2019-12

ASSESSING THE USABILITY OF MONTEREY PHOENIX SOFTWARE

Gramp, Timothy

Monterey, CA; Naval Postgraduate School

NAVAL POSTGRADUATE SCHOOL

MONTEREY, CALIFORNIA

THESIS

ASSESSING THE USABILITY OF MONTEREY PHOENIX SOFTWARE

by

Timothy Gramp

December 2019

Thesis Advisor: Kristin M. Giammarco
Second Reader: Robert Semmens

THIS PAGE INTENTIONALLY LEFT BLANK

REPORT DOCUMENTATION PAGE			*Form Approved OMB No. 0704-0188*

1. AGENCY USE ONLY *(Leave blank)*	**2. REPORT DATE** December 2019	**3. REPORT TYPE AND DATES COVERED** Master's thesis

4. TITLE AND SUBTITLE ASSESSING THE USABILITY OF MONTEREY PHOENIX SOFTWARE	**5. FUNDING NUMBERS**
6. AUTHOR(S) Timothy Gramp	
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Naval Postgraduate School Monterey, CA 93943-5000	**8. PERFORMING ORGANIZATION REPORT NUMBER**
9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES) N/A	**10. SPONSORING / MONITORING AGENCY REPORT NUMBER**

11. SUPPLEMENTARY NOTES The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.

12a. DISTRIBUTION / AVAILABILITY STATEMENT Approved for public release. Distribution is unlimited.	**12b. DISTRIBUTION CODE** A

13. ABSTRACT (maximum 200 words)

The current and future system engineer's toolkit increasingly contains complex software applications to aid in the defining and designing of systems. This thesis presents an examination of the software usability reported by a group of DOD military and civilian personnel who were exposed to one such application, the Monterey Phoenix (MP) modeling tool, for the first time. The objective of this analysis is to test the hypothesis that, among new users, those with prior modeling background will report a better MP user experience and better software usability than those without prior modeling background.

Naval Postgraduate School students and faculty who responded to an invitation to participate in an ONR-sponsored, usability study completed a carefully constructed protocol designed to identify modeling experience level, provide a brief introductory tutorial to the MP software, present participants with a simple modeling task using MP, and capture feedback from users at the completion of the protocol. Participant feedback produced data for use in the calculation of System Usability Scale scores, Net Promoter Scores, and the NASA Task Load Index, which are established measures of usability. Finally, a Monterey Phoenix subject-matter expert developed and applied a grading rubric in the evaluation of MP models that study participants produced.

14. SUBJECT TERMS software development, MBSE, software usability, system engineering tools	**15. NUMBER OF PAGES** 89
	16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	**18. SECURITY CLASSIFICATION OF THIS PAGE**	**19. SECURITY CLASSIFICATION OF ABSTRACT**	**20. LIMITATION OF ABSTRACT**
Unclassified	Unclassified	Unclassified	UU

NSN 7540-01-280-5500 Standard Form 298 (Rev. 2-89) Prescribed by ANSI Std. 239-18

THIS PAGE INTENTIONALLY LEFT BLANK

Approved for public release. Distribution is unlimited.

ASSESSING THE USABILITY OF MONTEREY PHOENIX SOFTWARE

Timothy Gramp
Civilian, Department of the Navy
BS, Geneva College, 1999

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN SYSTEMS ENGINEERING MANAGEMENT

from the

NAVAL POSTGRADUATE SCHOOL
December 2019

Approved by: Kristin M. Giammarco
Advisor

Robert Semmens
Second Reader

Ronald E. Giachetti
Chair, Department of Systems Engineering

THIS PAGE INTENTIONALLY LEFT BLANK

ABSTRACT

The current and future system engineer's toolkit increasingly contains complex software applications to aid in the defining and designing of systems. This thesis presents an examination of the software usability reported by a group of DOD military and civilian personnel who were exposed to one such application, the Monterey Phoenix (MP) modeling tool, for the first time. The objective of this analysis is to test the hypothesis that, among new users, those with prior modeling background will report a better MP user experience and better software usability than those without prior modeling background.

Naval Postgraduate School students and faculty who responded to an invitation to participate in an ONR-sponsored, usability study completed a carefully constructed protocol designed to identify modeling experience level, provide a brief introductory tutorial to the MP software, present participants with a simple modeling task using MP, and capture feedback from users at the completion of the protocol. Participant feedback produced data for use in the calculation of System Usability Scale scores, Net Promoter Scores, and the NASA Task Load Index, which are established measures of usability. Finally, a Monterey Phoenix subject-matter expert developed and applied a grading rubric in the evaluation of MP models that study participants produced.

THIS PAGE INTENTIONALLY LEFT BLANK

TABLE OF CONTENTS

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF FIGURES

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF TABLES

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF ACRONYMS AND ABBREVIATIONS

CP	concurrent probing
CTA	concurrent think aloud
DOD	Department of Defense
HSR	human subject research
INCOSE	International Council on Systems Engineering
ISO	International Organization for Standardization
M&S	modeling and simulation
MBSE	model-based system engineering
MP	Monterey Phoenix
NASA	National Aeronautics and Space Administration
NASA-TLX	NASA task load index
NAVSEA	Naval Sea Systems Command
NPS	Naval Postgraduate School
NPS	net promoter score
ONR	Office of Naval Research
RP	retrospective probing
RTA	retrospective think aloud
RTLX	raw task load index (NASA)
SE	system engineering
SOS	system of systems
SUS	system usability scale
UAV	unmanned aerial vehicle
UCD	user-centered design

THIS PAGE INTENTIONALLY LEFT BLANK

EXECUTIVE SUMMARY

This thesis presents an examination of the software usability reported by a group of DOD military and civilian personnel who were exposed to the Monterey Phoenix (MP) modeling tool for the first time. The objective of the analysis was to test the hypothesis that, among these new users, those with prior modeling background will report a better MP user experience and better software usability than those without prior modeling background.

Naval Postgraduate School students and faculty who responded to an invitation to participate in an ONR-sponsored usability study completed a carefully-constructed protocol designed to identify modeling experience level, provide a brief introductory tutorial to the MP software, present participants with a simple modeling task using MP, and capture feedback from users at the completion of the protocol. The study team developed an exit survey to structure this feedback and produce information for use in the calculation of System Usability Scale scores, Net Promoter Scores, and the NASA Task Load Index, which are established measures of usability. Finally, a MP subject matter expert developed and applied a grading rubric in the evaluation of MP models that study participants produced.

Analysis of the limited data gathered during the study suggests that the more experienced system modelers reported greater satisfaction and better usability with MP than modelers without prior experience. The analysis included higher SUS averages and Net Promoter Scores for the experienced group. Further, the group of experienced modelers reported a lower workload as measured by the raw NASA TLX average. Additionally, the inexperienced modelers spent more time during the modeling task referring back to the tutorial than did their more experienced counterparts. Lastly, evaluation of the models created by both groups yielded a slightly greater performance score for the experienced modelers.

Additional analysis of the data gathered appears to confirm the intuitive assessment that users with programming comfort and skills also report a greater level of usability with the text-based MP coding interface.

Given the relatively small sample size (13) of the users represented, the study team could not conduct extensive, statistical analysis of the data gathered. Even the comparisons previously presented must be viewed with the awareness of this very limited sample size and the impacts that may have on the findings. At the time of this writing, several additional respondents had indicated an interest in completing the study, so these results should be included in a final analysis to compare with these findings.

ACKNOWLEDGMENTS

Thanks to my thesis advisor, Dr. Kristin Giammarco, for her helpful feedback and guidance throughout this process. Thanks also to Ross Eldred for creating the surveys and managing the communications with the participants during the study; his curation of the study data allowed me to focus my energy on the analysis.

THIS PAGE INTENTIONALLY LEFT BLANK

I. INTRODUCTION

A. PURPOSE

This thesis intends to study the relationship between the different levels of architecture modeling experience possessed by new Monterey Phoenix (MP) users and the satisfaction of those users with their exercise of the tool. The research method will employ a software usability study to test the following hypothesis: DOD military and civilian personnel with prior modeling background report having a better MP user experience than those without prior modeling background. To quantify user experience, the study examines both the feedback of the users (qualitative) as well as their objectively-scored performance (quantitative) in completing a simple modeling task with the software.

B. OBJECTIVE

Through providing a greater understanding of the relationship between past modeling experience and MP ease of use, this thesis analysis and its underlying usability study expect to drive the targeted MP user base within the DOD. This study also supports recommendations for future enhancements to the MP software as well as the development of focused training materials designed to address the unique challenges encountered by users with different levels of architecture modeling experience.

C. PROBLEM STATEMENT

Monterey Phoenix [1–3] is a "Navy-developed formal language and approach for modeling systems, software, hardware, people, organizational, and/or environmental behaviors and their interactions with one another" (Giammarco 2018).

Prior to this study, many who are familiar with MP assumed that users with prior modeling or software development experience find MP easier to learn and use. According to NPS associate professor, Dr. Kristin Giammarco, the body of student work seen in an academic setting seems to challenge the notion that prior modeling experience is a prerequisite to a positive and productive user experience with MP modeling. In fact, the possibility exists that seasoned modelers may report less satisfaction with MP use due to

learned behaviors and preconceived patterns of thought. An analogy exists in the sports world.

> It's a common misconception that athletes can easily transition to the links [golf]. The truth is a little more complicated. While athletes have a leg up on some of the mental and physical aspects of the game, golf remains a fundamentally different sport from any other. In fact, sometimes the very athleticism that makes a pro athlete so good at their own sport is a hindrance on the golf course. (Leadbetter, 2015)

However, beyond anecdotal evidence, developers and sponsors of MP currently lack the needed understanding of the relationship between past modeling experience and software ease of use.

D. SCOPE AND LIMITATIONS

While the study described in this thesis utilizes software usability testing methods, the primary objective of the study remains the comparative evaluation of the expressed satisfaction as well as task performance of the selected groups of modeling users. Some recommendations for improving the general usability of the MP software may emerge from the users' feedback, but the intent remains to test the hypothesis that more experienced modelers report having a better MP user experience than those without prior modeling background.

E. THESIS ORGANIZATION

This thesis comprises five chapters including this introduction. Chapter II provides background on the meaning and methods of software usability testing and reviews a selection of the literature available on those methods. The review offers context for the study topic of this thesis and helps to contrast the objectives of this study from what may be formally understood as software usability testing. Chapter III describes the methodology selected for this study and how the hypothesis was tested. Chapter IV presents the results of the study and discusses the analysis conducted. Finally, Chapter V offers conclusions and recommendations for future study.

F. RESEARCH METHODOLOGY

To test the hypothesis regarding the expressed usability of MP, this thesis first explores some of the available literature on software usability testing concepts and metrics to establish a basis for the comparison among subject groups. Then, the author leveraged a software usability study sponsored by the Office of Naval Research (ONR) following a Human Subjects Research (HSR) protocol to gather experimental data on users' satisfaction with the MP product. Analysis of the results of this study led to the conclusions and recommendations reached within this paper.

THIS PAGE INTENTIONALLY LEFT BLANK

II. BACKGROUND

A. MONTEREY PHOENIX

Monterey Phoenix is "a Navy-developed formal language and approach for modeling behaviors for systems, software, hardware, people, and organizations, and their dependencies on one another and the environment" (Giammarco 2018). System architects and developers employ MP to model business or operational processes as well as system and system-of-system architectures. However, the MP approach to modeling applies lightweight formal methods to add mathematical rigor to the definition of a process or system model. According to the NASA website, the phrase "mathematically rigorous" when referring to formal methods "means that the specifications used in formal methods are well-formed statements in a mathematical logic and that the formal verifications are rigorous deductions in that logic" (NASA 2016). Put more simply, using formal methods to define a system architecture allows a verifiable precision in each behavior or interaction that can be subsequently checked by a process that always results in the same outcome.

Defining an architecture in MP outputs a set of scope-complete event traces that represent every possible outcome or scenario variant for the system. Scope-complete in this context indicates that the possible outcomes generated from MP are limited to a user-specified number of iterations through the model. Each event trace presents a graphical representation of the actors and interactions between them in that scenario.

The sample MP model defined in Figure 1 depicts a simple architecture representing a communication scenario. The ROOT keyword defines a base actor or object, and the syntax that follows defines the activities performed by each actor. In this case the Sender can perform the send action zero to many times, and the Receiver can perform the receive action zero to many times. Adding the COORDINATE keyword constrains the behaviors of the Sender and Receiver to ensure that the receive action only occurs after the send action. Without this constraint applied, the model allows the Sender to send and the Receiver to receive without any relationship between the two actions.

```
17 SCHEMA simple_message_flow
18
19 ROOT Sender:   (* send *);
20 ROOT Receiver: (* receive *);
21
22 COORDINATE  $x: send    FROM Sender,
23             $y: receive FROM Receiver
24    DO ADD $x PRECEDES $y; OD;
```

Figure 1. Sample MP Model Grammar

When the MP model is run as defined, two possible event traces result. Figure 2 shows both of these generated traces. In the first trace, the Sender does not perform the send action which, due to the constraint, also means that the Receiver does not perform the receive action. The other possible outcome in this scenario results when the Sender performs the send action one time. Now, the Receiver also performs the receive action one time while the arrow indicates that the send action must precede the receive. These event traces result when the MP model is run for scope 1. Increasing the scope to 2 creates a third event trace where Sender performs a second send action which also allows the Receiver to perform a second receive action.

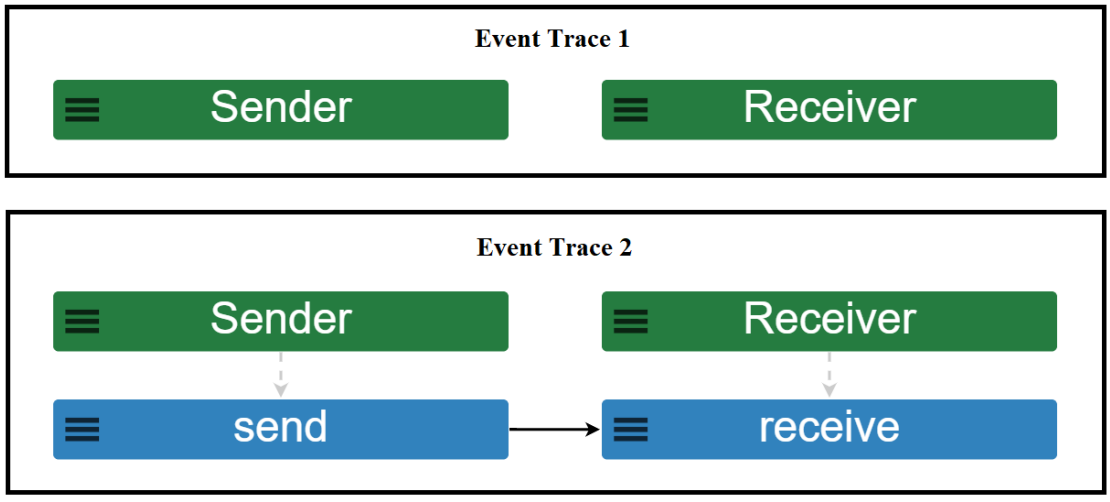

Figure 2. MP Event Traces

Mikhail Auguston (1991) introduced the concept of using behavior models employing event grammars and traces and developed the MP grammar in partnership with the Naval Postgraduate School. Prototypes for the current MP software created by Dr. Auguston were eventually made accessible to the public through the current MP-Firebird web application with graphical user interface (GUI) developed by Philip McCullick and Michael Nigh at NPS CED3 on a CRUSER-sponsored project managed by Kristin Giammarco in the NPS Systems Engineering Department. This GUI provided a friendly interface to compliment the compiler and trace generator created by Dr. Auguston. NPS staff including Cliff Whitcomb and Kristin Giammarco further aided the maturation of the MP software eventually creating the visual layout for event traces shown in Figure 2.

Monterey Phoenix technology has won two best paper awards and has been the subject of many invitational presentations including the System of Systems Engineering Collaborators Information Exchange (SoSECIE), the NAVSEA Modeling & Simulation (M&S) Forum, the INCOSE SoS Working Group, and NASA's Independent Verification & Validation Facility. Monterey Phoenix is recognized for its provision of a new approach to the detection, prediction, classification and control of emergent system of system (SoS)-level behaviors (Giammarco and Auguston 2018). Many proponents of MP throughout the DOD and its partners have expressed a desire for an interface that makes the tool easier for them to use, especially leveraging their familiarity and training investment in other graphical and popular system architecting tools being adopted within DOD acquisition as part of model-based systems engineering (MBSE) or digital engineering efforts (Kristin Giammarco, personal communication, July 20, 2019).

B. WHAT IS USABILITY

According to the International Organization for Standardization (ISO) "[Usability refers to] the extent to which a product can be used by specified users to achieve specified goals with effectiveness, efficiency and satisfaction in a specified context of use" (ISO 9241–11, 2018). This study examines both the effectiveness of the MP software as well as the satisfaction expressed by MP users but not for the express purpose of improving those qualities or characteristics. Rather, the study compares the usability reported by users with

different levels of domain experience where the domain comprises architecture and behavior modeling.

C. HOW TO EVALUATE USABILITY

1. Formative vs. Summative Evaluation

According to the Usability Body of Knowledge website, usability testing generally falls within two broad categories: formative and summative evaluation (Scriven 1967).

Formative evaluation, as the name suggests, occurs during the design and development of a system and seeks to form that design through early user involvement. This type of evaluation provides direct feedback to the development team through iterative cycles of testing. Walter Maner lists the potential benefits of formative evaluation as the following (1997):

- may be done very early in the design process, when about 10% of the project resources have been expended
- may give the first solid measurements of task performance
- may help designers gain empathy for persons trying to use the software in real situations
- may help developers decide when the project can move on to the next stage
- may increase user interest and eventual acceptance of the final product
- may uncover problems that were not noticed during iterative prototyping

Summative evaluation on the other hand looks at systems which have reached a level of maturity to allow their use by the intended audience under realistic conditions. Summative testing tends to be more formal in nature gathering quantifiable metrics and producing detailed reports. Summative usability testing may establish a usability benchmark or allow comparison with usability requirements (Usability Body of Knowledge n.d.).

2. Without Users: Inspection

Jakob Nielsen (1995) describes four basic ways of evaluating user interfaces:

- automatically (usability measures computed by running a user interface specification through some program)

- empirically (usability assessed by testing the interface with real users)
- formally (using exact models and formulas to calculate usability measures)
- informally (based on rules of thumb and the general skill and experience of the evaluators) (377-378)

According to Nielsen, automatic and formal evaluations are either not currently feasible or scalable to large systems (1995). Due to difficulties or expense that may be associated with recruiting actual users during the design of a system, usability inspection techniques provide an alternative means of evaluation. These techniques often require trained usability professionals and generally fall into the formative category of evaluation as they can occur before the system design has been completed. Some examples of usability inspection techniques include heuristic evaluation (Nielsen 1994), cognitive walkthroughs (Lewis et al. 1990), formal usability inspections (Kahn and Prail 1994), pluralistic walkthroughs (Bias 1991), feature inspection (Bell 1992), consistency inspection, and standards inspection (Wixon et al. 1994). Several of these inspection methods utilize heuristics or usability design principles for comparison with the proposed design while others walk through the user's function or task process and attempt to identify potential usability challenges within that process.

3. With Users: Testing

a. In-Person Usability Testing

When users are available in the proximity of a testing facility or usability lab, system owners may opt to conduct moderated usability testing in person. While discussing the SilverPlatter company experience with usability testing, Elizabeth Morley stated that "the key to successful testing is to observe" (1995). Observation may include direct interaction with users in the room or through the use of a two-way mirror that separates users from observers which may include developers. Recording sessions with users allows analysis and findings not detected during the active user interaction.

The moderator of a usability test may choose from common techniques described in an article by Jen Romano Bergstrom for Usability.gov (2013).

> Concurrent Think Aloud (CTA) is used to understand participants' thoughts as they interact with a product by having them think aloud while they work. The goal is to encourage participants to keep a running stream of consciousness as they work.
>
> In Retrospective Think Aloud (RTA), the moderator asks participants to retrace their steps when the session is complete. Often participants watch a video replay of their actions, which may or may not contain eye-gaze patterns.
>
> Concurrent Probing (CP) requires that as participants work on tasks—when they say something interesting or do something unique, the researcher asks follow-up questions.
>
> Retrospective Probing (RP) requires waiting until the session is complete and then asking questions about the participant's thoughts and actions. Researchers often use RP in conjunction with other methods—as the participant makes comments or actions, the researcher takes notes and follows up with additional questions at the end of the session.

When choosing among these techniques, the moderator must consider the objectives of the study and understand that choosing to interrupt the user with either of the concurrent approaches may negatively affect performance metrics or the natural thought and work process of the user. Alternatively, waiting until the session is complete as with the retrospective approaches relies on the memory of the user which may lead to poor data collected.

While recruiting users for in-person testing, hiring professional, experienced moderators, and procuring the facility and recording equipment required for effective usability test execution may seem like a very expensive approach, research shows that "most usability problems can be detected with only three to five subjects" and that "the most severe usability problems are detected by the first few subjects" (Virzi 1992). Further, Virzi finds that repeating the same test with additional users is unlikely to reveal new information. Of note, this research provides evidence for the "small scope hypothesis" which states that most flaws or "bugs can be found by testing a program for all test inputs within some small scope" (Jackson 2006). Monterey Phoenix leverages this same principle to support the assertion that most design flaws or missing and unexpected behaviors can be discovered by evaluating the model at a relatively small scope.

b. Remote Evaluation

When in-person usability testing is impossible or impractical due to distribution of the user base or the costs associated with a traditional usability lab, modern software and network capabilities allow remote evaluation as an alternative. In their book on the subject of remote research, authors Bolt and Tulathumutte (2010) describe both qualitative and quantitative methods or remote testing that are either moderated or unmoderated respectively. The moderated, or qualitative, approaches resemble the in-person techniques previously described but utilize remote screen and audio sharing software to allow interactions with the subjects. Popular applications supporting this protocol include Adobe Connect, GoToMeeting, and WebEx.

Bolt and Tulathumutte describe unmoderated, or quantitative, methods that allow researchers to gather usability data from users without needing to interact directly. These methods are distributed across a spectrum from concrete to conceptual methods as shown in Figure 3.

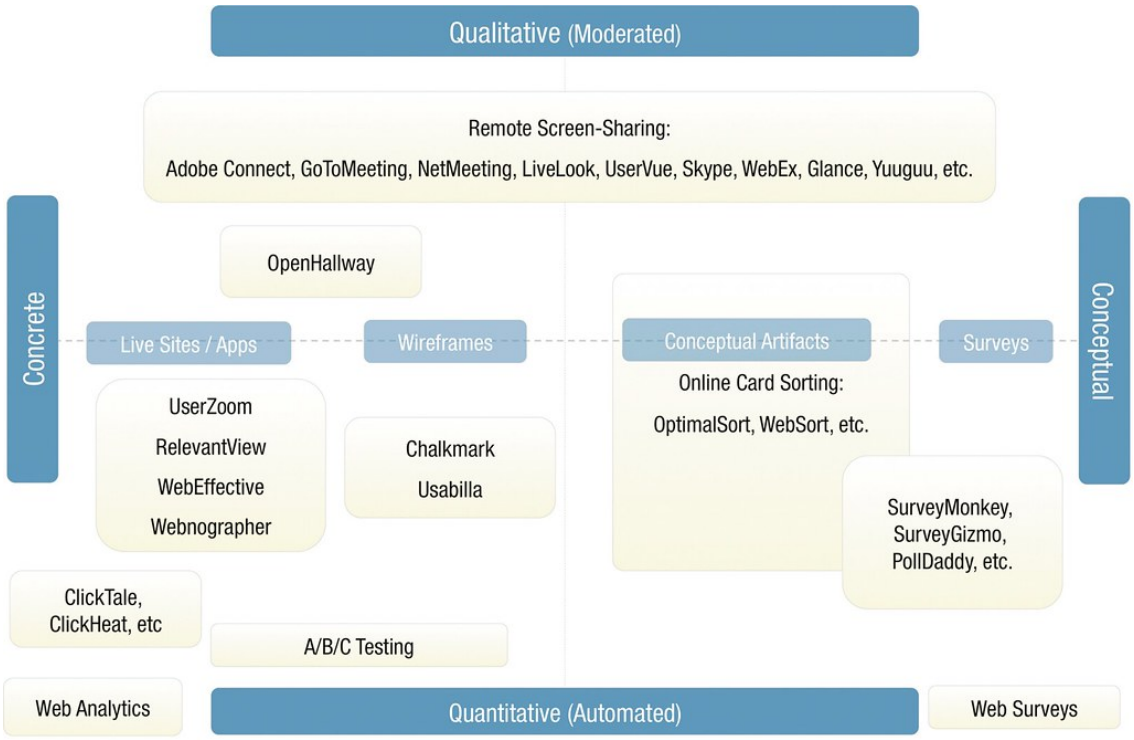

Figure 3. Remote Evaluation Methods. Source: Bolt and Tulathumutte (2010).

Some of these quantitative methods rely on analysis of web analytics to see which web pages were visited or on click-maps to reveal where on the page users are most frequently clicking. When the actual site or application is not yet available, online surveys or conceptual artifacts and wireframes provide alternate analytical opportunities.

c. Wizard of Oz

An interesting method for assessing the usability of computer systems that have not yet completed development originated from the work of J.F. Kelley in 1984. Termed the Wizard of Oz (the "Oz Paradigm" in his paper), this methodology places the usability tester or experimenter outside the view of the user like the Wizard behind the curtain in the popular movie. The tester then plays the role of the to-be intelligent system, simulating its function by intercepting the interactions of the user and responding to assess how the user behaves. This method allows the observation of actual user interactions even though the

system software has not yet been written and can effectively shape the design of that software (Kelley 1984).

d. Benchmarking

While several of the usability testing methods presented here aim to shape the design and development of an emerging system, often investigators desire to establish a baseline of usability for existing systems. The benchmark study offers a raw measure of user performance and may serve as the basis for subsequent investment decisions for changes to the system and comparing future usability improvement as a result of those changes (Berkun 2003). In his essay on the art of usability benchmarking, author Scott Berkun provides several caveats when designing a benchmarking study.

1. "Never ask participants in the study to use verbal protocol during the study; verbal protocol is a way to try and understand what users are thinking while they are performing tasks. Verbal protocol has negative impacts on many measurable user performance metrics, such as time to complete tasks."

2. Wait for a stable software build.

3. Identify core tasks (frequently performed or for other reasons like something every user has to do at least once).

4. Establish core metrics. The most common examples according to Berkun include the following:

    - Success/Failure within a time threshold
    - Time on task
    - # of errors before completion

5. Set performance goals for tasks. Goals may be either arbitrary initially or may adopt the performance of a competing product to serve as a baseline for the system under test.

D. USABILITY METRICS

1. System Usability Scale

Of the three dimensions of usability described within the ISO standard (ISO 9241–11, 2018) user satisfaction proves to be difficult to assess but also lends itself to generalization which can be compared across systems (Brooke 1996). Unlike measures of effectiveness which can be unique to the purpose of each system and efficiency which may also be very context and system specific, satisfaction assessments depend more on the response of the user rather than the performance of the system directly. In 1996, John Brooke developed a method for assessing user satisfaction with a system through a simple ten question survey. Brooke determined that lengthy questionnaires would likely not be completed by the user if issued after a potentially frustrating system experience. As a result, Brooke decided to keep the questionnaire very short. Starting with a pool of 50 candidate questions, Brooke offered the survey to users of two very different systems: one that was generally considered "easy to use" and one that was considered challenging even for technically proficient users. Responses to the survey questions came in the form of a Likert scale in the hopes of eliciting extreme opinions from users that would show general agreement when compared across those users. The Likert scale offered five options ranging from "strongly disagree" to "strongly agree" as shown in Figure 4.

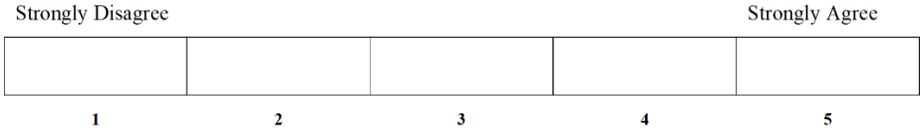

Figure 4. Example Likert Scale

Brooke used the responses from these initial surveys to select the ten questions that generated the most extreme results. Also, half of the questions selected resulted in strong agreement while the other half resulted in strong disagreement. By alternating the sequence of questions to present one expected to generate agreement with one expected to generate disagreement, the survey forced respondents to think carefully about each question. The

original 10 questions selected appear in Table 1 and have remained constant since Brooke created the scale.

Table 1. System Usability Scale (SUS). Source: Brooke (1996)

	Strongly Disagree				Strongly Agree
1. I think that I would like to use this system frequently.	1	2	3	4	5
2. I found the system unnecessarily complex.	1	2	3	4	5
3. I thought the system was easy to use.	1	2	3	4	5
4. I think that I would need the support of a technical person to be able to use this system.	1	2	3	4	5
5. I found the various functions in this system were well integrated.	1	2	3	4	5
6. I thought there was too much inconsistency in this system.	1	2	3	4	5
7. I would imagine that most people would learn to use this system very quickly.	1	2	3	4	5
8. I found the system very cumbersome to use.	1	2	3	4	5
9. I felt very confident using the system.	1	2	3	4	5
10. I needed to learn a lot of things before I could get going with this system.	1	2	3	4	5

Nathan Thomas describes the method for calculating the usability score on the UsabilityGeek website. (2015)

1. "Users will have ranked each of the 10 templates questions shown in Table 1 from 1 to 5, based on their level of agreement"
2. "For each of the odd numbered questions, subtract 1 from the score"
3. "For each of the even numbered questions, subtract their value from 5"
4. Sum these new scores to calculate a total modified score
5. Multiply the new total score by 2.5
6. The resulting score will be normalized on a scale of 100 but does not represent a percentage

Research on SUS and analyzed data from over 5000 users across 500 different evaluations shows that the average usability score with this method is a 68 (Sauro 2011). Jeff Sauro found that by grouping usability scores according to percentile, a corresponding letter grade (A,B,C,D,E,F) could be assigned with that average score of 68 representing a C. Figure 5 shows the graph Sauro created to represent this scoring approach.

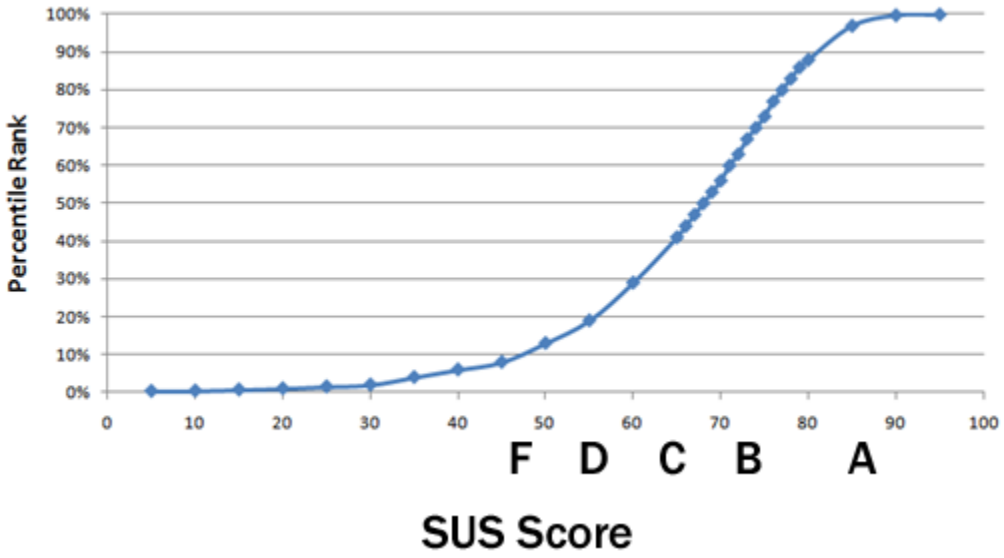

Figure 5. How Percentile Ranks Associate with SUS Scores and Letter Grades. Source: Sauro (2011).

2. NASA Task Load Index

Measuring the efficiency dimension of system usability presents challenges. Primarily, efficiency can be unique to the context and purpose of a system (Brooke 1996). In 1988, Sandra Hart and Lowell Staveland developed an index for assessing the workload created by use of a system (1988). The NASA Task Load Index (NASA-TLX) is "a multi-dimensional scale designed to obtain workload estimates from one or more operators while they are performing a task or immediately afterwards" (Hart 2006). Hart defines workload as the human cost (e.g., fatigue, stress, illness, and accidents) of accomplishing mission requirements. While some aspects of workload could be measured empirically such as the number of accidents resulting from a task, Hart and Staveland proposed that workload metrics would need to be generalized according to the perceptions of each human user to allow for comparison across different systems. As a result, the NASA-TLX comprises "six subscales that represent somewhat independent clusters of variables: Mental, Physical, and Temporal Demands, Frustration, Effort, and Performance."

According to the NASA Task Load Index Pencil and Paper Package, interested system owners can assess the NASA-TLX overall workload score based on a weighted average of ratings on six subscales and through the use of a simple questionnaire (Human Performance Research Group, NASA Ames Research Center 2016a). This questionnaire with the six subscales is shown in Figure 6.

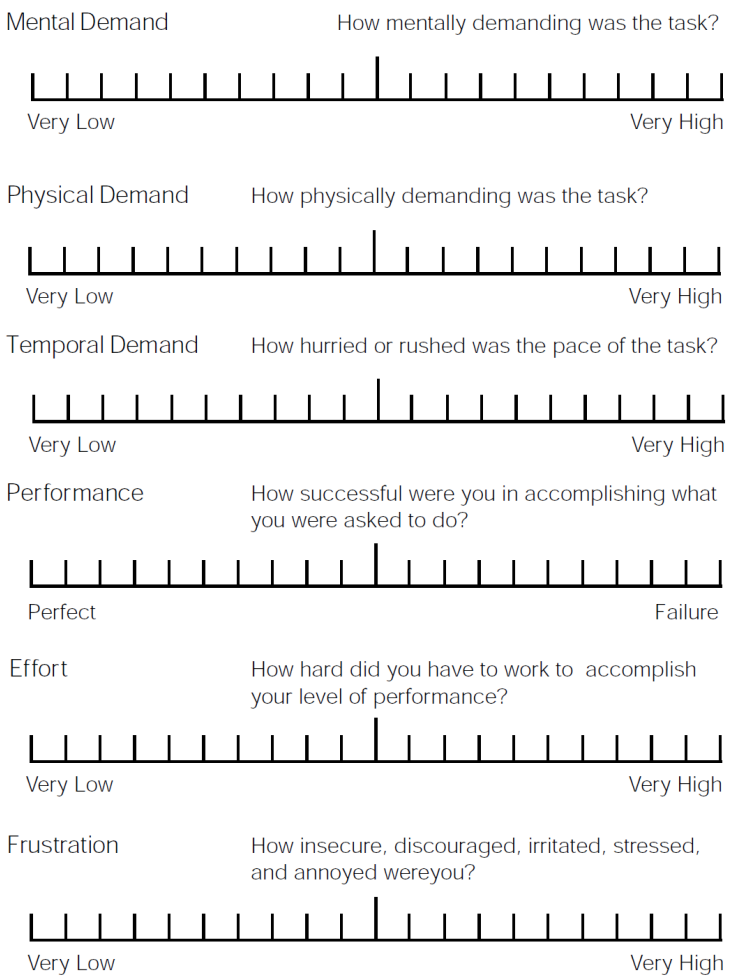

Figure 6. NASA-TLX Example Collection Instrument. Source: Human Performance Research Group, NASA Ames Research Center (2016b).

The original survey scale allowed 21 selections in response to each question on a spectrum ranging from very low to very high for five of the six questions and good to poor for the performance question. More recent surveys, as shown in Figure 5, replace good with perfect and poor with failure. Ms. Hart notes in her pencil and paper instructions that this particular dimension has caused some confusion as it operates in the reverse of the other rating scales used.

The original pencil and paper instructions describe the administration of the survey to assess the NASA-TLX (Human Performance Research Group, NASA Ames Research

Center 2016a). Immediately following the performance of the task to assess, researchers issue the rating scale questionnaire to the user. On completing the six-question scale, the user must complete a pairwise comparison of the sources of workload to support the development of a weighting system for calculation of the actual index value. The pairwise comparison asks the user to select between each pairing of the workload factors to assess which factor contributed most significantly to the workload of the task.

After collecting the survey from each respondent, the researcher calculates the index by first developing the weighting for each of the six factors unique to each respondent. The weight for each factor is simply the count of the number of times the user selected that factor in the 15 pairwise comparison questions. The analyst then multiplies the actual rating by the weighting previously calculated. The sum of these adjusted ratings must then be divided by 15 to yield the final weighted rating for each user. This weighted rating provides a score that can then be further compared and analyzed among different users having performed the same or very similar task.

3. Net Promoter Score

Net Promoter Score**Error! Bookmark not defined.**, or NPS**Error! Bookmark not defined.**,[1] measures customer experience and predicts business growth (NICE Satmetrix 2017). The Net Promoter Score provides a very simple way to request and assess the overall satisfaction with or loyalty to a company's brand or a product like MP. Through a single question, the surveyor assesses the likelihood that a consumer enjoyed their overall experience and will refer others to try the company, brand, or product.

> "How likely is it that you would recommend [brand / product] to a friend or colleague?"

When employed as part of a software usability study, NPS reveals overall perception or satisfaction with the user experience of the application. According to NICE

[1] Net Promoter, NPS, and the NPS-related emoticons are registered trademarks, and Net Promoter Score and Net Promoter System are service marks, of Bain & Company, Inc., Satmetrix Systems, Inc. and Fred Reichheld.

Satmetrix, analysts calculate NPS from the aggregate responses to the survey question by first grouping the respondents as according to the scale shown in Figure 3.

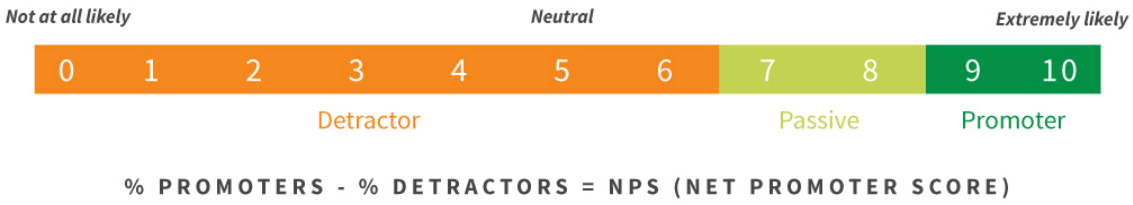

Figure 7. Net Promoter Score. Source: (NICE Satmetrix 2017).

NICE Satmetrix defines the three groups as follows:

1. **Promoters** (score 9–10) are loyal enthusiasts who will keep buying and refer others, fueling growth.
2. **Passives** (score 7–8) are satisfied but unenthusiastic customers who are vulnerable to competitive offerings.
3. **Detractors** (score 0–6) are unhappy customers who can damage your brand and impede growth through negative word-of-mouth. (NICE Satmetrix 2017)

The analyst calculates the percentage of Detractors and subtracts that value from the percentage of Promoters which results in the NPS for the system or application in the case of a usability study.

4. Free-Text Responses and Observations

In addition to the rating scales and systems described here, usability studies often ask open-ended questions or simply observe the user during performance of a designated task or function within the system. Although the utility of these comments to the calculation of specific metrics may be limited, they can be very helpful in identifying ways to improve the product (Albert and Tullis 2013). In some cases, free-text responses can identify common responses which may be adapted for use as metrics on overall usability.

E. USER-CENTERED DESIGN

Usability testing is only one part of a larger engineering process. Often referred to as usability engineering (Paul et al. 2014) or user centered design (U.S. Department of Health & Human Services n.d.) or human-centerd design as defined in the international standard ISO 9241–210:2019, this collection of engineering processes may vary widely in implementation practices (Bornoe and Stage 2014). Though these specific implementations may vary, user centered design (UCD) generally differs from system engineering in its focus on the user rather than on the system. In fact, user centered design has been described as "the opposite of the system-driven philosophy generally used in engineering" (Seffah and Metzker 2004).

However, when looking more broadly at the technical and management processes employed within system engineering and shown in Figure 8, the alignment of SE and UCD becomes more apparent.

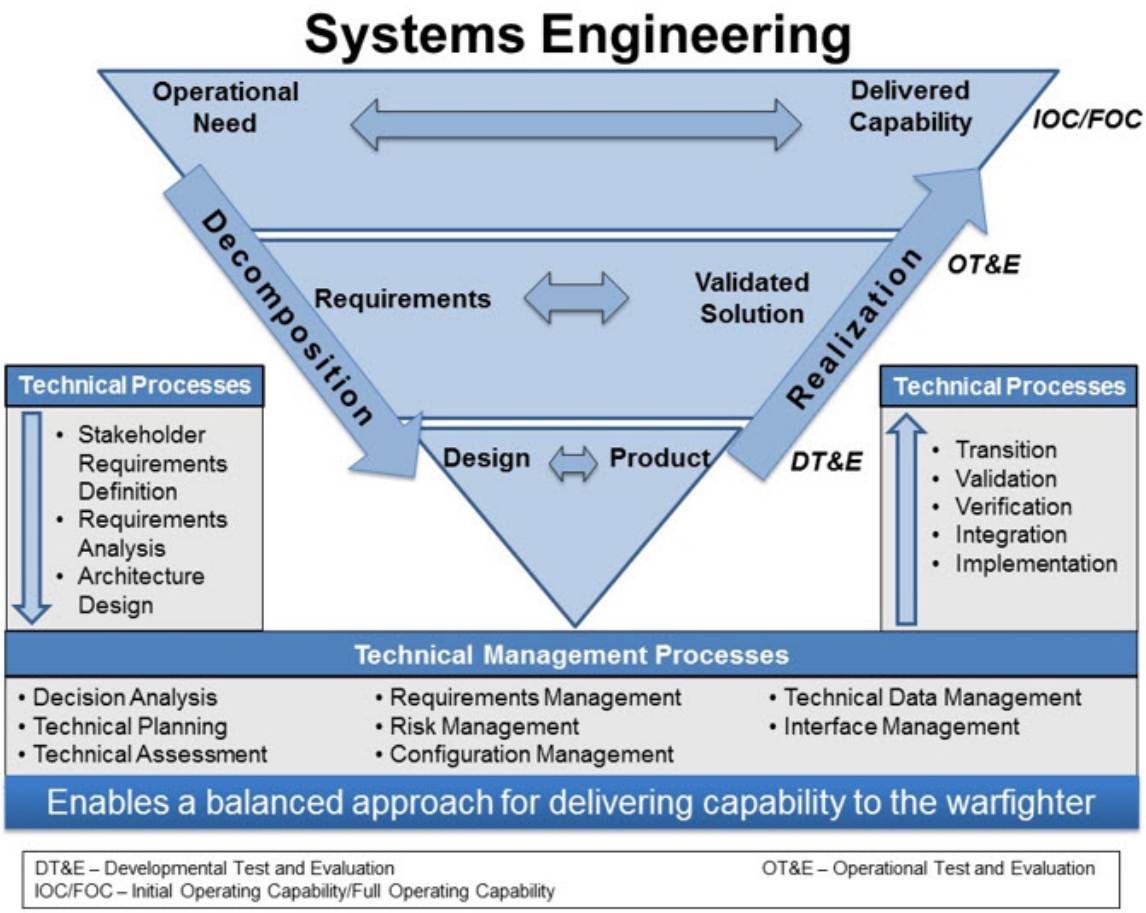

Figure 8. System Engineering Processes. Source: Defense Acquisition University(2017).

Both SE and UCD follow the pattern of: identifying requirements, designing a solution, developing and integrating capability, verifying and validating the solution, and transitioning the system to operations. Figure 9 illustrates a step-by-step guide for performing UCD and shows where usability testing generally falls within the system development process.

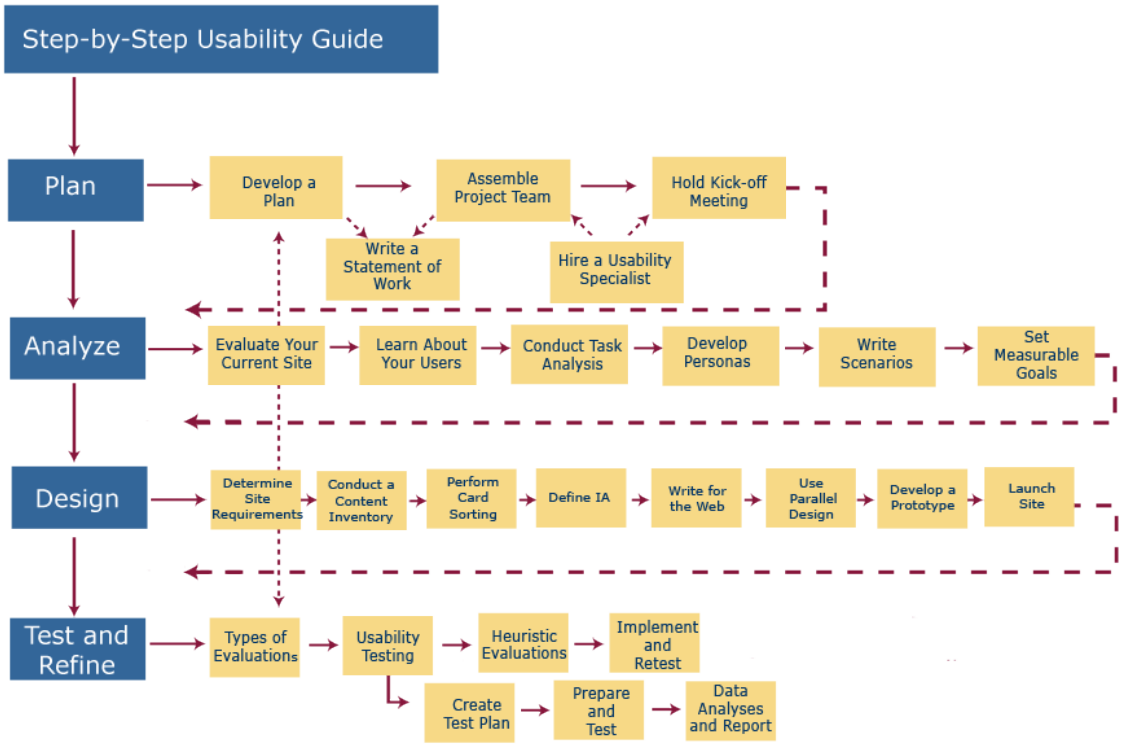

Figure 9. User-Centered Design Process Map. Source: U.S. Department of Health & Human Services (n.d.).

Usability engineering and UCD are also closely related to human factors engineering which "refers to the design of machines, machine systems, work methods, and environments to take into account the safety, comfort, and productiveness of human users and operators" (Holstein and Chapanis 1999). In some ways, usability can be thought of as a measure of the successful outcome of human factors engineering.

As previously indicated, usability engineering, like general system engineering practices, both leads to and supports the development of system requirements. In the case of usability requirements for an interface design, these may take the form of goals supporting the following as viewed from the lens of the primary system user:

> "Efficiency of use: goals are easy to accomplish quickly and with few or no user errors"

> "Intuitiveness: the interface is easy to learn and navigate; buttons, headings, and help/error messages are simple to understand"

> "Low perceived workload: the interface appears easy to use, rather than intimidating, demanding and frustrating" (Foraker Labs n.d.)

Given that usability requirements such as the above may be difficult to quantify or measure, the techniques of UCD and usability testing frequently support an iterative approach to delivering a software solution that successfully achieves the usability goals.

III. METHODOLOGY

Prior to initiating the usability study for this thesis, the NPS Institutional Review Board reviewed and approved the approach and study methods proposed here.

A. STUDY SUBJECTS

The subject population solicited for this study included NPS students and NPS civilian employees. The study team generated an email invitation, which was distributed to all NPS students, staff and faculty to ensure the broadest possible participation. The email described the study purpose and requirements including time required and completion timeline. To avoid bias that may accompany users with prior MP software experience, the invitation stated that eligibility to participate in the study was contingent on having no prior use of MP.

The study coordinator contacted eligible respondents and provided them with a consent form which they needed to sign if they agreed to participate in the study. Upon receiving a signed consent form from each willing respondent, the coordinator distributed an entry questionnaire to each subject. This questionnaire collected information about the general background of the participant as well as their level of modeling experience measured in years. Responses to these questions enabled the study team to assign each subject a unique study identifier and place them in one of two groups: no prior modeling experience (0 years) or at least some prior modeling experience (greater than 0 years). In addition, the questionnaire presented a series of questions designed to assess the comfort of the subject with software programming including the creation of formulas in Microsoft Excel. These questions presented a Likert scale and provided the research team additional data for use in analyzing usability perceptions at the completion of the study.

Each subject group received the same instructions, information, and MP software. Instructions permitted subjects to ask general questions about MP modeling and indicated subjects would receive responses via email.

B. MONTEREY PHOENIX FAMILIARIZATION

As the study excluded users with prior MP modeling experience, an expert MP user prepared a short tutorial designed to introduce new users to both the MP grammar as well as the user interface itself. The tutorial consisted of four parts, which were narrated to allow a controlled delivery of the tutorial to all subjects. Playback of all parts of the video tutorial as well as tool interactions suggested by the tutorial requires approximately 90 minutes.

1. Part 1 – Behavior Modeling Concepts

The first part of the tutorial provides an overview of the MP tool and seeks to answer general questions about why MP was built and how behavior models created in MP can be used. The tutorial defines key concepts such as root, composite, and atomic events from which all modeled system behaviors are built. Illustrations presented the basic relations between events as either inclusion or precedence. Figure 2 displays both relations as the Sender root event includes the send event and the send event precedes the receive event. Figure 2 also renders an example of the event trace which represents a single instance of behavior within MP. Lastly, this portion of the tutorial explains that an event grammar such as MP composes and constrains the structure of possible event traces.

2. Part 2 – Tool Navigation

With the basic concepts of behavior modeling explained, part 2 of the tutorial provides the high-level features of the MP user interface which is web browser-based and available at https://firebird.nps.edu. Developers created MP as a single page application (SPA) with minimal navigation required to access the different features of the tool. The import and export menu options allow the user to open and save MP models. The user enters and modifies MP model definition code in a panel to the left of the screen. This panel offers color coding of key words to assist the modeler during development. The tutorial explains the use of scope to control the number of iterations for events defined to occur more than once. Leveraging the Small Scope Hypothesis from Daniel Jackson's work in lightweight formal methods (2006) which states that most flaws can be exposed on small counterexamples, MP generally reveals most system behavior issues when run at only

scope 1, 2 or 3. Finding these flaws in logic early in the design of a system greatly reduces the cost of fixing them after implementation.

3. Part 3 – Modeling Systems and Interactions

Now that the subject understands the MP basic language and interface, this part of the tutorial expands on the ways in which MP modeling represents system behaviors both within the system of interest and with external systems as well. To model behaviors involving the exchange of energy, matter, material wealth, or information, MP employs verb-oriented atomic or composite events. The MP modeler implements COORDINATE and SHARE ALL code statements to create the relationships and component exchanges within the behavior model. Through the addition and removal of these COORDINATE and SHARE ALL statements, the user constrains the model and generates event traces that represent all possible system outcomes for the scope selected.

4. Part 4 – Behavior Compositions and Good Modeling Practices

According to the tutorial in part four, events can be composed into a model of behavior using sequential, concurrent, alternate, optional, and iterating operations. The tutorial then continues and describes best practices for creating MP system behavior models which include the following:

- Build models incrementally, running the model after every change to ensure the model still runs as expected.
- Use comments at the top and throughout to explain the model or different parts of the model.
- Use indentation to align the code for readability and quick comprehension.

C. USER TASK DESCRIPTION

Vetted and consenting subjects receive the self-guided MP software tutorial videos to equip them with a fundamental orientation to the tool they will be using in the study. As described previously, watching the four parts of the tutorial requires approximately 60 minutes for a single viewing with an additional 30 minutes allotted for practice activities

within the MP tool as part of the guidance. Instructions then directed subjects to read a mission narrative that describes representative behaviors for Unmanned Aerial Vehicle (UAV) launch preparation procedures (see Appendix C). The provided instructions then ask study participants to model this mission narrative individually using the MP tool and while recording the modeling session via the NPS Blackboard Collaborate software. Collaborate recordings associate to the de-identified subject identifier and only capture the subject's computer screen along with any voice comments added by the user. Detailed Collaborate setup instructions accompany the study package received by all subjects. To facilitate the gathering of qualitative observations, subjects received guidance to voice-narrate their steps, reasoning, questions, and solutions as they are using the MP software. The study allots 30 minutes to accomplish this recorded MP modeling activity.

The final task presented to study subjects on completion of the modeling task requires them to complete an exit questionnaire (see Appendix B) which should take approximately 20 minutes to populate.

Participants may complete all the study tasks presented here from any internet-connected computer and should require approximately 2.5 hours of devoted time.

D. OBSERVATION AND DATA COLLECTION

Collected data includes the exit questionnaire responses, questions generated by the subjects, audio narrative, chat, and Collaborate application screen share video of the MP software in use by each subject. The de-identified data collected provides analysis of the primary aspects of usability of interest to this study: user satisfaction, efficiency, and capability effectiveness. The research investigators performed both qualitative and quantitative analysis of this data and based conclusions about MP software usability on the quantitative and qualitative results.

Subject-authored MP models allowed additional qualitative and quantitative analysis by the researchers as did observation of the process used to construct the model and types of problems encountered during MP modeling.

The quantitative approach employed hypothesis testing to determine whether there is a statistically significant difference between subject groups' reported satisfaction, efficiency and effectiveness.

While not comprehensively reported within this document, the study team will deliver additional participant feedback provided on the exit questionnaire or during the recorded modeling session to the MP software developers to plan follow on improvements for future software releases.

1. User Satisfaction

As user satisfaction depends on the perceptions of each user, the study employed an exit questionnaire to gather data for calculating two well-established measures based on ratings scales. The questionnaire presented respondents with the System Usability Scale and the Net Promoter Score set of questions. The study analysis team calculated the SUS score from each respondent and then established the average and standard deviation within each study group. Next, the team calculated a Net Promotor Score as reported within each study group. After calculating the average SUS score and the NPS for each group, the team compared the two scores among the groups.

2. Efficiency

To measure the efficiency reported by the study participants, the exit questionnaire offered an open-ended question seeking the portion of time spent referring back to the MP tutorial during the modeling task. The study team established a heuristic based on experience and intuition that users who spend more time consulting documentation than using the software indicate poor usability of the system or confusion with the capabilities offered. Therefore, the study team calculated the average of the portion of time reported by each user within the subject groups resulting in a referral score for each group with a lower percentage representing greater usability of the product. The possibility exists that this measure reflects the quality or shortcomings of the tutorial itself, but the researchers attempted to control for this by providing each user the same recorded tutorial and ensuring that all users had the same amount of prior experience with MP.

The study employed a second and primary measure of efficiency, the NASA Task Load Index. Following the original NASA TLX pencil and paper instructions, the research team prepared the workload rating scale and sources of workload comparison tool as part of the exit questionnaire. After some discussion concerning the overall length of the exit questionnaire, the principal investigator suggested the removal of the weighting procedure as well as the question regarding physical workload given that this dimension is not applicable in a computer-based modeling task. The original developer of the NASA TLX, Sandra Hart, examined these kinds of modifications to the protocol when she revisited the tool 20 years after its creation (2006). Ms. Hart provided the following assessment:

> In other articles, the authors propose and apply a modified version of the original scale. Some add subscales (6 articles), while others delete them (12 articles) or redefine the existing subscales to improve the relevance to the target task or experimental questions. While increasing the fit between the generic NASA-TLX labels and definitions to a situation can be an excellent strategy, it does require establishing the validity, sensitivity, and reliability of the new instrument before using it. A good example of such an effort may be found in Park & Cha (1998) where several variants of NASA-TLX were evaluated for use by Korean drivers. A practical problem with adding, deleting, and re-defining subscales and continuing to refer to the result as "NASATLX" even though the new scale shares only a passing similarity with the original is that it makes it difficult to summarize the circumstances under which the original scale is and is not useful.
>
> The most common modification made to NASA-TLX has been to eliminate the weighting process all together or weighting the subscales and then analyzing them individually. The former has been referred to as Raw TLX (RTLX) and has gained some popularity because it is simpler to apply; the ratings are simply averaged or added to create an estimate of overall workload. In the 29 studies in which RTLX was compared to the original version, it was found to be either more sensitive (Hendy, Hamilton, & Landry, 1993), less sensitive (Liu & Wickens, 1994), or equally sensitive (Byers, Bittner, Hill, 1989), so it seems you can take your pick.

Given this assessment, the team agreed to leverage the Raw TLX approach and eliminate the weighting process. The team calculated the simple average workload score for each group based on the individual user responses and then compared the average scores to identify any statistically significant differences between what was reported.

3. Task Effectiveness

Establishing a measure of the effectiveness for the study required the creation of an evaluation tool unique to the MP product. The research team determined that creating an accurate and complete model represented a desired end state for application use as it is a prerequisite for the creation and analysis of the resultant event traces. Dr. Kristin Giammarco, an MP subject matter expert, prepared a rubric for objectively grading the outputs of the study participants' modeling efforts. Table 2 summarizes the rubric created by Dr. Giammarco.

Table 2. Grading Rubric for MP Models. Source: Dr. Kristin Giammarco, email to author, July 19, 2019.

	Max Point Value
	100
Point Adjustment -->	
Documentation	**10**
MP model leads with a comments section containing introductory information (e.g., purpose, sources, author, date)	6
Model contains comments throughout if needed to explain intention, assumption, or rationale.	4
Form	**35**
MP model contains a SCHEMA name	2
Model contains correct number of root events	2
Root events present are not missing any events described in the narrative	8
Event inclusion is employed where appropriate	5
The model generates the expected number of scenarios upon running	5
The model generates the expected scenarios	5
Events in different roots with precedence relations are coordinated correctly	8
Readability	**20**
Spacing is present between statements	3
Spaces are present on either side of events encapsulated in parentheses or brackets	2
Opening and closing parentheses are aligned	3
Sequential events are vertically stacked and left-aligned	3

	Max Point Value
Closing parentheses following alternative events are right-aligned to the pertinent OR operator	2
The concluding semicolon for root event grammar rules are lined up beneath its corresponding ROOT keyword	1
The left hand part of grammar rule is on its own line (applies to both roots and composites)	1
COORDINATE statements are broken up across multiple lines and indented	2
Order of roots results in well-organized event traces that are easy to follow	3
Diversity	**20**
Model contains alternate events	10
Model contains one or more optional events	5
Model contains iterating events	2
Model contains concurrent events	1
Model contains shared events	2
Efficiency	**15**
No redundant statements or expressions are unnecessarily present	5
Constraints are applied as soon as all pertinent events have been declared	5
Elapsed time for generating scenarios is within the expected range for the model size	5
FREE TEXT COMMENT	
Score	**100**

The criteria used in the grading rubric are not intended to measure the usefulness of the MP model towards answering an analysis question; rather they examine structural elements of the model common to any analysis. Again, the purpose of this evaluation is to establish how effectively study participants are able to generate a complete model using the MP software. Therefore, the researchers required a means of defining and measuring model completeness itself.

Study participants each submitted two MP models they created while following the tutorial and in response to the study task presented in Appendix C. By providing the same task to all participants, researchers established a uniform basis for evaluation. As an

evaluation, the models created by study participants received a numeric score through the assessment of the degree to which each model complied with the elements of the rubric.

The model created during the tutorial and collected as part of each participant submission served as a baseline for MP modeling while following explicit step-by-step instructions. A tutorial model with a very low score might indicate a fundamental failure to understand the tool purpose or function.

Analysts averaged the model grades among the users within each subject group to record an overall performance grade to be compared as a measure of user performance within the tool.

4. Potential Software Improvements

The free-text questions found in Appendix B provided subjective characterization of user experience but did not allow rigorous comparison between the user groups. Observations from the recorded modeling sessions likewise failed to yield comparative metrics among the subject groups. However, in instances where similar comments were reported by study participants, the study team highlighted those comments and reported them for completeness. The team documented recommendations for MP software improvements for use by the development team for the platform. The de-identified feedback responses appear in Appendix D.

THIS PAGE INTENTIONALLY LEFT BLANK

IV. RESULTS AND DISCUSSION

A. DATA ANALYSIS

At the time of this analysis, 13 subjects had completed the entire usability study protocol. As a result, there was insufficient data to determine statistical significance in the reported outcomes. However, this initial analysis forms a baseline against which future usability studies of MP may be compared.

Per the study design, analysis divided respondents into two groups:

- N - no prior modeling experience (0 years)
- S – some prior modeling experience (greater than 0 years)

The N group contained six participants while the S group contained seven participants. Participants with at least some prior modeling experience reported between a few months and 20 years of experience.

1. User Satisfaction

Upon compilation of all available survey responses, the usability study found that MP received an average System Usability Scale score of 51.2 out of 100 possible. When compared with the average of thousands of previously studied SUS scores, MP's reported SUS score falls well below the score of 68, which has been shown to indicate acceptable usability.

Figure 10 presents a comparison of the calculated average SUS scores for the two primary groups. The group with no prior modeling experience reported an average SUS of 47.9 which was below the overall average score. For the group of users with some experience, the SUS scores yielded an average of 53.9 which was above the overall average score.

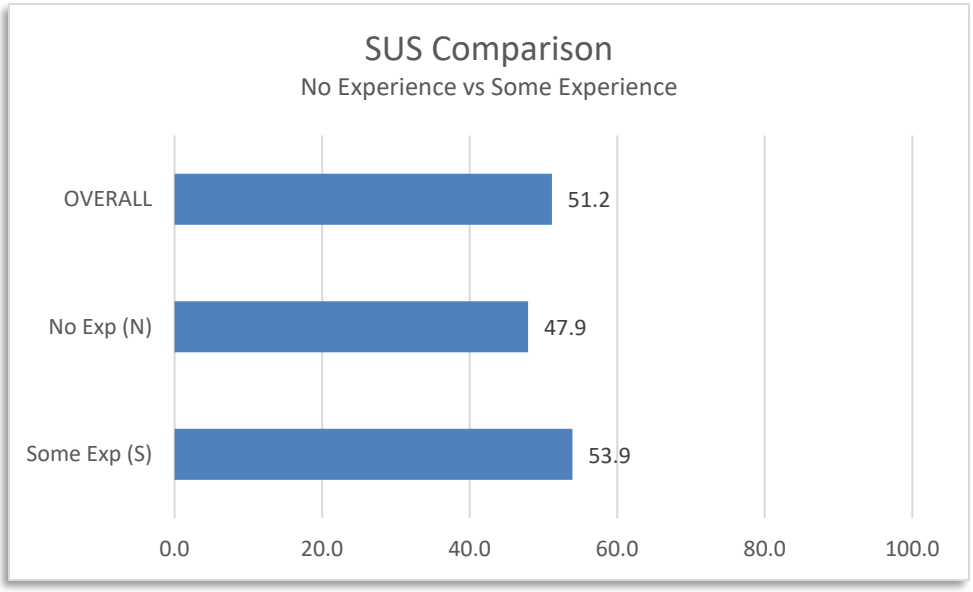

Figure 10. Average SUS Scores for Users with Different Prior Modeling Experience

The Net Promoter Score survey yielded an overall average score of -30.8 while the two study groups N and S reported averages of -66.7 and 0.0 respectively. Figure 11 illustrates this NPS comparison. It should be noted that NPS scores can range between -100 to +100 or all detractors to all promoters.

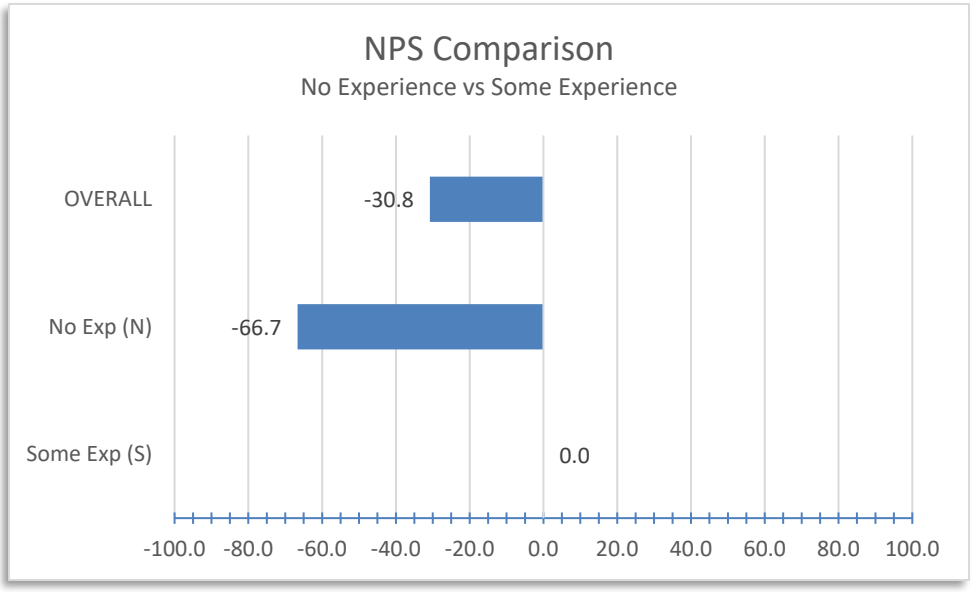

Figure 11. Average Net Promoter Scores for Users with Different Prior Modeling Experience

As an excursion, respondents were grouped according to their responses to some other entrance survey questions. In these questions, the survey asked participants to indicate their:

- preference for graphical modeling languages (Variant A)
- comfort with coding or learning to code (Variant B)
- personal view that programming was not their strong suit (Variant C)

As these questions presented a Likert scale, responses ranged from strong disagreement (1) to strong agreement (5). To divide participants into groups for each question, agreement (4) or strong agreement (5) with the question placed the respondent into an affirmative group while neutral (3), disagreement (2) or strong disagreement (1) with the question placed the respondent into a negative group.

Table 3. Study Group Variations Based on Entrance Survey Question Responses

Variant	Survey Question Grouping	Likert Scale Response	# of Participants
A	Graphical Pref	> 3	5
	No Graphical Pref	< 4	8
B	Comfortable Coding	> 3	5
	Not Comfortable Coding	< 4	8
C	Programming Not Strong Suit	> 3	7
	Programming Strong Suit	< 4	6

Figure 12 presents average SUS scores calculated for each of the study group variations shown in Table 3.

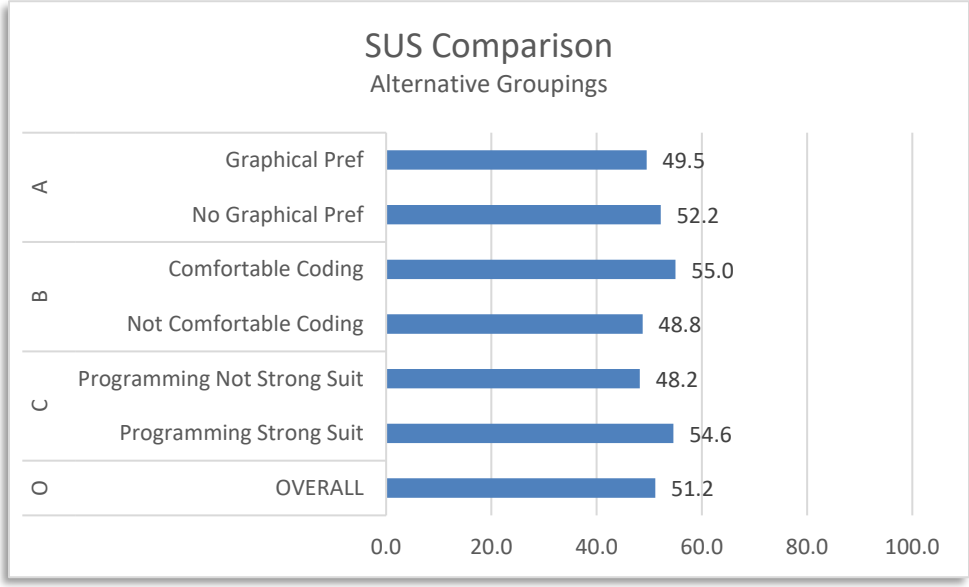

Figure 12. Average SUS Scores for Study Group Variations A, B, and C

Net Promoter Scores for these study group variations were also calculated, and Figure 13 presents the comparison of these average scores as well as the overall average NPS. Table 4 presents the full statistical calculations for these user satisfaction metrics.

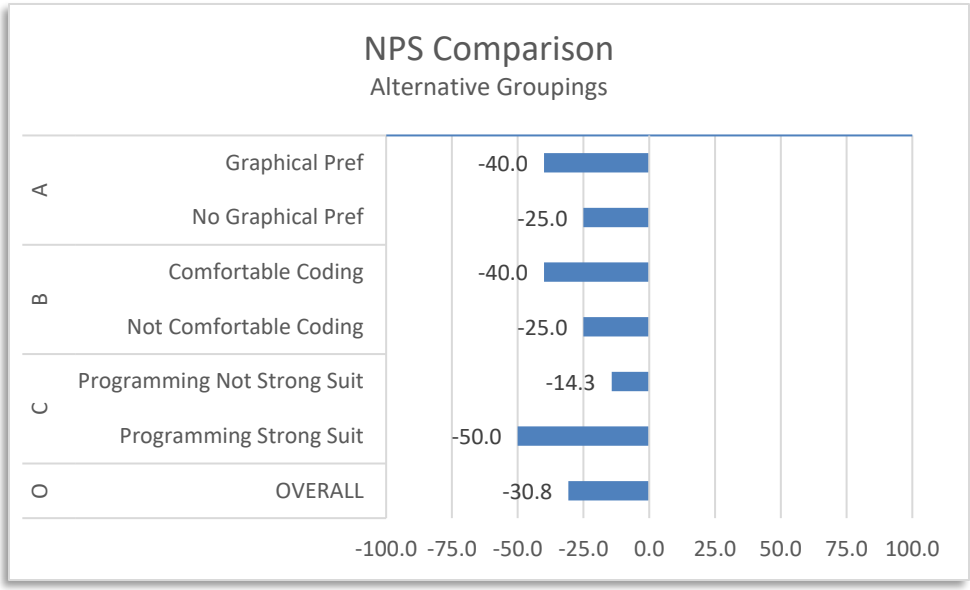

Figure 13. Average NPS for Study Group Variations A, B, and C

Table 4. User Satisfaction Statistics

	Study Group	Population Size	SUS (Avg)	SUS (StdDev)	NPS	Promoters / Detractors
P	Some Exp (S)	7	53.9	4.0	0.0	2 / 2
	No Exp (N)	6	47.9	8.4	-66.7	0 / 4
A	Graphical Pref	5	49.5	4.8	-40.0	1 / 3
	No Graphical Pref	8	52.2	8.1	-25.0	1 / 3
B	Comfortable Coding	5	55.0	3.5	-40.0	0 / 2
	Not Comfortable Coding	8	48.8	7.6	-25.0	2 / 4
C	Programming Not Strong Suit	7	48.2	8.0	-14.3	2 / 3
	Programming Strong Suit	6	54.6	3.3	-50.0	0 / 3
O	OVERALL	13	51.2	6.9	-30.8	2 / 6

2. Efficiency

The efficiency metric for this study comprises two averages: the referral score or percentage of time spent referring back to the tutorial, and the NASA Task Load Index (TLX). Figure 14 depicts the comparison of the average percentage of time participants in each grouping spent referring back to the tutorial provided with the study. Within the

primary study grouping based on modeling experience, respondents with no experience reported a referral score of 35% while respondents with some experience reported 29% on average. These values have no baseline for external comparison and were calculated only to serve as a relative comparison within each grouping.

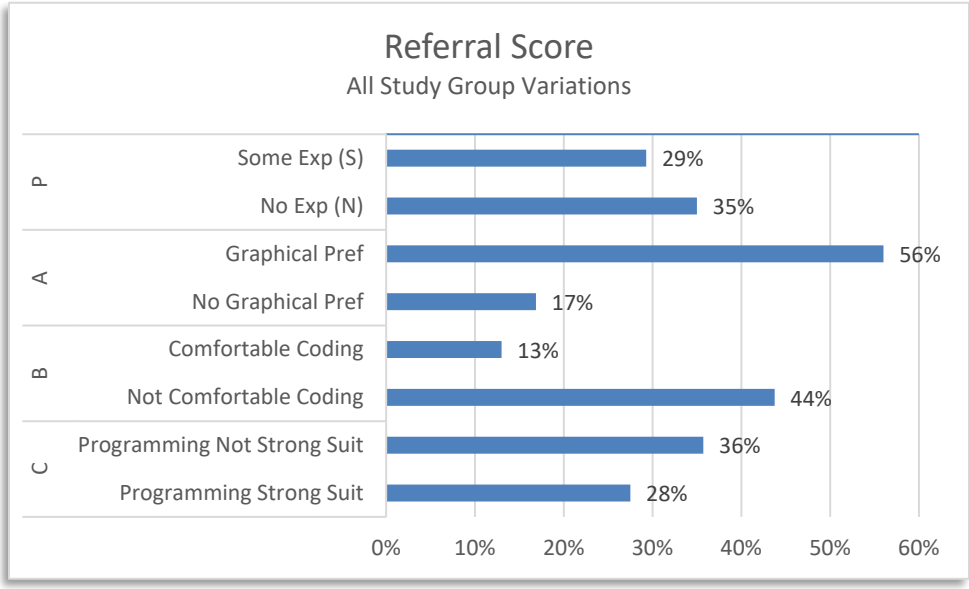

Figure 14. Average Referral Score Across All Study Group Variations

Figure 15 presents the raw TLX averages for comparison within the primary study groups based on modeling experience as well as within the study group variations determined by the entrance survey question responses presented in Table 3.

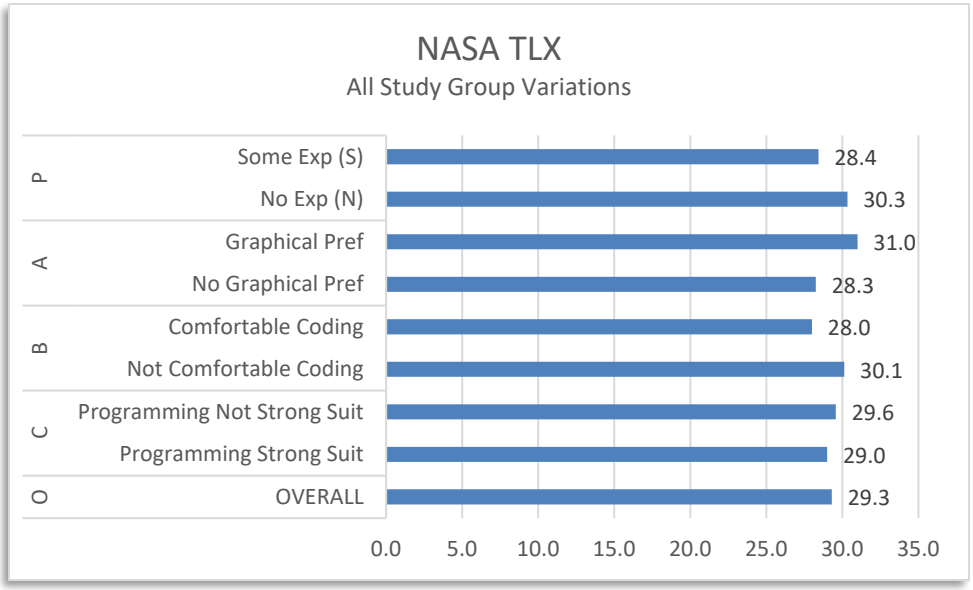

Figure 15. Average NASA TLX Across All Study Group Variations

Within the primary study groups (P), study participants with no modeling experience yielded an average TLX value of 30.3 while participants with at least some modeling experience resulted in an average TLX of 28.4 where higher values indicate a greater perception of workload. A full listing of the statistical data for all groups is shown in Table 5.

Table 5. Efficiency Statistics

	Study Group	Population Size	TLX (Avg)	TLX (StdDev)	Referral (Avg)	Referral (StdDev)
P	Some Exp (S)	7	28.4	5.5	29%	35
	No Exp (N)	6	30.3	4.5	35%	36
A	Graphical Pref	5	31.0	4.4	56%	44
	No Graphical Pref	8	28.3	5.3	17%	14
B	Comfortable Coding	5	28.0	4.6	13%	8
	Not Comfortable Coding	8	30.1	5.4	44%	39
C	Programming Not Strong Suit	7	29.6	5.5	36%	34
	Programming Strong Suit	6	29.0	4.8	28%	36
O	OVERALL	13	29.3	5.0	32%	34

3. Task Effectiveness

Models created by study participants received grades according to the rubric described in Table 2. The average grade for the study group that indicated no previous modeling experience was 76.5 while the average for the more experienced group was 77.4. These averages and the overall average model grade are shown in Figure 16.

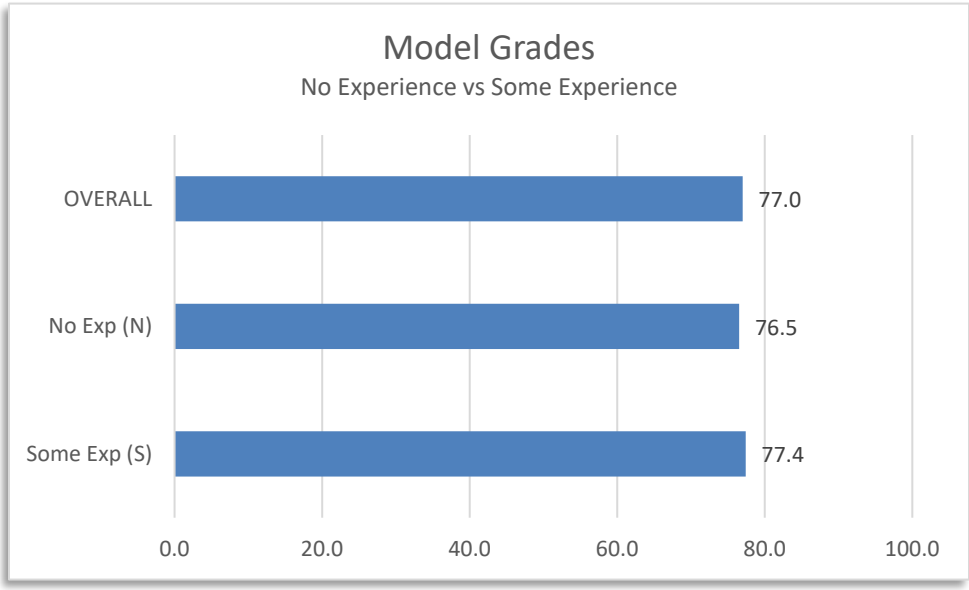

Figure 16. Average Model Grades for Experienced and Inexperienced Modelers

As with the other usability metrics, additional analysis of the model grades provided alternate comparisons based on the study variations from Table 3. A breakdown of these alternate groupings is presented in Figure 17 while the full statistical data for these comparisons is shown in Table 6.

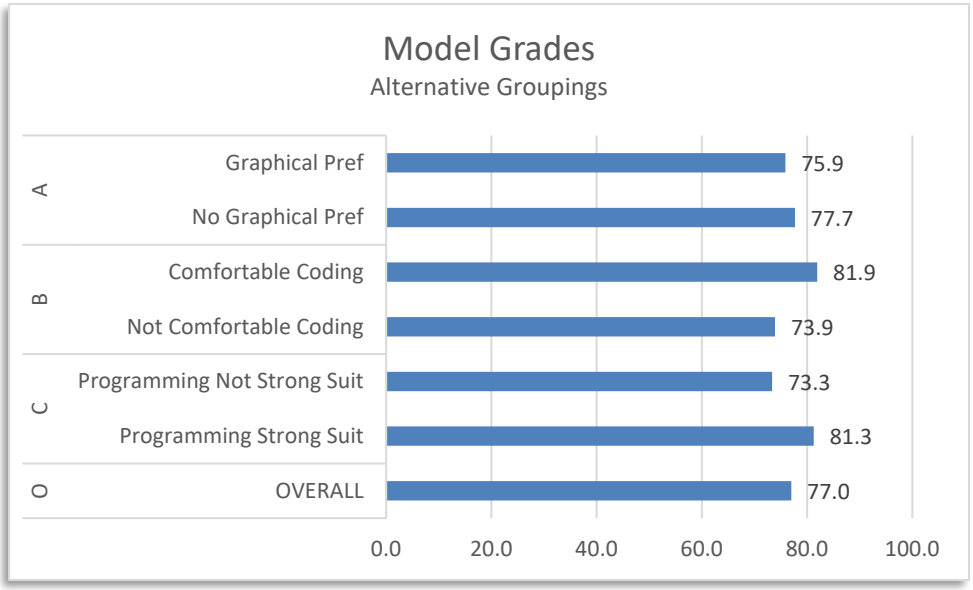

Figure 17. Average Model Grades for Study Group Variations A, B, and C

Table 6. Task Effectiveness Statistics

	Study Group	Population Size	Grade (Avg)	Grade (StdDev)
P	Some Exp (S)	7	77.4	10.7
	No Exp (N)	6	76.5	12.7
A	Graphical Pref	5	75.9	10.4
	No Graphical Pref	8	77.7	12.2
B	Comfortable Coding	5	81.9	9.5
	Not Comfortable Coding	8	73.9	11.5
C	Programming Not Strong Suit	7	73.3	12.4
	Programming Strong Suit	6	81.3	8.6
O	OVERALL	13	77.0	11.2

B. OBSERVATIONS

1. User Satisfaction

With an average SUS score of 51.2 that is well below the historically observed average of 68 found with many other systems previously studied, Monterey Phoenix

appears to be in need of usability improvements. Between the groups analyzed, inexperienced modelers reported a lower average SUS score of 47.9 than modelers with some experience who reported a SUS score of 53.9. This result appears to confirm the hypothesis that more experienced modelers report greater satisfaction and usability with MP. However, the reader should note that neither group reported a SUS score at or above the historical average of 68. Further, with such small sample sizes of less than 10 respondents for each group, definitive conclusions should be reserved until more sample data points are gathered.

Similarly to the SUS result, the Net Promoter Score of -30.8 for MP indicates that those surveyed are unlikely to recommend the use of MP to their colleagues. Many usability practitioners consider a positive NPS to be broad indicator of user satisfaction, while a negative score as in this case indicates that the user base is likely to discourage growth of use. With the primary study groups, those with no experience presented an NPS of -66.7 with no respondents as promoters and 4 of the 6 in the group as detractors. The experienced modeling group responded more positively with an NPS of 0 having two promoters and two detractors respectively. Once again, results indicate that the more experienced modelers do report greater satisfaction and usability with MP when compared with inexperienced modelers. Of course, a definitive comparison requires a larger sample size to demonstrate statistical significance.

With the excursions into alternative study groupings, the results appear to confirm what intuition might suggest. For users who entered the study with a preference for graphical modeling tools and languages, exit survey results yield a SUS average of 49.5 while those without a graphical preference reported a SUS average of 52.2. Likewise, users with a graphical preference provided an NPS of -40 while those without a preference yielded an NPS of -25, slightly less likely to be a detractor. Given that MP features text-based rather than graphical modeling, one might expect an even greater difference in these results.

Similarly, users who reported being comfortable coding generated an average SUS of 55.0 while those who did not reported an average SUS of 48.8. The same groups presented an NPS of -40 and -25 respectively which is somewhat interesting because the

coding-friendly group appears even less likely to recommend the code-based MP toolset than those uncomfortable with coding. This may confirm the golf analogy identified earlier where coders actually experience greater difficulty using the MP high-level architecture language due to their familiarity with programming languages that follow different conventions.

The groups compared based on their perception of programming yielded very similar results as the previous code-comfort groups. With a SUS average of 48.2, those who did not feel programming was their strong suit indicated less satisfaction than others who provided an average SUS score of 54.6. Once again and counterintuitively, those who presumably felt more confident in their programming skills generated an NPS of -50 and were far less likely to promote MP than their less confident counterparts who reported an NPS of -14.3.

2. Efficiency

Unlike with the SUS and NPS results, the efficiency measures cannot yield any broad conclusions in themselves. Rather these values serve as a point of comparison between the study groups.

When examining the study participants based on their modeling experience, those with no experience reported greater referral rates to the tutorial (35%) than experienced modelers who indicated 29% of time spent referring back to the help material. In the same way, the inexperienced group experienced a greater workload with a NASA TLX of 30.3 than the other group that reported a TLX of 28.4. These results indicate that the experienced group found MP to be more efficient by comparison and therefore more usable. This further confirms the original hypothesis that more experienced users express greater usability, though again, the small sample size available prevents a definitive conclusion from being drawn at this time.

Among the three variations of the study groups, outcomes largely matched expectations. Users with a graphical preference experienced a greater workload with a TLX average of 31 while those without a preference generated an average TLX of 28.3. Given the lack of a graphical interface, one might anticipate this difference. In an even stronger

confirmation, the graphical preference group indicated they had to refer back to the tutorial for 56% of the time spent performing the modeling task while the group without preference spent only 17%. It appears the lack of graphical interface greatly reduced the efficiency for users who preferred graphical modeling tools and languages.

With almost the same results, those who expressed comfort with coding experienced less workload with a TLX average of 28 while those who did not feel comfort coding experienced a workload of 30.1. The more striking result comes from the referral score where coders only found themselves referring to the tutorial 13% of the time on task while less comfortable coders spent 44% of their time looking back to the guide. These results clearly indicate that coders are much more efficient with MP than those who are not comfortable coding.

The final group who were asked about programming strength yielded less conclusive results. With a TLX average of 29.6, users who did not feel programming was their strong suit experienced only a slightly greater workload than the others who reported an average TLX of 29.0. The referral scores fell more in line with expectations as stronger programmers referred back to the tutorial just 28% of the available time while less confident programmers spent 36% of their time flipping back to the tutorial. While not as dramatic, these results also confirm that programmers appear to be more efficient in their use of MP and so find the tool more usable.

3. Task Effectiveness

As described previously, the effectiveness of the MP tool cannot readily be measured against an outside standard. Rather, a rubric developed by an experienced architect with a deeper understanding of MP modeling was used to assess how effectively new users were able to utilize the tool. The two primary study groups based on modeling experience yielded very similar effectiveness scores. Experienced modelers performed slightly better with an average grade of 77.4 than their inexperienced counterparts who achieved an average of 76.5. While this margin was very small, it does indicate that the original hypothesis is not disproved and that more experienced modelers find MP to be more effective and usable.

The other study group variations resulted in more pronounced differences in tool effectiveness. Graphical preference led to a slightly weaker performance with an average grade of 75.9 when compared with those without a graphical preference who scored very much in line with the overall average at 77.7. However, users who were comfortable coding scored significantly better with an average grade of 81.9 compared with those less comfortable coding who scored below the 77-point average with their own average of 73.9. In a nearly identical outcome, those who did not feel programming was their strong suit averaged just 73.3 points while more confident programmers exceeded the overall average with a score of 81.3. These scores suggest that those with a coding background will experience more effective outcomes with MP and so find the tool to be more usable.

C. RECOMMENDATIONS FOR MP IMPROVEMENTS

Beyond simply comparing the usability expressed by the planned study groups, the MP development team should consider the overall usability concerns identified by the study. The average SUS score of 51.2 indicates that these new users were not very satisfied with their user experience. Further, with an NPS of -30.8 these users are very unlikely to aid in the growth of the use of MP.

The nature of these scores does not provide a detailed understanding of the usability issues encountered by study participants. Other factors may have negatively influenced the participants' responses. Constraining the study protocol to only 2.5 hours may have artificially rushed participants and amplified any usability challenges. The exit survey did collect additional feedback for consideration by the development team. Without the ability to engage the study participants to discuss this feedback, however, many of the comments only point to areas for possible future study rather than specific recommendations for improvement. This points to the value of an interactive user-centered design activity as the preferred method for improving the usability of the MP software.

With these caveats, some common feedback areas worth discussing briefly in this thesis include syntax clarification and editor support as well as more helpful error messages.

1. Syntax Clarification and Editor Support

Monterey Phoenix implements a modeling language with unique key words, operators, and other language constructs. Study participants expressed a few suggestions that might aid new users in becoming more proficient with the tool and language syntax. At the most simplistic level, one user requested the addition of a short code reference accessible from the MP user interface to assist beginners. Going further, another user suggested the use of pre-defined commands or templates for creating a common object, relationship, or exchange. These templates would require only the addition of the unique name assigned to that element or event. In the most dramatic suggestion for making the syntax easier, yet another subject requested the ability to simply draw the event relationships from the root or nested events rather than having to type the constraints and precedence.

At least one user that appeared familiar with software integrated development environments (IDE) expressed a few suggestions for improving the ergonomics of the editor itself. Monterey Phoenix employs several nested layers of events surrounded with curly braces and semicolons. Enabling the editor with rainbow braces would allow a better visual identification of each nested layer as the opening and closing brackets would be set to a unique color pairing. Another suggestion sought a quick toggle button or command to comment or uncomment a line of code. In a similar way, this user proposed a feature where any text selection would immediately be surrounded by a delimiter pair when a key such as the opening or closing curly brace was pressed. Lastly, the study participant requested an auto-formatting feature that is assumed to aid with the currently manual tasks of capitalization, spacing, and indentation. Follow-up to ensure a correct understanding of these feature requests must be performed prior to software modifications.

Finally, one subject recommended the consideration of a language workbench with a "projectional editor" rather than the current text editor and compiler. The commenter provided a web reference of one such tool, the Meta Programming System (MPS) by JetBrains. More information about MPS can be found at https://www.jetbrains.com/mps/. This approach further abstracts the MP modeling language into a more plain English, domain-appropriate language that hides some of the complexity of the MP language syntax.

2. More Helpful Error Messages

Another area with a few similar improvement suggestions involved the error messages that MP presents when the compiler encounters a syntax problem such as a missing semicolon or improper use of a keyword. At least two users suggested that the error codes or messages could be made easier to understand. Ease of understanding presents its own challenges, but the addition of more detailed explanations of the cause of the compiler error could improve understanding of what is required to correct the issue. The author has experienced MP errors that may not point to the actual root cause of the error but actually take the modeler to a place further down in the code where a nested symptom of the base error exists. Improving the specificity of the error location and providing a brief explanation for how to correct the root error may address confusion and frustration introduced by misleading error messages.

Lastly, one user noted that the red error indicator occasionally presents as a false positive, meaning that no actual error exists at the location specified. Initial analysis indicates this may be related to the above problem where the root error actually exists much higher in a nested instruction. This problem might be corrected by an improved code compiler capable of better identifying the root error, but developers might also consider the IDE improvements noted earlier that requested a better way to visually identify the nested braces and semicolons that MP employs.

3. A Word of Caution

While it has been mentioned already, caution must be exercised when simply reading user suggestions and recommended improvements. Without context, the meaning of the suggestion may be misunderstood and could lead to correcting the wrong problem or a problem that did not actually exist. The prudent course for improving software usability including that of MP is the employment of a user-centered design approach with interactive participation of a small group (4–6) of representative users. Usability benchmark testing such as with the study employed by this thesis does provide valuable insights when it is compared with subsequent tests after new features or updates are released, but benchmarking tends to be a trailing indicator.

THIS PAGE INTENTIONALLY LEFT BLANK

V. CONCLUSION AND RECOMMENDATIONS

A. CONCLUSION

This thesis presented an examination of the software usability reported by a group of DOD military and civilian personnel who were exposed to the MP modeling tool for the first time. The objective of the analysis was to test the hypothesis that, among these new users, those with prior modeling background will report a better MP user experience and better software usability than those without prior modeling background.

Naval Postgraduate School students and faculty who responded to an invitation to participate in an ONR-sponsored usability study completed a carefully-constructed protocol designed to identify modeling experience level, provide a brief introductory tutorial to the MP software, present participants with a simple modeling task using MP, and capture feedback from users at the completion of the protocol. The study team developed an exit survey to structure this feedback and produce information for use in the calculation of System Usability Scale scores, Net Promoter Scores, and the NASA Task Load Index which are established measures of usability. Finally, a MP subject matter expert developed and applied a grading rubric in the evaluation of MP models that study participants produced.

Although we lack the sample size in this study to claim a finding, our results are suggestive in every single way. Analysis of the limited data gathered during the study suggests that the more experienced system modelers reported greater satisfaction and better usability with MP than modelers without prior experience. The analysis included higher SUS averages and Net Promoter Scores for the experienced group. Further, the group of experienced modelers reported a lower workload as measured by the raw NASA TLX average. Additionally, the inexperienced modelers spent more time during the modeling task referring back to the tutorial than did their more experienced counterparts. Lastly, evaluation of the models created by both groups yielded a slightly greater performance score for the experienced modelers.

Additional analysis of the data gathered appears to confirm the intuitive assessment that users with programming comfort and skills also report a greater level of usability with the text-based MP coding interface.

Given the relatively small sample size (13) of the users represented, the study team could not conduct extensive, statistical analysis of the data gathered. Even the comparisons previously presented must be viewed with the awareness of this very limited sample size and the impacts that may have on the study results. At the time of this writing, several additional respondents had indicated an interest in completing the study, so these results should be included in a final analysis for comparison.

B. FUTURE RESEARCH

The data collected for this paper is most useful for establishing a baseline of the usability of MP. As noted, while the free-text questions from the exit survey may help identify potential software improvements, the baseline scores do not identify the possible root issues with the application. In fact, at the time of this writing, none of the study participants had taken advantage of the offer of live help during the modeling task which may have helped address areas of confusion or frustration. A future study might incorporate one of the in-person usability testing methods with a small group of four to six actual MP users. This testing as part of a user-centered design methodology should yield specific areas for improvement of the software usability.

APPENDIX A. ENTRANCE QUESTIONNAIRE

A. FREE TEXT QUESTIONS

1. Name:
2. Email address:
3. Phone number:
4. Role at NPS (Student/Staff/Faculty):
5. Curriculum (enrolled in, if student; teach for, if faculty):
6. Courses taken using modeling or programming (if student):
7. Courses taught using modeling or programming (if faculty):
8. Short description of work performed at NPS, besides teaching (if faculty or staff):
9. If service member, number of years of military operational experience:
10. Number of years of modeling experience I have (fractional amounts ok):
11. Of those years, number of years experience I have with modeling architecture or high-level design (fractional amounts ok):
12. Number of Monterey Phoenix models I have created:
13. Architecture modeling tools and notations I have used (e.g., CORE, Innoslate, Magic Draw, SysML, UML, DoDAF):
14. Other modeling & simulation tools I have used (e.g., Matlab, ExtendSim, Risk Simulator):
15. Programming languages I have used (e.g., Python, C++, Java):

B. STRUCTURED QUESTIONS

Using a Likert Scale (5 response options from strongly agree to strongly disagree):

16. I prefer graphical modeling languages to text-based modeling languages.

17. I am comfortable with coding.

18. Programming is not my strong suit.

19. I can create formulas in office tools such as Microsoft Excel.

20. I am motivated to learn new modeling languages and tools.

APPENDIX B. EXIT QUESTIONNAIRE

A. SYSTEM USABILITY SCALE (SUS)

The following ten questions are derived from the original work by John Brooke in his 1996 essay, "SUS: a 'quick and dirty' usability scale."

Using a Likert Scale (5 response options from strongly agree to strongly disagree):

1. I think that I would like to use this product frequently.
2. I found the product unnecessarily complex.
3. I thought the product was easy to use.
4. I think that I would need the support of a technical person to be able to use this product.
5. I found [that] the various functions in this product were well integrated.
6. I thought [that] there was too much inconsistency in this product.
7. I would imagine that most people would learn to use this product very quickly.
8. I found the product very [awkward] to use.
9. I felt very confident using the product.
10. I needed to learn a lot of things before I could get going with this product.

B. NET PROMOTER SCORE

The Net Promoter Score question was developed and posted by NICE Satmetrix on the website netpromoter.com last updated in 2017.

On a scale of zero to ten (0 = would not recommend to 10 = highly recommend):

11. How likely are you to recommend this product to a friend or colleague?

C. NASA TASK LOAD INDEX

The questions for the NASA Task Load Index were described by Hart and Staveland in 1988 in their work titled "Development of NASA-TLX (Task Load Index): Results of empirical and theoretical research."

Using the NASA Task Load Index (10 response options from very low to very high):

12. How mentally demanding was the task?
13. How hurried or rushed was the pace of the task?
14. How successful were you in accomplishing what you were asked to do?
15. How hard did you have to work to accomplish your level of performance?
16. How insecure, discouraged, irritated, stressed, and annoyed were you?

D. FREE TEXT QUESTIONS

21. What fraction of time on the UAV modeling task did you spend referring back to the tutorial?

22. If you could change anything about this product, what would it be and why?

23. What do you like best about this product and why?

24. What kinds of questions do you see MP being used to answer?

25. How could the functionality of this product be extended?

26. Any other feedback or suggestions about your experience using this product?

APPENDIX C. UAV MODELING TASK

Open MP-Firebird (https://firebird.nps.edu) in a new browser tab. Clear the contents of the text editor (highlight all and delete). Take 30 minutes to complete as much of the following task as you can, using a timer to remind you when to stop if needed.

Create an MP model that implements the following narrative, using good commenting and indentation practices as able. Run the model after coding each step to get immediate feedback about whether it is generating the scenarios expected.

1. Mission Commander confirms the mission plan.

2. Ground Crew conducts preflight activities, which includes powering up UAV, inspecting UAV for faults or damage, and reporting flight readiness status back to the Mission Commander.

3. After the Mission Commander has verified flight readiness, Mission Commander requests launch permission from Range Control.

4. Range Control receives the launch request, and either provides permission to launch or denies the launch request.

5. If Mission Commander receives launch permission, it advises the Swarm Commander that the UAV is ready for launch. The Swarm Commander then prepares for launch.

6. Otherwise, if Mission Commander receives a denial of permission to launch, the Mission Commander delays the mission and the Swarm Commander does nothing.

If you are satisfied with your model before time is up, try inserting the following variants:

- After the UAV powers on, it provides one or more status indications back to the Ground Crew. The Ground Crew receives each status indication sent.
- During the inspection of UAV for faults or damage, the Ground Crew may find and fix some issues.
- Range Control may place the launch request on hold before deciding to permit or deny the launch request. In that case, the Mission Commander waits for the decision.

If you have a question or issue during the modeling activity, please contact Dr. Giammarco at [phone number] (0500-01300 Pacific M-F, 1600–1800 Pacific M-W).

When time is up, please save your model with name StudyID_UAV.mp and submit it as an attachment along with your StudyID_tutorial.mp model to [email].

THANK YOU FOR YOUR TIME!

APPENDIX D. USER-IDENTIFIED SOFTWARE DEFICIENCIES AND USABILITY FEEDBACK

A. SYNTAX CLARIFICATION AND EDITOR SUPPORT

"Definitely need a clear concise explanation of the syntax. I've programmed in Java and C++ but the syntax of this modeling software was difficult for me to grasp especially the (* *) and (+ +) annotations."

"Produce a one page summary of all the typical phrases."

"I would add a small code reference for beginners."

"The editor is pretty good, but could be more ergonomic. Add editor features found in IDE's:

- Rainbow brackets/braces

- cmd-/ to toggle a line's comment state

- select text, press an opening delimiter to have that selected text surrounded by that delimiter.

- auto formatting"

"Pre-defined commands only requiring fill-in names."

"If there is a way to draw the root to other sub behaviors instead of typing in the commands."

"It might be worthwhile to investigate the use of a language workbench with a projectional editor (as opposed to a text editor + parser/compiler). This would allow you to eliminate most editing errors and cognitive overhead surrounding syntax. An Open Source example of such a tool can be found here: https://www.jetbrains.com/mps/"

"The pull down examples page could have examples of each function laid in with explanations."

"If there is literature or a book explaining the syntax and how to use the tool with examples, other than the tutorial I'd like to see them."

"Short-cuts are needed."

"I enjoyed doing the tutorial so was a little disappointed when I could not get it to work in the task. I have no experience at all of coding but thought

I was beginning to pick a little bit of it up in the tutorial but was unable to translate that in to the task."

"I wasn't sure how to do if/then statements very well which I believe were covered but I couldn't recall how to stop all events if something else were to occur. I believe it was in the coordination and Share All functions."

"I like the idea of using plain language in coding, but the actual syntax to get it to work is still difficult."

"The grammar is simple and relatively intuitive; however once you add all of the rules together it gets a lot less intuitive."

"I did not get traces to work because I could not get the syntax correct."

"All errors I encountered seemed to stem from notation errors or accidentally typing an incorrect name for the event I was attempting to describe."

"Multiple OR commands were not intuitive."

B. MORE HELPFUL ERROR MESSAGES

"Explanations with the error messages could also be potentially made easier to understand."

"Make the error codes more understandable"

"Sometimes the red error indicator is a false positive for the error."

"In cases where I expected two traces to be generated for two events in a hierarchy, only one was generated."

C. BETTER EXPLANATION OF MONTEREY PHOENIX PURPOSE

"What product does this actually produce?"

"I think it would be useful to have a testing framework where you can make assertions about what it means to be a 'successful' process outcome and 'unsuccessful' outcome and have the tool identify traces that meet those assertions."

"Considering I was unable to get the basics of this product I am unsure how to answer this question."

"Requires a lot of initial knowledge about program and manpower to input requirements prior to a result."

"Expand on your description of why MP should be useful. I get the idea of revealing unforeseen behavior, but when I try to apply this to actual processes that I deal with, I don't see the use."

"I was really trying to think through all the processes I deal with daily, and which ones would benefit from mapping with MP. Unfortunately, after doing the exercise, I think it would just be a waste of my time to use this for any of my real-world processes. I just don't think there would be any ROI for the time I spend getting the process coded. I think it might be a fun puzzle to get it coded, but at the end, what would I do with the result?"

D. WHAT DO YOU LIKE BEST ABOUT MONTEREY PHOENIX?

"It's text based so I was able to stub out the pieces of the model that I wanted/needed to build. If I was able to focus and work on more examples with an instructor I suspect I could be a power user."

"I really liked the color coordination between the code and the graphs displayed by MP. It helped me to get a better idea about how the code and the produced graph were related to one another. I feel like this better enabled me to plan out what I needed to type in the code section to obtain the model I desired."

"The product has promise especially since it is using a web interface to capture modeling behavior."

"Allows simple knowledge of a system, but can present numerous paths and interdependencies that may be unseen by the planner."

"I like the idea of using plain language in coding, but the actual syntax to get it to work is still difficult."

"Immediate knowledge of an error and where the error was."

"When used correctly I can see a high value of seeing different sequence, scope, and how there can be many different combinations of the behaviors. "

"The way the language (and its editor support) express relationships and composition is comfortable and expressive."

"It graphs different scenarios nicely and considers different factors."

"I really like the color coding. It helps with the readability and understanding what pieces are tied together."

“I bet once the syntax, format, and a better facility with the basics is captured, it’s powerful for deterministic systems. For emergent behavior studies, I’d be disappointed if it didn’t have a time-dynamic aspect.”

“The grammar is simple and relatively intuitive; however once you add all of the rules together it gets a lot less intuitive. If I were given a two day class in person, I think I could use this.”

“Very streamlined - the generation of these graphs was better than what I’ve seen in Innoslate. Generating them through the use of formal modelling seems more efficient for users that are familiar with the tools.”

LIST OF REFERENCES

Albert, William, and Thomas Tullis. 2013. *Measuring the User Experience : Collecting, Analyzing, and Presenting Usability Metrics*. San Francisco: Elsevier.

Auguston, M. 1991. "FORMAN - Program Formal Annotation Language." Proceedings of the 5th Israel Conference on Computer Systems and Software Engineering. Herclia: IEEE Computer Society Press. 149–154.

Bell, B. 1992. "Using programming walkthroughs to design a visual language." PhD diss, University of Colorado. Technical Report (CU-CS-581-92)

Bergstrom, Jennifer Romano Ph.D,. 2013. "Moderating Usability Tests." Usability.gov. April 2. Accessed July 2019. https://www.usability.gov/get-involved/blog/2013/04/moderating-usability-tests.html.

Berkun, Scott. 2003. "The Art of Usability Benchmarking." October. https://scottberkun.com/essays/27-the-art-of-usability-benchmarking/.

Bias, R. 1991. "Walkthroughs: Efficient collaborative testing." *IEEE Software* 8,5, September: 94–95.

Bolt, N., and T. Tulathumutte. 2010. *Remote Research: Real Users, Real Time, Real Research*. New York, NY: Rosenfeld Media.

Bornoe, Nis, and Jan Stage. 2014. "Usability Engineering in the Wild: How Do Practitioners Integrate Usability Engineering in Software Development?" Lecture Notes in Computer Science. Aalborg, Denmark : Springer. 199–216.

Brooke, John. 1996. "SUS: a 'quick and dirty' usability scale." In *Usability Evaluation In Industry*, edited by Patrick W. Jordan, B. Thomas, Ian Lyall McClelland and Bernard Weerdmeester, 189–194. London: CRC Press.

Chartier, Donald. 1995. "Usability Begins at Home." *Software Magazine*, September: 8.

Defense Aquisition University. 2017. *Defense Acquisition Guidebook*. DAU. February 26. Accessed July 2019. https://www.dau.mil/tools/t/Defense-Acquisition-Guidebook.

Foraker Labs. n.d. "Requirements Specification." Usability First. Accessed July 27, 2019. http://www.usabilityfirst.com/about-usability/requirements-specification/.

Giammarco, Kristin. 2018. Monterey Phoenix Home - About. November 6. Accessed June 2019. https://wiki.nps.edu/display/MP/About.

Giammarco, Kristin, and Kathleen Giles. 2017. “Verification and validation of behavior models using lightweight formal methods.” Proceedings of the 15th Annual Conference on Systems Engineering Research. Redondo Beach, CA: Springer.

Giammarco, Kristin, and Mikhail Auguston. 2018. “Monterey Phoenix—Behavior Modeling Approach for the Early Verification and Validation of System of Systems Emergent Behaviors.” In *Engineering Emergence*, by Larry B. Rainey and Mo Jamshidi, 357–388. Boca Raton: CRC Press.

Hart, S. G., and L. E. Staveland. 1988. “Development of NASA-TLX (Task Load Index): Results of empirical and theoretical research.” In *Human Mental Workload*, by P. A. Hancock and N. Meshkati [Eds]. Amsterdam: North Holland Press.

Hart, Sandra G. 2006. “Nasa-Task Load Index (NASA-TLX); 20 Years Later.” Proceedings of the Human Factors and Ergonomics Society Annual Meeting 50 (9). Moffett Field, CA: Human Factors and Ergonomics Society. 904–908.

Holstein, William K., and Alphonse Chapanis. 1999. “Human-factors engineering.” Encyclopaedia Britannica. July 26. Accessed July 27, 2019. https://www.britannica.com/topic/human-factors-engineering.

Human Performance Research Group, NASA Ames Research Center. 2016a. “NASA TLX Paper / Pencil Version.” National Aeronautics and Space Administration. December 22. Accessed July 8, 2019. https://humansystems.arc.nasa.gov/groups/TLX/downloads/TLX_pappen_manual.pdf.

Human Performance Research Group, NASA Ames Research Center. 2016b. “NASA TLX Task Load Index.” National Aeronautics and Space Adminstration. December 22. Accessed July 2019. https://humansystems.arc.nasa.gov/groups/TLX/downloads/TLXScale.pdf.

Jackson, Daniel. 2006. *Software Abstractions: Logic, Language, and Analysis*. Cambridge, Mass.: MIT Press.

Kahn, M. J., and A. Prail. 1994. “Formal usability inspections.” In *Usability Inspection Methods*, by J. Nielsen and R L. Mack (Eds), 141–172. New York: John Wiley & Son.

Kelley, J. 1984. “An Iterative Design Methodology for User-Friendly Natural Language Office Information Applications.” *ACM Transactions on Information Systems (TOIS)* 2 (1), March: 26–41.

Leadetter, David. 2015. “Why Pro Athletes Struggle at Golf.” ThePlayersTribune. May 13. Accessed June 2019. https://www.theplayerstribune.com/en-us/articles/david-leadbetter-golf-swing.

Lewis, C., P. Poison, C. Wharton, and J. Rieman. 1990. “Testing a walkthrough methodology for theory-based design of walk-up-and-use interfaces.” Proc. ACM CHI’90 Conf. Seattle, WA: ACM. 235–242.

Longo, Luca, and Pierpaolo Dondio. 2015. “On the Relationship between Perception of Usability and Subjective Mental Workload of Web Interfaces.” 2015 IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology (WI-IAT). Singapore: IEEE. 345–352.

Maner, Walter. 1997. Formative Usability Evaluation. March 15. Accessed June 2019. https://web.cs.dal.ca/~jamie/teach/WaltManer/Formeval.htm.

Meiert, Jens Oliver. 2007. Revitalizing SUS, the System Usability Scale. April 23. Accessed July 2019. https://meiert.com/en/blog/revitalizing-sus-the-system-usability-scale/.

Morley, Elizabeth T. 1995. “The SilverPlatter experience.” CD-ROM Professional vol. 8 issue 3, March: 111–115.

NASA. 2016. Langley Formal Methods Program - César Muñoz - Welcome. April 10. https://shemesh.larc.nasa.gov/fm/fm-what.html.

NICE Satmetrix. 2017. NetPromoter.com. Accessed June 2019. https://www.netpromoter.com/know/.

Nielsen, Jakob. 1994. “Heuristic evaluation.” In *Usability Inspection Methods*, by J. Nielsen and R. L. Mack [Eds], 25–64. New York: Wiley & Sons.

——. 1995. “Usability Inspection Methods.” Conference Companion on human factors in computing systems. New York: ACM. 377–378.

Paul, Prantosh K, Kalyan Kumar, Dipak Chatterjee, and R. Rajesh. 2014. “Usability Engineering And User Interface Design For Electronic Information Systems And Its Subsystems: Overview.” *Information Studies* vol. 20, issue 1, January: 23–32.

Sauro, Jeff. 2011. “Measuring Usability With The System Usability Scale (SUS).” MeasuringU. February 2. Accessed July 2019. https://measuringu.com/sus/.

Scriven, M. 1967. “The methodology of evaluation.” In *Perspectives Of Curriculum Evaluation*, by R. Tyler, R. Gagne and M. Scriven [Eds], 39–83. Chicago: Rand McNally.

Sears, Andrew. 1997. “Heuristic Walkthroughs: Finding the Problems Without.” *International Journal of Human-Computer Interaction* 9(3) 213–234.

Seffah, Ahmed, and Eduard Metzker. 2004. “The Obstacles and Myths of Usability and Software Engineering.” Communications of the ACM, vol.47(12) 71–76.

Song, Songzheng, Jiexin Zhang, Yang Liu, Mikhail Auguston, Jun Sun, Jin Song Dong, and Tieming Chen. 2015. “Formalizing and verifying stochastic system architectures using Montery Phoenix, Software & Systems Modeling, Springer Berlin Heidelberg Model Driven Engineering Languages and Systems (MODELS).” ACM/IEEE 18th International Conference on. IEEE.

Thomas, Nathan. 2015. UsabilityGeek. July 13. Accessed July 2019. https://usabilitygeek.com/how-to-use-the-system-usability-scale-sus-to-evaluate-the-usability-of-your-website/.

U.S. Department of Health & Human Services. n.d. Usability.gov. Accessed July 2019. https://www.usability.gov.

User Experience Professionals’ Association [UXPA]. n.d. Usability Body of Knowledge. Accessed June 2019. http://usabilitybok.org/.

Virzi, R. A. 1992. “Refining the test phase of usability evaluation: how many subjects is enough?” *Human Factors*, 34 457–468.

Wixon, D., S. Jones, L. Tse, and G. Casaday. 1994. “Inspections and design reviews: Framework, history, and reflection.” In *Usability Inspection Methods*, by J. Nielsen and R L. Mack [Eds], 79–104. New York: John Wiley& Sons.

INITIAL DISTRIBUTION LIST

1. Defense Technical Information Center
   Ft. Belvoir, Virginia

2. Dudley Knox Library
   Naval Postgraduate School
   Monterey, California