and Embedded Software Systems Seminar on Model Engineering, October 21 2008.

[2] The Ptolemy Project. http://ptolemy.eecs.berkeley.edu/.

Appendix A

Listing 1: XML Format for a Fuzzy Logic Rules File

```
<?xml version=”1.0” encoding=”utf−8” standalone=”yes”?>
<FUNCTION_BLOCK>
  <VAR_INPUT NAME=”power” TYPE=”REAL” RANGE=”1␣3” />
  <VAR_OUTPUT NAME=”cost” TYPE=”REAL” RANGE=”20000␣60000” />
  <FUZZIFY NAME=”power”>
    <TERM NAME=”Solar” POINTS=”1␣0␣0␣2” />
    <TERM NAME=”Wind” POINTS=”0␣0␣0␣1” />
    <TERM NAME=”Default” POINTS=”2␣0␣0␣3” />
  </FUZZIFY>
  <DEFUZZIFY METHOD=”CoG” ACCU=”MAX” NAME=”cost”>
    <TERM NAME=”low” POINTS=”20000␣0␣0␣25000” />
    <TERM NAME=”medium” POINTS=”27000␣0␣0␣40000” />
    <TERM NAME=”high” POINTS=”40000␣0␣0␣60000” />
  </DEFUZZIFY>
  <RULEBLOCK AND=”MIN” OR=”MAX”>
    <RULE NUMBER=”1” TEXT=”if␣power␣is␣Solar␣then␣cost␣is␣low” />
    <RULE NUMBER=”2” TEXT=”if␣power␣is␣Wind␣then␣cost␣is␣high” />
    <RULE NUMBER=”3” TEXT=”if␣power␣is␣Default␣then␣cost␣is␣medium” />
  </RULEBLOCK>
</FUNCTION_BLOCK>
```

Appendix B

Listing 2: Input XML Format Expected by the TSSTtoPTII Model Generator

```
<linked-list>
  <gov.nasa.jpl.trades.ui.menu.ExportArchitecture_-ArchitectureExport>
    <architectureName>Office prefered</architectureName>
    <options class="linked-list">
      <gov.nasa.jpl.trades.ui.menu.ExportArchitecture_-ArchitectureExport_-OptionExport>
        <optionName>Solar</optionName>
        <associatedDimensions>Power</associatedDimensions>
        <metadata>
          <description>
            <entryBy>Unknown</entryBy>
            <lifeStart>2009-06-26 21:40:20.590 PDT</lifeStart>
            <lifeEnd>2019-07-04 21:40:20.590 PDT</lifeEnd>
          </description>
        </metadata>
        <this_-1 reference="../../.."/>
      </gov.nasa.jpl.trades.ui.menu.ExportArchitecture_-ArchitectureExport_-OptionExport>
      <gov.nasa.jpl.trades.ui.menu.ExportArchitecture_-ArchitectureExport_-OptionExport>
        <optionName>High gain</optionName>
        <associatedDimensions>Antenna</associatedDimensions>
        <metadata>
          <description>
            <entryBy>Unknown</entryBy>
            <lifeStart>2009-06-26 21:40:35.949 PDT</lifeStart>
            <lifeEnd>2019-07-04 21:40:35.949 PDT</lifeEnd>
          </description>
        </metadata>
        <this_-1 reference="../../.."/>
      </gov.nasa.jpl.trades.ui.menu.ExportArchitecture_-ArchitectureExport_-OptionExport>
      <gov.nasa.jpl.trades.ui.menu.ExportArchitecture_-ArchitectureExport_-OptionExport>
        <optionName>Electrical</optionName>
        <associatedDimensions>Propulsion</associatedDimensions>
        <metadata>
          <description>
            <entryBy>Unknown</entryBy>
            <lifeStart>2009-06-26 21:40:42.824 PDT</lifeStart>
            <lifeEnd>2019-07-04 21:40:42.824 PDT</lifeEnd>
          </description>
        </metadata>
        <this_-1 reference="../../.."/>
      </gov.nasa.jpl.trades.ui.menu.ExportArchitecture_-ArchitectureExport_-OptionExport>
    </options>
    <outer-class/>
  </gov.nasa.jpl.trades.ui.menu.ExportArchitecture_-ArchitectureExport>
</linked-list>
```