The Sellers-Chell Method:

A Standardized Technique for Assessing and Selecting a Model-Based Systems Engineering (MBSE) Tool

Brian Chell and Dr. Jerry Sellers

Abstract

Model-Based Systems Engineering (MBSE) is a growing discipline for applying modeling tools to capture, connect, communicate and control a wide variety of system and project information throughout the life-cycle. A major challenge for engineers and organizations eyeing a move toward MBSE within their projects is the large, and growing, number and variety of tools available. Yet the selection of a specific tool has wide-ranging, enterprise-level implications for an entire organization, potentially influencing, enhancing (or constraining) systems engineering practices and protocols at almost every level. Despite the importance, tool selection is often arbitrary or narrow, focusing on the needs of a single user without wider consideration for the needs of the organization. Surprisingly, few studies have done on systematic techniques to compare and contrast tools in ways that will support organizational selection. The research in this paper describes a standardized technique to help address this problem. The Sellers-Chell Method (SCM) uses the 17 basic Systems Engineering processes described in the NASA Procedural Requirements document 7123.1B as a framework to create an objective "Tool Capabilities Inventory." The method then asks the assessor to create a simple, pre-defined system model example in the tool under consideration and then perform a "Tool Usability Assessment" derived from the famed Cooper-Harper pilot workload scale. The SCM was used to assess several different MBSE tools and found to have face validity in providing consistent results. While far from perfect, it has built-in flexibility to allow users to tailor it to their specific needs and offers the opportunity for fair and consistent results leading to better fit between a specific MBSE tool and an organization's needs.

CONTENTS

NOMENCLATURE

COTS	Commercial off-the-shelf
INCOSE	International Council on Systems Engineering
MBSE	Model-Based Systems Engineering
OMG	Object Management Group
SCM	Sellers-Chell Method
SysML	Systems Modeling Language
TCI	Tool Capabilities Inventory
TUA	Tool Usability Assessment
UML	Unified Modeling Language

1. Introduction

As Model-Based Systems Engineering becomes more popular many individuals and organizations are looking to introduce it into their systems engineering practices. A major challenge in doing this is the difficulty of choosing a tool with which to do so. There are many different choices yet no accepted industry standard for how to select one. As a result, these individuals and organizations might purchase a tool that does not have the capabilities and characteristics they require. Conversely, they might find a tool that does, yet allocate too many resources into the selection process. One way to mitigate these negative outcomes is to create a framework in which an individual or a team could test MBSE tools and assess them in a consistent and straightforward manner.

There are several examples in previous literature of ways in which to assess general commercial off-the-shelf (COTS) software. Within these examples there are many that have been used by organizations to evaluate and select software successfully, but only a few instances in which these methods have been used to select MBSE tools specifically. However, to be more widely applicable, these general methods need to be refined to provide the specificity required for selection within a given domain. This paper provides a discussion of the development and implementation of the Sellers-Chell Method (SCM), a standardized technique explicitly designed to assess MBSE tools.

The SCM has two facets, a *Tool Capabilities Inventory* (TCI) and a *Tool Usability Assessment* (TUA). The Tool Capabilities Inventory was based upon the 17 systems engineering processes as defined in NASA 7123.1B. The evaluators will first decide which of these processes they want their MBSE tool to assist with and then use the tool to create the products related with that process. For instance, the first process in the NASA Systems Engineering Engine is "Stakeholder Expectations Definition," one of the products that could be created to show a tool can perform this process would be a stakeholder context diagram.

Once the capabilities have been assessed, then the *Tool Usability Assessment* is used to further differentiate the various tools. The TUA is derived from a modified version of the Cooper-Harper rating scale Cooper, Harper (1969), which is an international standard for rating the handling qualities of aircraft.

The TUI scale rates the MBSE tool from one to ten on how easy it was to create the chosen systems engineering products. It can also include comment data to make the assessment more complete.

After testing this method on many different MBSE tools it was found that together, the SCM using the *Tool Capabilities Inventory* and the *Tool Usability Assessment* provides a standardized, yet flexible, technique to successfully evaluate and choose an MBSE tool for any organization.

2. Background

In this section a short introduction to MBSE and SysML will be provided. In addition, the paper will review previous work done on the topics of software selection methods, specifically when it was used to choose an MBSE tool. Finally, the paper will describe the NASA Systems Engineering Engine and the Cooper-Harper Scale that form the basis for the TCI and TUA respectively.

2.1. Model-Based Systems Engineering and SysML

Model-Based Systems Engineering is a relatively new way of using software models to replace much of the documentation generally required for systems engineering. Its usage can span the entire lifecycle of a project from start to finish and from the system of systems level down to the component level. It has been documented to improve communication with stakeholders and within the engineering team as well as improve quality, increase productivity, and reduce risk on designs. It is especially beneficial for complex projects. As defined in Sellers (2016) "Model-based systems engineering is the art and science of applying software based tools to capture systems engineering evidence in a systematic, disciplined way, to connect system relationships, control system configuration and communicate a common system site picture in the form of an integrated model to all stakeholders throughout the lifecycle."

SysML, created by the Object Management Group (OMG), is a widely used modeling language for the performance of MBSE. It uses and extends on the Unified Modeling Language (UML) to provide useful systems engineering functionality such as the ability to input requirements. It uses syntax more suited to systems engineering than does UML, which is largely used for modeling software. As defined by Finance, G. (2010) it is "A standard modelling language for systems engineering to analyze, specify, design, and verify complex systems, is intended to enhance systems quality, improve the ability to exchange systems

engineering information amongst tools, and help bridge the semantic gap between systems, software, and other engineering disciplines." It is assumed that the reader has some familiarity with both MBSE and SysML so a lengthy description of each is not required. For additional information about both MBSE and SysML the reader is invited to review references 6, 7, and 21.

2.2. Previous Research into Software and MBSE Tool Selection

This subsection will examine previous examples of software selection methods with focus on when they have been applied to MBSE tools.

2.2.1. COTS Selection Processes

There are many journal references describing methods for evaluating and selecting Commercial Off-The-Shelf (COTS) software. A good overview of these methods can be found in "COTS Selection Best Practices in Literature and in Industry" Land et al (2008) in which an extensive research of COTS selection literature was performed as well as a survey of experts who have roles in selecting COTS software. During their research, Land et al (2008) found several consistencies between the different methods in that they largely shared 4 processes; "there is a *preparation process*, an *evaluation process*, a *selection process* and (only in some of the methods) *supporting process(es)*." Within these four processes they found further consistencies allowing them to make thirteen recommendations for groups wishing to select COTS software.

One of the papers referenced by Land et al (2008) was "COTS Software Selection Process" by Lin et al (2006). The method described in this paper uses all four of the selection processes listed above and can be used to select COTS software for any type of project. This conclusion is particularly interesting in the context of the SCM described in this paper because another team, de Jong et al (2011), used it as a reference to create a method for evaluating and selecting a SysML tool for Sandia National Laboratories.

The de Jong team first made a list of the stakeholder's needs and then split them up between which ones were deemed as mandatory or desired. Many of these needs addressed the supporting processes as described above. Due to constraints in time and budget the team eventually used a Pugh Matrix and Pairwise Comparisons to select two tools among three choices.

A similar method can be found on a website by PivotPoint Technology Corp. (2014) where there is an article containing a process for selecting SysML tools for MBSE. This process is quite similar to the approach eventually used by the Sandia team. The PivotPoint approach starts by defining functional and nonfunctional needs, weighting them, and then testing and evaluating a group of tools. This process was used on the PivotPoint website to give ratings and brief reviews for nine different tools.

2.2.2. INCOSE Tools Group

The International Council on Systems Engineering (INCOSE) Tools Database Working Group has provided lists of many different tools that could be used for systems engineering, going well beyond the scope of MBSE. As for finding the best of these tools they sent out surveys to makers of system architecture and requirements management tools with questions as to their capabilities. These resulted in an extensive capabilities matrix that shows, according to the vendors, whether or not a given tool can perform a certain function within those domains. The lists were last updated in December of 2002 and the number of available tools, as well as their capabilities have evolved considerably since then.

2.2.3. Steiner Presentation

In Steiner (2008) a presentation was made that posed some questions about SysML tool functionality that a corporation might want to answer when choosing a tool. Then, using the Water Distiller Example from OMG's SysML tutorial, a test was conducted regarding the functionality of four different tools. Steiner found that none of the four tools as of 2008 were able to "'fully' implement SysML. The presentation explored further into how each tool displayed the different diagrams and noted some differences between them.

2.3. NASA Procedural Requirements Document 7123.1B

In this subsection the NASA Procedural Requirements Document 7123.1B will be discussed. It is important in the context of this paper as it underpins the first of two major facets upon which the SCM is based. The NASA Procedural Requirements Document 7123.1B was written "to clearly articulate and establish the requirements on the implementing organization for performing systems engineering." NASA (2013) In it the entire domain of systems engineering is broken down into 17 processes, split up between

three groups, System Design Processes, Product Realization Processes, and Technical Management Processes. These processes apply lessons learned over NASA's history to give a "general overview" of how systems engineering should be performed at NASA and since then these 17 processes have become industry agreed tasks. Using these tasks NASA created a Systems Engineering Engine which can be seen in Figure 1 below. It is very similar to the well-established "V-Model" that is widely accepted among systems engineers. The systems engineering processes described in NASA 7123.1B has been used as the foundation for the TCI due to their widely accepted face validity. In Appendix B the purpose of each process as given by NASA 7123.1B is given.

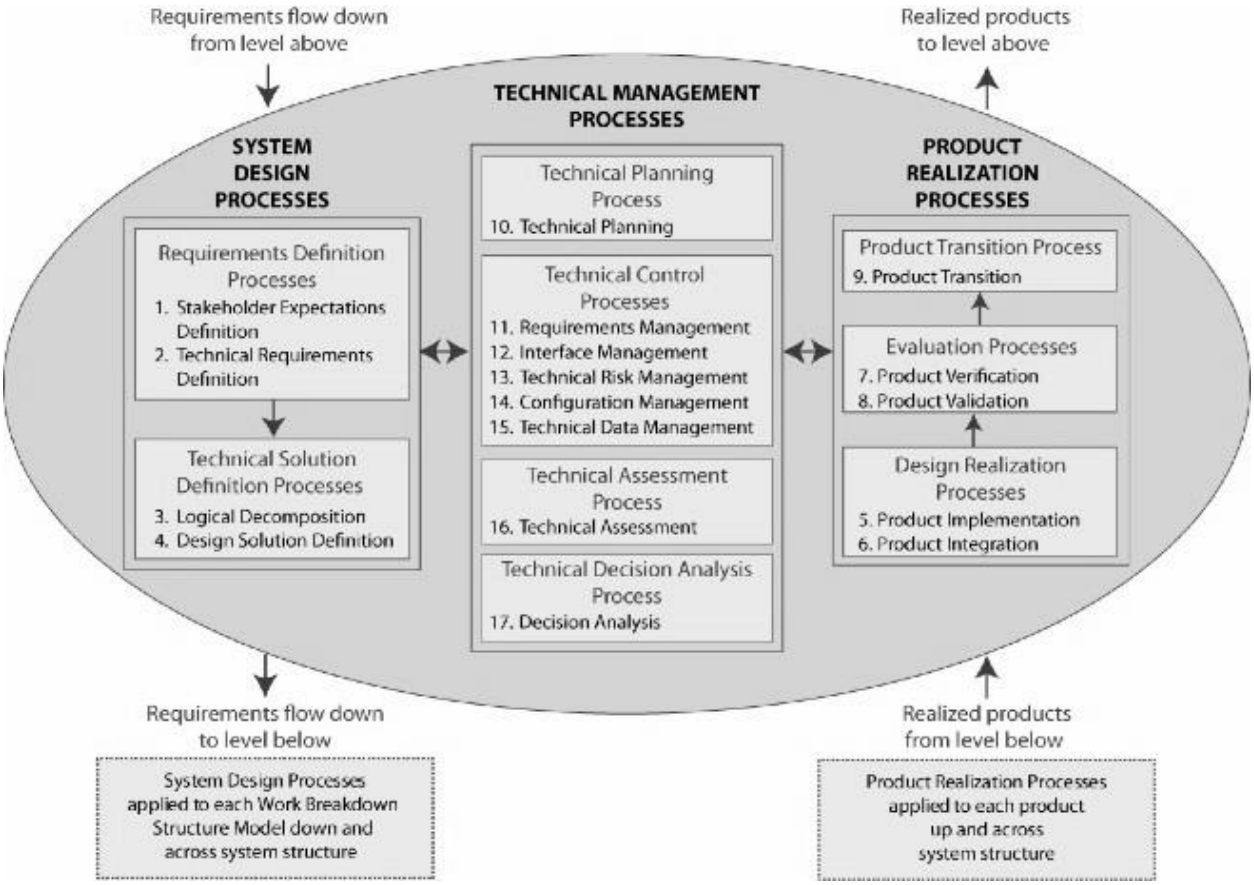


Figure 1: NASA Systems Engineering Engine

2.4. Cooper-Harper Scale

In this subsection the Cooper-Harper Scale will be discussed. It is the basis of the TUA, the second major facet of the SCM. The Cooper-Harper Scale is a way for pilots to evaluate the flight handling characteristics of airplanes. Its first version was published in 1957 and was then updated to its current form in 1969 and has become an internationally accepted way to test these characteristics. The Cooper-Harper Scale relies on a decision tree allowing the pilot to rate the aircraft's performance and then uses pilot comments to help define that rating. As stated in Bailey et al (2009), "The rating is subjective yet, due to the structure of the scale and by appropriate execution of the test, training, and its protocol, it quantifies the vehicle's handling characteristics for a given task."

The decision tree, shown in Figure 2 below, is "fundamental to the attainment of meaningful, reliable, and repeatable ratings." Harper, Cooper (1984) It is set up so that the pilot faces first a series of yes/no questions asking about the "Adequacy for Selected Task or Required Operation" starting with the most fundamental characteristic of flight handling, "Is it controllable?" If the aircraft is not controllable then there are no more decisions to make and it receives the worst score of 10. However, if the aircraft is controllable then two further yes/no questions are available. Once the pilot decides that the decision is a "yes" then they have one of three choices to make regarding both the aircraft characteristics and the "Demands on the Pilot in the Selected Task or Operation." Making this decision yields the pilot rating.

Noting that only having a rating "leads to the assumption that the numerical pilot rating can represent the entire qualitative assessment," Cooper and Harper referred to the pilot's comments as the "backbone" of the scale. Instead of breaking the 10-point scale and giving a score of, for instance, 3.5, they wanted these qualifications to be handled by comment data.

The Cooper-Harper Scale is well-validated and has been used to evaluate many tasks including general workload Hill et al (1992), spacecraft handling qualities Bailey et al (2009), and unmanned vehicle user-interface displays Cummings et al (2010). The Tool Usability Assessment that forms the second fundamental facet of the Sellers-Chell Method (SCM) is derived from the Cooper-Harper Scale.

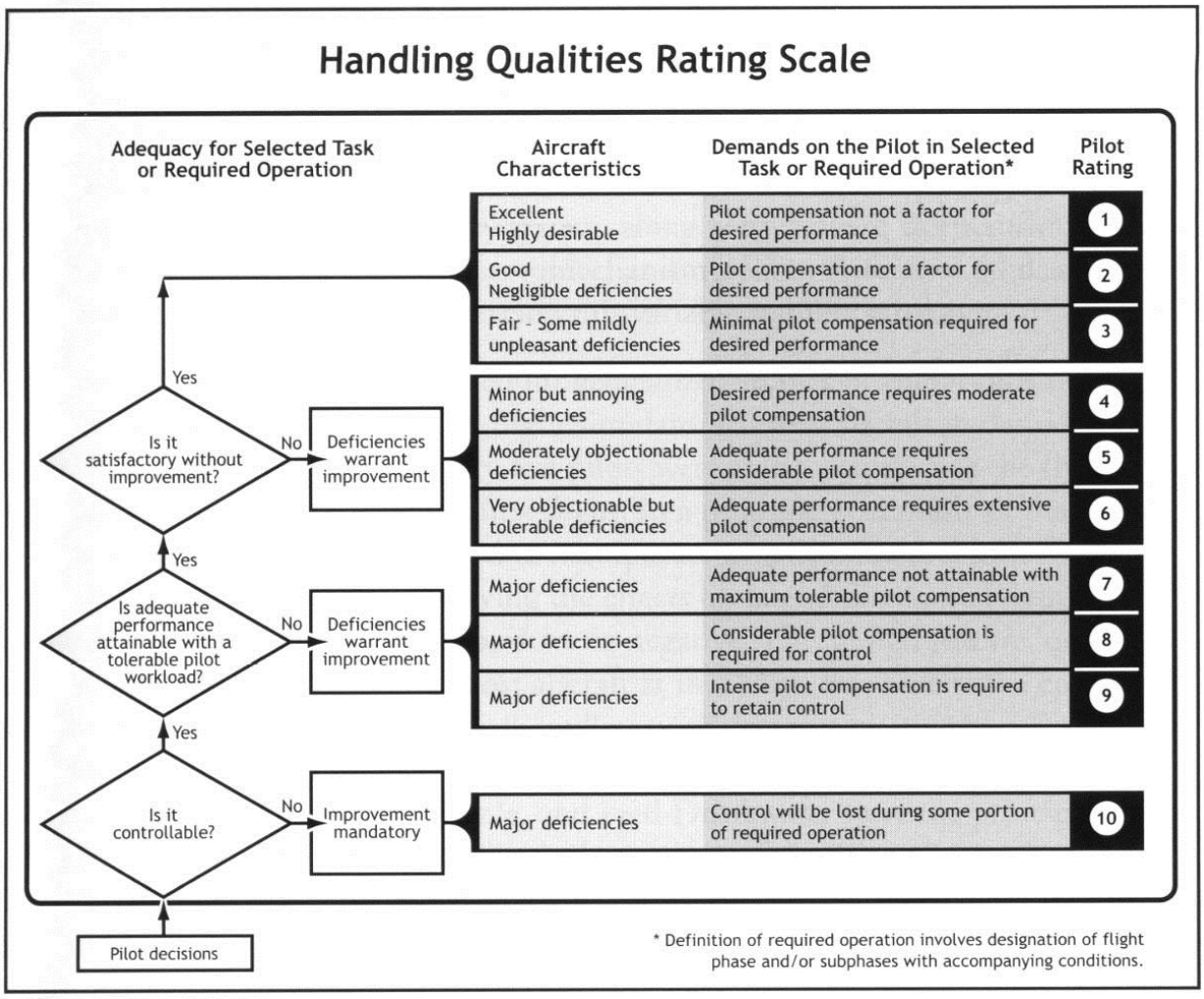


Figure 2: Cooper-Harper Rating Scale

2.5. Approach

The SCM builds upon the previous research into evaluating COTS software and the times it has been used for MBSE tools. This is mainly done by providing structure for many of the steps that are generalized when using one of the domain independent COTS software selection methods. The SCM uses the systems engineering processes defined in NASA 7123.1B and the Cooper-Harper Scale to create a Tool Capabilities Inventory and a Tool Usability Assessment which when used together can evaluate a tool both subjectively and objectively. It is intended to be easy to follow and not be as time-consuming as other methods for selecting software. A list of activities has been made that an individual or a team can follow

to find an MBSE tool that will accomplish the systems engineering processes it is required to. This approach utilizes some ideas from COTS software selection techniques to structure the method but the foundation is the 17 systems engineering processes from NASA 7123.1B and a Cooper-Harper Scale modified to have relevance for MBSE tools. It should be noted that this paper is focusing on what are considered to be MBSE tools and therefore the method explained here will have less relevance to model based engineering tools such as Simulink or Modelcenter.

2.5.1. Tool Capabilities Inventory

The Tool Capabilities Inventory (TCI) is a matrix that allows the user to evaluate all of the systems engineering processes in NASA 7123.1B as well as general supportability attributes of a potential MBSE tool. Similar to the approach taken by de Jong et al (2011) and Steiner (2008), the TCI asks the user to first think about their needs. In this context, their needs are described by the specific products of the systems engineering lifecycle they want to be supported by the tool, as well as what characteristics they want that tool to have.

As can be seen in Figure 3 below, the Tool Capabilities Inventory contains a list of products that a project may want to create during each of the systems engineering processes in NASA 7123.1B. At the end of the TCI is the General Supportability section that suggests other attributes that the organization might want to consider in their evaluation, such as whether the tool has platform independence, floating licenses, or adequate customer support. Empty spaces are given at the end of the inventory for a team to input their own specific supportability needs.

To use the Tool Capabilities Inventory the individual or team that is performing the evaluation would assign a weight to each of the capabilities or characteristics that are important for their needs. With such a long list, it is likely that some or perhaps many of these will receive a weight of zero. In the right-hand column is the score denoting how well the tool performs creating a product or how robust its supportability characteristic is. A score of zero is poor, and correlates to a score of 7-10 on the Tool Usability Assessment. A score of one would correlate to 4-6, and a two with 1-3. In terms of general supportability characteristics, a score of 0 would mean the tool does not have or support this characteristic,

1 would mean it has this characteristic to some but not fully satisfactory extent, and 2 would mean the tool has this characteristic to a satisfactory extent. The lack of granularity in this 3-level scale needs to be made up with comments and discussion amongst the team as is necessary with aircraft handling ratings done with the Cooper-Harper Scale. Of course, there is inherent flexibility here as a team can change these rating scales to best fit their needs and comfort levels.

Tool Capabilities Inventory		
Design		
Can the tool make these products?		
1. Stakeholder Expectations Definition	**Weight**	**0, 1, 2**
Stakeholder Context Diagram		
OV-1 Diagram		
Data Flow Diagram		
Needs/Goals/Objectives Document		
Conops Document		
SEMP		
2. Technical Requirements Definition	**Weight**	**0, 1, 2**
Requirements Document		
SysML Requirements Diagram		
Key Performance Parameters list		
Requirements Traceability Diagram		
Requirements Validation Report		
QFD Matrix		
3. Logical Decomposition	**Weight**	**0, 1, 2**
Context Diagram		
SysML Activity Diagram		
SysML Block Definition Diagram		
NxN Diagram		
Use Case Diagrams		
IDEF0 Diagram		
Sequence Diagram		
FFDB Diagram		
4. Design Solution Definition	**Weight**	**0, 1, 2**
Product Breakdown Structure		
Internal Block Diagrams		
Asset Diagrams		
Decision Trees		
Pugh Matrices		

State Diagrams		
Realize		
5. Product Implementation	**Weight**	**0, 1, 2**
Decision Trees to support buy, build decisions		
WBS Workpackages		
6. Product Integration	**Weight**	**0, 1, 2**
AIV Flow Diagrams		
7. Product Verification	**Weight**	**0, 1, 2**
Verification Requirements		
Verification Event Flow		
Verification Task Definitions		
Verification Task Procedures		
Verification Plan		
8. Product Validation	**Weight**	**0, 1, 2**
Validation Requiremetns		
Validation Event Flow		
Validation Task Definitions		
Validation Task Procedures		
Validation Plan		
9. Product Transition	**Weight**	**0, 1, 2**
Transition Flow Diagram		
Manage		
10. Technical Planning	**Weight**	**0, 1, 2**
WBS		
Network		
Gantt Chart		
11. Requirements Management	**Weight**	**0, 1, 2**
SRD		
Baseline Definition		
Requirements Validation Quality Scores		
12. Interface Management	**Weight**	**0, 1, 2**
IxI Matrix		
13. Technical Risk Management	**Weight**	**0, 1, 2**
Risk Matrix		
Fault Tree		
Mitigation Traceability		
14. Configuration Management	**Weight**	**0, 1, 2**
Change History		
15. Technical Data Management	**Weight**	**0, 1, 2**
All		
16. Technical Assessment	**Weight**	**0, 1, 2**

Entrance and Success Criteria (for each baseline) linked to products		
17. Decision Analysis	**Weight**	**0, 1, 2**
Decision Trees		
General Supportability		
Does the tool have these characteristics?	**Weight**	**0, 1, 2**
Usability		
Platform Independence (PC, Mac, etc.)		
Securability		
Supports Classified Environment		
Floating Licenses		
Adequate Customer Support		
Used elsewhere in Company		
Cost		
Scalability		
Collaboration		
Documented API		
Parametric Modeling		

Figure 3: Tool Capabilities Inventory

Once the desired products and general supportability characteristics have been given weights, the evaluators are asked to attempt to develop a specific set of systems engineering products for a sample system of interest. Once this is complete, they will then apply the Tool Usability Assessment to find the score for each chosen process.

2.5.2. Tool Usability Assessment

The Tool Usability Assessment makes it easier for the evaluator to consistently assign scores for the products created in the Tool Capabilities Matrix. It is derived from the Cooper-Harper scale described above and is shown in Figure 4 below.

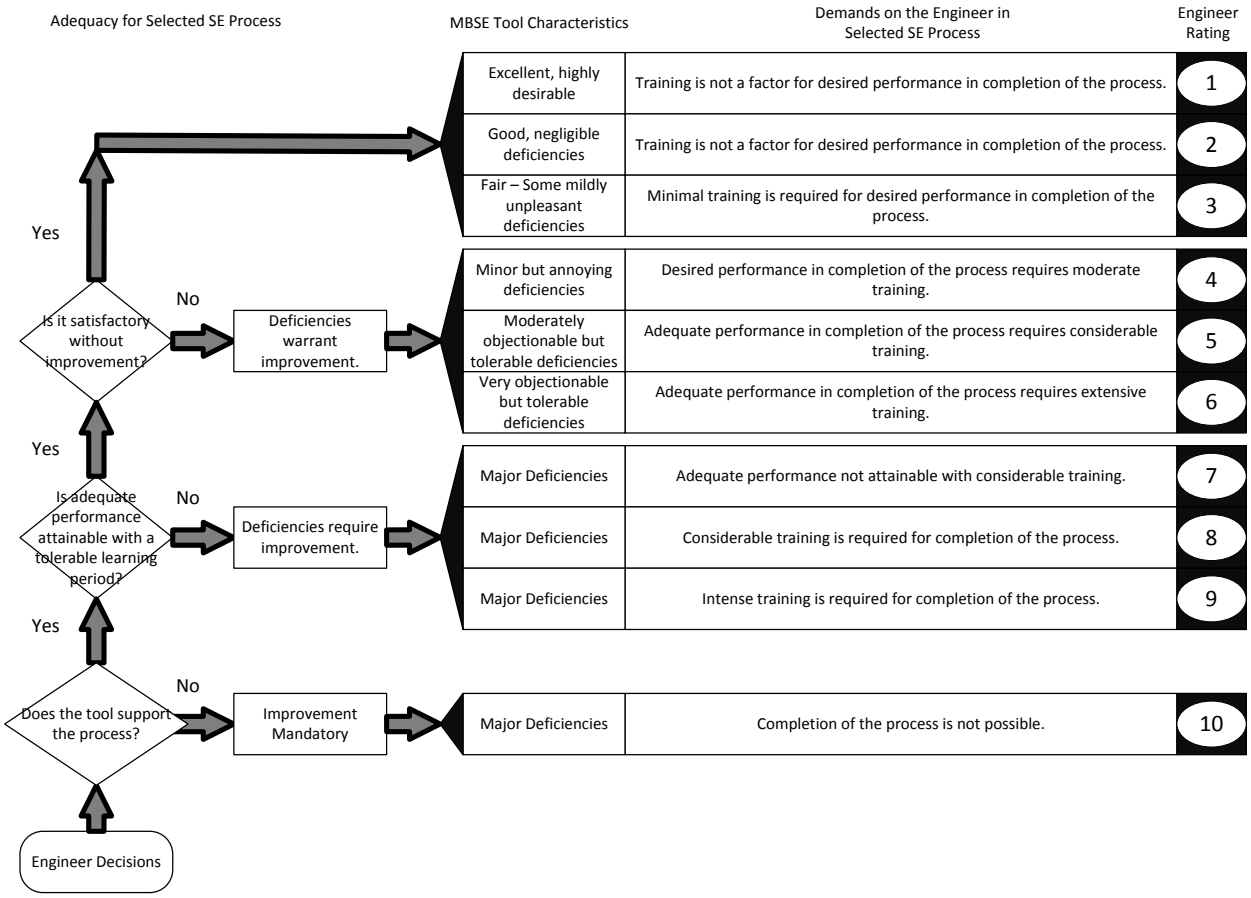


Figure 4: Tool Usability Assessment

Note that the TUA stayed true to the original language of the Cooper-Harper scale for both the questions and performance definitions. This consistency was applied to maintain the overall validity of the Tool Usability Assessment to the greatest extent possible.

To use the TUA, a user simply starts at the bottom-left and answers up to three binary questions. Once they move on to the right side of the diagram, the evaluator will make a final decision on the score by looking at both the MBSE tool characteristics and the demands that were placed on them in making the product. Once this decision has been made a score of 0, 1, or 2 can be written into the Tool Capabilities Inventory.

There is a fundamental difference between the Cooper-Harper approach to evaluating flight characteristics and the approach taken here for evaluating MBSE tools. This difference stems from the fact that the pilot and engineer roles in the original Cooper-Harper approach will be assumed by the same person or people in this approach. In the original scale, comment data was seen as being very important and necessary to gain a full understanding of the ratings. Had the pilots given the engineers simply a group of numbers to represent their evaluations of the plane's flight characteristics, then the engineers would not have known what to change had the plane been given a bad score.

Evaluator comments and discussion are still very important in this method as it will define the score but can also help to clarify any biases that some evaluators might have from experience in using one tool. This bias is explored more in the results section of this paper. Of course, once the final scores are tallied, discussion within the team is necessary to select a tool.

3. The Sellers-Chell Method

Following is a list of instructions outlining the Sellers-Chell Method proposed by this paper. The instructions for using the tool under evaluation to develop products for a pre-defined systems of interest (solar fan system) can be found in Appendix C.

3.1. The Sellers-Chell Method for Evaluating MBSE Tools

1. Form a team to do the evaluation (although this could be accomplished by a single individual, group participation is more likely to provide diverse viewpoints within an organization).
2. Select the tools to evaluate (a list of MBSE tools is given in Appendix A).
3. Select the systems engineering processes, resulting products, and the general supportability characteristics to be evaluated from the Tool Capabilities Inventory.
4. Discuss and assess preliminary weights for each product and characteristic chosen.
5. Select a system that is simple and well-known, or use the solar-powered fan system defined in Appendix C to create the systems engineering products with.

6. Create the chosen products using the tools. It is probably easiest but not necessary to do this in the following order.
    - Define high-level requirements
    - Add functions and their hierarchy
    - Add assets and their hierarchy
    - Add relationships within and between functions and assets
    - Create products
7. Once the products are created, use the Tool Usability Assessment to evaluate the tool's performance in creating the products but also the general supportability characteristics.
8. Input the scores into the Tool Capabilities Inventory.
9. Compare the ratings and discuss them.
10. Select your MBSE tool.

4. Results

The SCM was tested by using a simple model of a solar-powered fan and creating basic systems engineering products from the Tool Capabilities Inventory using five MBSE tools (Innoslate, CORE, Rhapsody, MagicDraw, Enterprise Architect). The performance of the tools was then evaluated with the Tool Usability Assessment, the scores can be found in Appendix C. It is very important to note that the objective of this exercise was NOT to find the best MBSE tool, but to generate some data points and experience to validate the SCM. Since the tester had previous experience using both Innoslate and CORE prior to this project, unsurprisingly, those two tools scored higher than the others on the TUA.

During this testing process, we found the Tool Capabilities Inventory used with the Usability Assessment produced consistent results. Innoslate (1.56) and CORE (1.38) both had relatively average scores while the three SysML-based tools were lower (MagicDraw 1.17, Rhapsody 0.83, Enterprise Architect 0.83). As mentioned before having previous experience with a certain tool will bias the TUA results. Looking at these scores, it should be noted that the tester had the most experience with Innoslate

with CORE coming in second. Furthermore, MagicDraw was the last SysML tool to be evaluated and by that point the tester had more experience with how to use SysML causing that score to be a bit higher than the other two.

The most important result gained from this testing is that the SCM is an easy-to-follow technique for evaluating an MBSE tool. Its ability to be customized recognizes that an individual or organization looking to implement MBSE has unique needs and that, at least for the moment, there isn't a tool that is clearly better than the others. The authors are confident that this method can help pick the right tool for the right job.

5. Conclusions

Since Model-Based Systems Engineering is a relatively new discipline, it has been difficult for individuals and organizations to find a good fit between needs of the organization and the tools available. It would be very useful to have a method of evaluating MBSE tools to make this process straightforward and to maximize the probability of finding a tool which can meet the unique performance requirements these individuals or organizations may have. By using ideas from previous research, the systems engineering practices as defined by NASA 7123.1B, and the Cooper-Harper Scale, this paper has proposed a method which can do that.

The Sellers-Chell Method, with its two main facets, the Tool Capabilities Inventory and the Tool Usability Assessment, is a way to evaluate MBSE tools both objectively and subjectively. The Tool Capabilities Inventory provides an objective evaluation of what a tool can and can't do. The Tool Usability Assessment allows evaluators to give subjective but repeatable scores to determine the how well a tool can meet both their functional and nonfunctional requirements. Since this method is able to evaluate tools over the entire systems engineering domain it can be tailored to the needs of anyone looking for an MBSE tool.

When the method was tested using five different MBSE tools it was found to be quite simple to use and to produce reasonably consistent results. Using the Sellers-Chell Method will help any individual or organization find tool for Model-Based Systems Engineering that fits their needs in a straightforward manner.

6. Future Work

The Sellers-Chell Method presented here represents a first effort toward a standardized technique for evaluating and selecting an MBSE tool. As a first effort, there is considerable opportunity for future work to test and improve this approach. With further use, there are many ways for it to be refined. A great way to do this would be to get groups of volunteers with similar levels of experience to evaluate MBSE tools using this method. Gathering more data in this way will help to validate the consistency of the method and provide information on whatever changes could make it more helpful to anyone that wishes to evaluate an MBSE tool. More data could also help the Tool Capabilities Inventory by finding the best way to assess weights or perhaps adjust the scores of 0, 1, and 2. Furthermore the Tool Usability Assessment was written to follow the Cooper-Harper Scale as closely as possible, more data might find that the wording of the decision tree needs to be changed.

7. References

1. Bailey, R. E., Jackson, E. B., Bilmoria, K. D., Mueller, E. R., Frost, C. R., Alderete, T. S. (2009). "Cooper-Harper Experience Report for Spacecraft Handling Qualities Applications" NASA/TM-2009-215767

2. Cloutier, R., Bone, M. (2010). "Compilation of SysML RFI- Final Report" Systems Modeling Language (SysML) Request For Information OMG Document: syseng/2009-06-01

3. Cooper, G.E., and Harper, Jr., R.P. (1969). "The Use of Pilot Rating in the Evaluation of Aircraft Handling Qualities" NASA TN D-5153

4. Cummings, M.L., Myers, K., Scott, S. D., (2010). "Modified Cooper Harper Evaluation Tool for Unmanned Vehicle Displays" PerMIS '10 Proceedings of the 10th Performance Metrics for Intelligent Systems Workshop Pages 235-242

5. De Jong, K., de Spain, M. J., Hernandez, M. E., Post, D. S., Taylor, J. L. (2011). "Process for Selecting Engineering Tools – Applied to Selecting a SysML Tool" SANDIA REPORT SAND 2010-7098

6. Friedenthal, S., Moore, A., Steiner, R. (2009). OMG Systems Modeling Language (OMG SysML™) Tutorial Object Management Group INCOSE

7. Finance, G. (2010). [online] "SysML Modelling Language Explained" http://www.omgsysml.org/SysML_Modelling_Language_explained-finance.pdf

8. Harper, R. P. Jr., Cooper, G. E. (1984). "Handling Qualities and Pilot Evaluation" Journal of Guidance, Control, and Dynamics, Vol. 9, No. 5, pp. 515-529.

9. Hart, L. E. (2015). Introduction to Model-Based Systems Engineering (MBSE) and SysML Delaware Valley INCOSE Chapter Meeting July 30, 2015

10. Hill, S. G., Lavecchia, H. P., Byers, J. C., Bittner, A. C., Zaklad, A. L., & Christ, R. E. (1992). Comparison of 4 Subjective Workload Rating-Scales. Human Factors, 34(4), 429-439

11. Land, R., Blankers, L., Chaudron, M., Crnković, I (2008). "COTS Selection Best Practices in Literature and in Industry" High Confidence Software Reuse in Large Systems, pp. 100-111

12. Lin, H., Lai, A., Ullrich, R. Kuca, M., Shaffer-Grant, J. Pacheco, S. Dalton, K., McClelland, K. Watkins, W., Khajenoori, S. (2006). "COTS Software Selection Process" Sandia Report, SAND2006-0478

13. NASA Procedural Requirements Document 7123.1B (2013). Systems Engineering Processes and Requirements. Washington, DC

14. NASA Systems Engineering Handbook NASA/SP-2007-6105 Rev1 (2007)

15. Nielsen, J. (2001). "Usability Metrics" Nielsen Norman Group https://www.nngroup.com/articles/usability-metrics/

16. Rasheed, M., Anwar, M. W. (2016). "A Systematic Investigation of Tools in Model Based System Engineering for Embedded Systems" System of Systems Engineering Conference (SoSE), 2016 11th

17. Sellers, J. J. (2016). Applied MBSE Chapter to be Published

18. Sparx Systems, Enterprise Architect 12 Reviewer's Guide, 2015, http://www.sparxsystems.com/downloads/whitepapers/EAReviewersGuide.pdf

19. Steiner, R. (2008). Deployment of SysML in Tools and Architectures: an Industry Perspective INCOSE SysML information Days

20. SysMl.tools Editor (2013). [online] "How to Define SysML Tool Evaluation Criteria for MBSE" http://sysml.tools/define-sysml-tool-evaluation-criteria/ PivotPoint Technology Corp.

21. Towers, J. (2015). “What is Model Based Systems Engineering” Z9 Issue. 2.0 INCOSE UK Leaflet

8. Appendix A

Following is a list of tools that can be used for MBSE. It is not meant to be exhaustive but contains most of the popular MBSE and SysML tools.

8.1. Microsoft Office Suite

https://www.office.com/

While Microsoft Office almost certainly doesn't need introduction it should be mentioned as the most popular tool with which to do systems engineering. Within the suite Excel is very commonly used for requirements management, Word for documents, and PowerPoint for diagrams.

8.2. IBM Rational Rhapsody Family

http://www-03.ibm.com/software/products/en/ratirhapfami

According to their website:

IBM® Rational® Rhapsody® (with Design Manager) is a proven solution for modeling and design activities. The family provides a collaborative design, development and test environment for systems engineers and software engineers.

The software supports UML, SysML and AUTOSAR, allows for control of defense frameworks (DoDAF, MODAF and UPDM) and complies with standards such as DO-178, ISO 26262 and more.

8.3. Visual Paradigm

https://www.visual-paradigm.com/features/

According to their website:

Visual Paradigm supports SysML and contains these features, according to their website:

Visual Paradigm features all the UML diagrams and ERD tools essentially in system and database design. Innovative modeling tools like Resource Catalog, Transitor and Nicknamer makes system modeling easy and cost-effective. Doc. Composer lets you produce detailed design specification ready to use in discussion with just a few clicks.

8.4. Atego Artisan Studio

http://www.atego.com/products/sysim/

According to their website:

Artisan Studio SySim™ is an innovative solution to enable early behavioral simulation of complex OMG SysML systems design. It transforms SysML designs into executable applications allowing you to check, run, observe and refine them. Artisan Studio SySim is fully model driven: architecture, behavior, simulation, definition and visualization are all achieved with SysML models, with no need to focus on lower level details such as code generation or target environments.

8.5. Astah SysML

http://astah.net/editions/sysml

According to their website:

SysML is already the industry standard for engineering large, complex systems; more safety and quality system development teams are rapidly adopting the language every day. Now every engineering specialty, from mechanical, to electronic to software engineering have a common language to design safe, complex systems architectures. We are pleased to welcome Astah to the fold, a unique modeling tool from Japan, from leading OMG member Change Vision.

8.6. Vitech CORE/GENESYS

8.6.1. CORE

http://www.vitechcorp.com/products/core.shtml

According to their website:

Using CORE allows you to:

- integrate requirements management to ensure that you capture customer needs accurately
- identify system functionality, complete system behavior analysis, and simulate system performance
- develop and trace system architecture from system to subsystems and component levels
- provide traceability from system design to Validation and Verification plans and procedures
- automatically produce system design documentation directly from the design repository to support team and customer review of the design progress

8.6.2. GENESYS

According to their website:

http://www.vitechcorp.com/products/genesys.shtml

Built upon insights from over 20 years with CORE, GENESYS is Vitech's next generation software delivering connected systems engineering across the enterprise. Reengineered from the ground up, GENESYS implements proven model-centric approaches leveraging modern technologies in a completely open architecture. The result is the power of a full MBSE environment with the usability of modern office tools integrated with your desktop, engineering, and enterprise environments to deliver your data your way.

8.7. 3SL Cradle

https://www.threesl.com/

According to their website:

Extend requirements and test management with model based systems engineering (MBSE). Build models using your choices of SysML, SASD, UML, ADARTS, architecture, eFFBD, IDEF and process notations. Reuse, share and link models for product variants and builds, system-of-systems architectures and agile sprints. Link to requirements, tests, issues, defects and all other information. Get traceability across your entire process, in one tool.

8.8. Sparx Systems Enterprise Architect

According to the Sparx Systems Enterprise Architect Reviewer's Guide:

Enterprise Architect is a visual platform for designing and constructing software systems, for business process modeling, and for more generalized modeling purposes. Enterprise Architect is based on the latest UML® 2.5 specification. UML defines a visual language that is used to model a particular domain or system (either proposed or existing). Enterprise Architect is a progressive tool that covers all aspects of the development cycle, providing full traceability from the initial design phase through to deployment, maintenance, testing and change control.

8.9. Spec Innovations Innoslate

https://www.innoslate.com/systems-engineering/

According to their website:

Use Innoslate to model both the behavioral and physical aspects of a system. The clean interface, simple relationships, and modern looking diagram visualizations make managing model entities easier than ever.

Currently, there are over 9 different diagrams to visualize behavioral models including: executable Action, Sequence, N-squared, and IDEF0. Physical models have 8 different diagrams including: Asset, Class, Use Case, and Organization Chart. All of the diagrams are drag droppable, allowing for quick model design and construction. The diagrams conform to the LML, SysML, or the IDEF0 standard.

8.10. No Magic MagicDraw

http://www.nomagic.com/products/magicdraw.html

According to their website:

MagicDraw is the award-winning business process, architecture, software and system modeling tool with teamwork support. Designed for Business Analysts, Software Analysts, Programmers, QA Engineers, and Documentation Writers, this dynamic and versatile development tool facilitates analysis and design of Object Oriented (OO) systems and databases. It provides the industry's best code engineering mechanism (with full round-trip support for Java, C++, C#, CL (MSIL) and CORBA IDL programming languages), as well as database schema modeling, DDL generation and reverse engineering facilities.

8.11. Modeliosoft Modelio

https://www.modeliosoft.com/en/products/modelio-sa-sysml.html

According to their website:

"The full range of **SysML diagrams** is supported, from block diagrams to internal and parametric diagrams." Also, "SysML provides **diagrams to model requirements**. With the dedicated Analyst module, architects can then organize requirements using groups, hierarchy and dependencies, and edit their properties using either the built-in **spreadsheet editor** or **MS Excel™**." And, "The Analyst module can be used in conjunction with **Rational Doors™**, or coupled to other tools if necessary."

8.12. Eclipse Papyrus

https://eclipse.org/papyrus/

According to their website:

Implemented standards: UML 2.5.0, SysML 1.1 & 1.4, OCL 2.3.1, fUML 1.1, ALF 1.0.1, MARTE 1.1 (incubation), EAST-ADL (incubation), RobotML (incubation), UML-RT (incubation) and ISO/IEC 42010.

Papyrus is an industrial-grade open source Model-Based Engineering tool. Papyrus has notably been used successfuly in industrial projects and is the base platform for several industrial modeling tools.

8.13. PTC Model-Based Systems Engineering Solution

http://www.ptc.com/model-based-systems-engineering

According to their website:

> [PTC Model-Based Systems Engineering Solution] includes all the capabilities you need to bring focus and rigor to your systems engineering program, including model-based systems engineering, model validation, product line engineering, an asset library for modular, systems of systems design, and model simulation to validate design ideas earlier in the product lifecycle. Built on a multi-user database for live collaboration, this industry-leading MBSE platform is capable of modeling software, systems, and product lines in a single toolset.

8.14. Altova Umodel

https://www.altova.com/umodel.html

UModel SysML Modeling Features:

- Supports all nine SysML diagrams
- Enables hyperlinks between diagrams, supporting documents, or Web pages
- Shared subprojects for team collaboration or reuse
- Elements can be assigned to diagram layers and selectively viewed or hidden
- Unlimited undo/redo encourages exploring new ideas
- XMI model interchange with other SysML tools
- SysML diagrams integrated with UML modeling for robust coverage of software project requirements
- SysML diagrams and elements included in automated project documentation

8.15. ANSYS SCADE Suite

http://www.ansys.com/products/embedded-software/ansys-scade-suite

According to their website:

> ANSYS SCADE Suite empowers users with a Model-Based Development Environment for critical embedded software. With native integration of the formally-defined SCADE language, SCADE Suite is the

integrated design environment for critical applications spanning requirements management, model-based design, simulation, verification, qualifiable/certified code generation, and interoperability with other development tools and platforms.

9. Appendix B

Below is a list of the 17 systems engineering processes split up by their phases, the descriptions are their purposes as given by NASA 7123.1B.

System Design Processes

9.1. Stakeholder Expectations Definition

The stakeholder expectations definition process is used to elicit and define use cases, scenarios, concept of operations, and stakeholder expectations for the applicable product life-cycle phases and product layer. The baselined stakeholder expectations are used for validation of the product layer end product during product realization.

9.2. Technical Requirements Definition

The technical requirements definition process is used to transform the baselined stakeholder expectations into unique, quantitative, and measurable technical requirements expressed as "shall" statements that can be used for defining a design solution definition for the end product and related enabling products of this layer.

9.3. Logical Decomposition

The logical decomposition process is used to improve understanding of the defined technical requirements and the relationships among the requirements (e.g., functional, behavioral, performance, and temporal) and to transform the defined set of technical requirements into a set of logical decomposition models and their associated set of derived technical requirements for lower levels of the system and for input to the design solution definition process.

9.4. Design Solution Definition

The design solution definition process is used to translate the outputs of the logical decomposition process into a design solution definition that is in a form consistent with the product life-cycle phase and product layer location in the system structure and that will satisfy phase exit criteria. This includes transforming the defined logical decomposition models and their associated sets of derived technical requirements into alternative solutions, then analyzing each alternative to be able to select a preferred

alternative and fully define that alternative into a final design solution that will satisfy the technical requirements. These design solution definitions will be used for generating end products either by using the product implementation process or product integration process as a function of the position of the product layer in the system structure and whether there are additional subsystems of the end product that need to be defined. The output definitions from the design solution (end product specifications) will be used for conducting product verification.

Product Realization Processes

9.5. Product Implementation

The product implementation process is used to generate a specified product of a product layer through buying, making, or reusing in a form consistent with the product life-cycle phase exit criteria and that satisfies the design solution definition specified requirements (e.g., drawings, specifications). Product Implementation

9.6. Product Integration

The product integration process is used to transform the design solution definition into the desired end product of the product layer through assembly and integration of lower level validated end products in a form consistent with the product life-cycle phase exit criteria and that satisfies the design solution definition requirements (e.g., drawings, specifications).

9.7. Product Verification

The product verification process is used to demonstrate that an end product generated from product implementation or product integration conforms to its design solution definition requirements as a function of the product life-cycle phase and the location of the product layer end product in the system structure. Special attention is given to demonstrating satisfaction of the MOPs defined for each MOE during conduct of the technical requirements definition process.

9.8. Product Validation

The product validation process is used to confirm that a verified end product generated by product implementation or product integration fulfills (satisfies) its intended use when placed in its intended

environment and to assure that any anomalies discovered during validation are appropriately resolved prior to delivery of the product (if validation is done by the supplier of the product) or prior to integration with other products into a higher level assembled product (if validation is done by the receiver of the product). The validation is done against the set of baselined stakeholder expectations. Special attention should be given to demonstrating satisfaction of the MOEs identified during conduct of the stakeholder expectations definition process. The type of product validation is a function of the form of the product, product life-cycle phase, and applicable customer agreement.

9.9. Product Transition

The product transition process is used to transition to the customer at the next level in the system structure a verified and validated end product that has been generated by product implementation or product integration for integration into an end product. For the top level end product, the transition is to the intended end user. The form of the product transitioned will be a function of the product life-cycle phase exit criteria and the location within the system structure of the product layer in which the end product exists.

Technical Management Processes

9.10. Technical Planning

The technical planning process is used to plan for the application and management of each common technical process. It is also used to identify, define, and plan the technical effort applicable to the product life-cycle phase for the product layer location within the system structure and to meet project objectives and product life-cycle phase exit criteria. A key document generated by this process is the SEMP.

9.11. Requirements Management

The requirements management process is used to:

a. Manage the product requirements identified, baselined, and used in the definition of the products of this layer during system design;

b. Provide bidirectional traceability back to the top product layer requirements; and

c. Manage the changes to established requirement baselines over the life cycle of the system products.

9.12. Interface Management

The interface management process is used to:

a. Establish and use formal interface management to assist in controlling system product development efforts when the efforts are divided between Government programs, contractors, and/or geographically diverse technical teams within the same program or project.

b. Maintain interface definition and compliance among the end products and enabling products that compose the system as well as with other systems with which the end products and enabling products must interoperate.

Note: A less formal interface management approach can be used in conjunction with requirements management and/or configuration management process activities when the technical effort is co-located in the same project.

9.13. Technical Risk Management

The technical risk management process is used to examine on a continuing basis the risks of technical deviations from program/project plans and to identify potential problems before they occur. Risk management is performed across the life of the program.

9.14. Configuration Management

The configuration management process for end products, enabling products, and other work products placed under configuration control is used to:

a. Identify the configuration of the product or work product at various points in time;

b. Systematically control changes to the configuration of the product or work product;

c. Maintain the integrity and traceability of the configuration of the product or work product throughout its life; and

d. Preserve the records of the product or end product configuration throughout its life cycle, disposing them in accordance with NPR 1441.1, NASA Records Retention Schedules.

9.15. Technical Data Management

The technical data management process is used to:

a. Provide the basis for identifying and controlling data requirements;

b. Responsively and economically acquire, access, and distribute data needed to develop, manage, operate, and support system products over their product life;

c. Manage and dispose data as records;

d. Analyze data use;

e. If any of the technical effort is performed by an external contractor, obtain technical data feedback for managing the contracted technical effort; and

f. Assess the collection of appropriate technical data and information.

g. Effectively manage authoritative data that defines, describes, analyzes, and characterizes a product life cycle.

h. Ensure consistent, repeatable use of effective PDLM processes, best practices, interoperability approaches, methodologies, and traceability.

i. Ensure product data accessibility and availability, including a method to archive the data.

9.16. Technical Assessment

The technical assessment process is used to help monitor progress of the technical effort and provide status information for support of the system design, product realization, and technical management processes.

9.17. Decision Analysis

The decision analysis process, including processes for identification of decision criteria, identification of alternatives, analysis of alternatives, and alternative selection, is applied to technical issues to support their resolution. It considers relevant data (e.g., engineering performance, quality, and reliability) and associated uncertainties. This process is used throughout the system life cycle to evaluate the impact of decisions on health and safety, technical, cost, and schedule performance. NASA/SP-2010-576, NASA Risk-informed Decision Making Handbook provides guidance for analyzing decision alternatives in a risk-informed fashion.

10. Appendix C

10.1. Solar Fan Testers Handout

In order to create this model within the tool, one can follow the steps below. It is probably easiest but not necessary to do them in order.

- Define high-level requirements
- Add functions and their hierarchy
    - Note that the tools sometimes use words other than function, i.e. action
- Add assets and their hierarchy (i.e. component)
- Allocate functions to assets
- Create interfaces (i.e. conduit, connection)
- Add inputs/outputs
- Allocate inputs/outputs to interfaces
- Create a System Definition Report (products may include the following)
    - To Capture Requirements
        - SysML Requirements Diagram (Figure 1, on page 4)
        - Requirements Document
    - To Capture Functional Architecture
        - SysML Activity Diagram (Figure 2)
        - NxN, IDEF0, Functional Flow Block Diagram
    - To Capture Physical Architecture
        - SysML Block Definition Diagram (Figure 3)
        - Asset Diagram, Context Diagram
    - To Capture Interfaces and I/O
        - SysML Block Definition Diagram (Figure 3)
        - Spider Diagram, IxI
- Rate how well the tool performed at creating the products using the modified Cooper-Harper rating scale found in Figure 6.
- Rate how well the tool provided and performed other features (could include the following)
    - Ability to automatically generate code
    - Add pictures to reports
    - Ease of exporting models to MS Office
    - Etc.

On the following pages you will find tables describing the solar fan system architecture. Also, Figures 1-3 show the SysML diagrams needed to describe this system. Figures 4-6 show other SysML diagrams that could be made. The modified Cooper-Harper Scale is on page 6 and page 7 has the rating sheet.

Table 1: List of Entities for solar fan system

Functions	Assets
F0.0 Perform System Functions	S0.0 Solar Fan System of Systems
F1.0 Provide Sunlight	S1.0 Solar Fan Sytem
F2.0 Convert Sunlight to Electricity	S1.1 Solar Array
F3.0 Produce Torque	S1.2 Motor/Fan Assembly
F4.0 Produce Airflow	S1.2.1 Motor Segment
	S1.2.2 Propeller
	S2.0 The Sun
	S3.0 User

Interfaces	Inputs/Outputs
I/F1.0 Sun to System Interface	I/O1.0 Sunlight
I/F2.0 Array to Motor Connector	I/O2.0 DC Electricty
	I/O3.0 Torque
	I/O4.0 Air Flow

Requirements
R1.0 System Requirements
R1.1 The system shall produce cooling airflow
R1.2 The system shall be powered by solar energy alone.
R1.3 The system shall be hand held.

Table 2: Functional Decomposition of Solar Fan System

Function	Generates	Performed By	Decomposed By	Decomposes
F0.0 Perform System Functions		S1.0 Solar Fan System	F1.0 Provide Sunlight F2.0 Convert Sunlight to Electricity F3.0 Produce Torque F4.0 Produce Airflow	
F1.0 Provide Sunlight	I/O1 Sunlight	S2.0 The Sun		F0.0 Perform System Functions
F2.0 Convert Sunlight to Electricity	I/O2 Electricity	S1.1 Solar Array		F0.0 Perform System Functions
F3.0 Produce Torque	I/O3 Torque	S1.2.1 Motor Segment		F0.0 Perform System Functions
F4.0 Produce Air Flow	I/O4 Air Flow	S1.2.2 Propeller		F0.0 Perform System Functions
F5.0 Cool Off		3.0 User		F0.0 Perform System Functions

Table 3: Physical (Asset) Decomposition of Solar Fan System

Asset	Decomposes	Decomposed By	Performs	Satisfies
S0.0 Solar Fan System of Systems		S1.0 Solar Fan System S2.0 The Sun		
S1.0 Solar Fan System	S0.0 Solar Fan System of Systems	S1.1 Solar Array S1.2 Motor/Fan Assembly	F0.0 Perform System Functions	R1.0 R1.3
S1.1 Solar Array	S1.0 Solar Fan System		F2.0 Convert Sunlight to Electricity	R1.2
S1.2 Motor/Fan Assembly	S1.0 Solar Fan System	S1.2.1 Motor Segment S1.2.2 Propeller		R1.1
S1.2.1 Motor Segment	S1.2 Motor/Fan Assembly		F3.0 Produce Torque	
S1.2.2 Propeller	S1.2 Motor/Fan Assembly		F4.0 Produce Airflow	
S2.0 The Sun	S0.0 Solar Fan System of Systems		F1.0 Provide Sunlight	
S3.0 User	S0.0 Solar Fan System of Systems		F5.0 Cool Off	

Table 4: Interfaces for Solar Fan System

Interface	Connects To	Transfers
I/F1.0 Sun to System	S2.0 The Sun S1.1 Solar Array	I/O1.0 Sunlight I/O3.0 Torque
I/F2.0 Array to Motor Connector	S1.1 Solar Array S1.2 Motor/Fan Assembly	I/O2.0 DC Electricity

Table 5: I/O for Solar Fan System

Input/Output	Generated By	Received By	Transferred By
I/O1.0 Sunlight	F1.0 Provide Sunlight	F2.0 Convert Sunlight to Electricity	I/F1.0 Sun to System Interface
I/O2.0 DC Electricit	F2.0 Covert Sunlight to Electricity	F3.0 Produce Torque	I/F2.0 Array to Motor Connector
I/O3.0 Torque	F3.0 Produce Torque	F4.0 Produce Air Flow	
I/O4.0 Air Flow	F4.0 Produce Air Flow	S3.0 User	

Figure 1: SysML Requirements Diagram

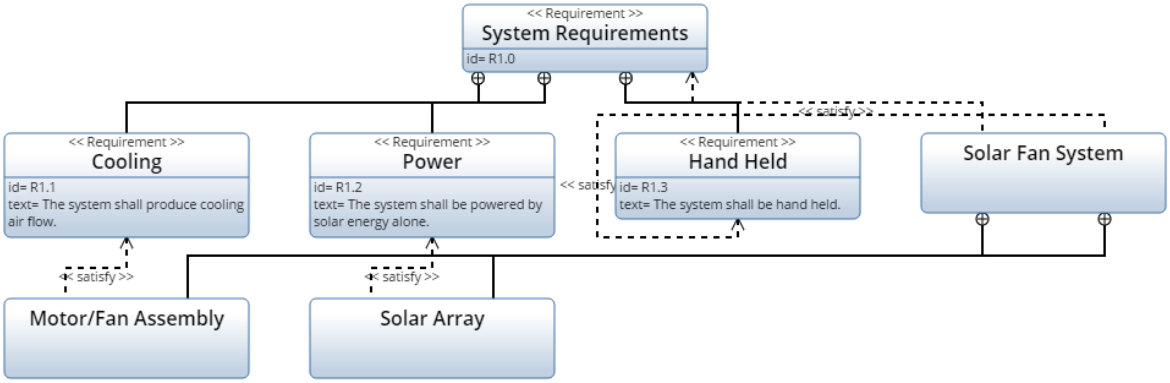


Figure 2: SysML Activity Diagram

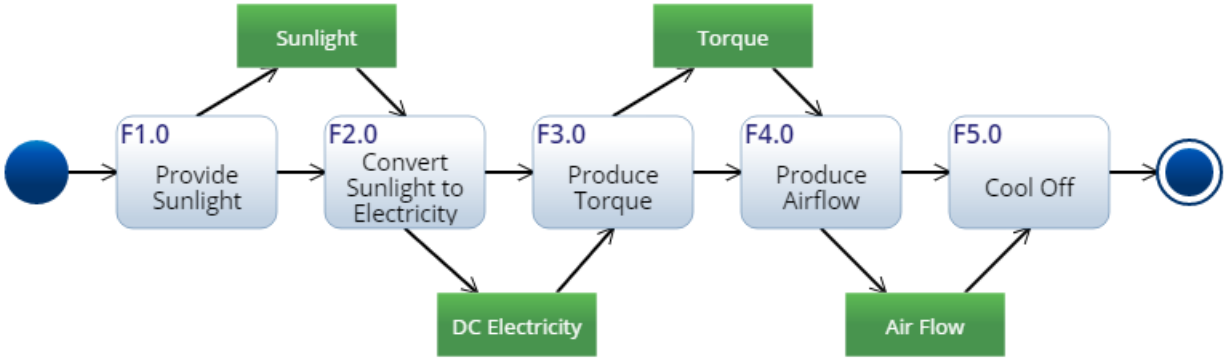


Figure 3: SysML Block Definition Diagram

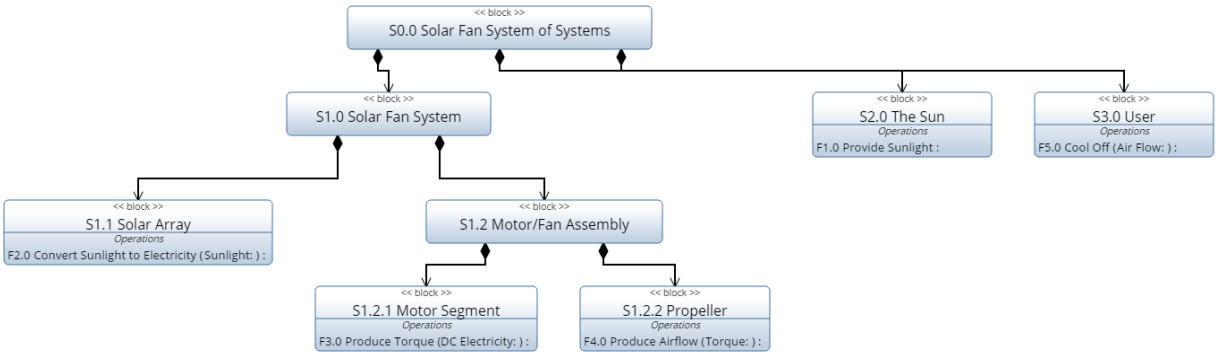


Figure 4: SysML Sequence Diagram

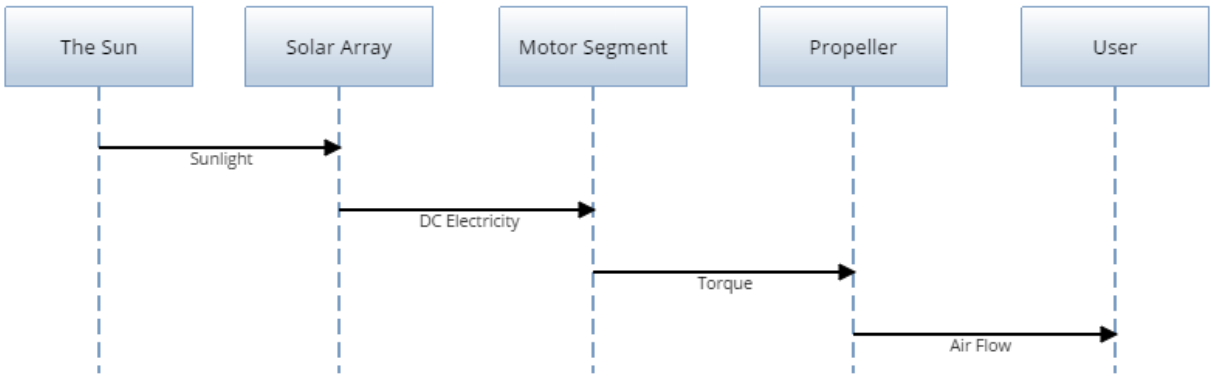
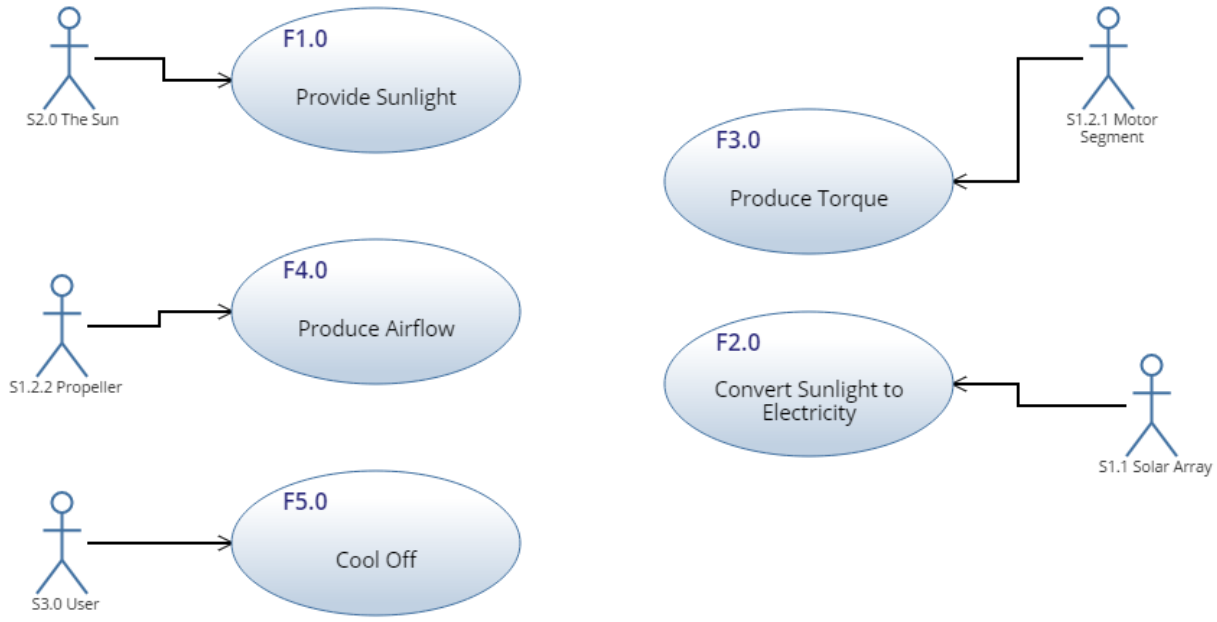


Figure 5: SysML Use Case Diagram



11. Appendix D

Testing Scores

Innoslate

Tool Capabilities Inventory		
Design		
Can the tool make these products?		
2. Technical Requirements Definition	**Weight**	**0, 1, 2**
Requirements Document	1	2
SysML Requirements Diagram	1	1
3. Logical Decomposition	**Weight**	**0, 1, 2**
Context Diagram	1	2
SysML Activity Diagram	1	2
SysML Block Definition Diagram	1	1
NxN Diagram	1	1
Use Case Diagrams	1	2
IDEF0 Diagram	1	1
Sequence Diagram	1	2

CORE

Tool Capabilities Inventory		
Design		
Can the tool make these products?		
2. Technical Requirements Definition	**Weight**	**0, 1, 2**
Requirements Document	0	x
SysML Requirements Diagram	1	1
3. Logical Decomposition	**Weight**	**0, 1, 2**
Context Diagram	1	1
SysML Activity Diagram	1	2
SysML Block Definition Diagram	1	1
NxN Diagram	1	2
Use Case Diagrams	1	1
IDEF0 Diagram	1	2
Sequence Diagram	1	1

Rhapsody

Tool Capabilities Inventory		
Design		
Can the tool make these products?		
2. Technical Requirements Definition	**Weight**	**0, 1, 2**
Requirements Document	0	x
SysML Requirements Diagram	1	0
3. Logical Decomposition	**Weight**	**0, 1, 2**
Context Diagram	1	0
SysML Activity Diagram	1	1
SysML Block Definition Diagram	1	0
NxN Diagram	0	x
Use Case Diagrams	1	2
IDEF0 Diagram	0	x
Sequence Diagram	1	2

MagicDraw

Tool Capabilities Inventory		
Design		
Can the tool make these products?		
2. Technical Requirements Definition	**Weight**	**0, 1, 2**
Requirements Document	0	x
SysML Requirements Diagram	1	1
3. Logical Decomposition	**Weight**	**0, 1, 2**
Context Diagram	1	1
SysML Activity Diagram	1	2
SysML Block Definition Diagram	1	1
NxN Diagram	0	x
Use Case Diagrams	1	2
IDEF0 Diagram	0	x
Sequence Diagram	1	0

Enterprise Architect

Tool Capabilities Inventory		
Design		
Can the tool make these products?		
2. Technical Requirements Definition	**Weight**	**0, 1, 2**
Requirements Document	0	x
SysML Requirements Diagram	1	1
3. Logical Decomposition	**Weight**	**0, 1, 2**
Context Diagram	1	0
SysML Activity Diagram	1	1
SysML Block Definition Diagram	1	0
NxN Diagram	0	x
Use Case Diagrams	1	1
IDEF0 Diagram	0	x
Sequence Diagram	1	2