

2009

Assessing the Accuracy of Task Time Prediction of an Emerging Human Performance Modeling Software - CogTool

Oluwakemi Damilola Adio

Louisiana State University and Agricultural and Mechanical College

Follow this and additional works at: https://digitalcommons.lsu.edu/gradschool_theses



Part of the [Construction Engineering and Management Commons](#)

Recommended Citation

Adio, Oluwakemi Damilola, "Assessing the Accuracy of Task Time Prediction of an Emerging Human Performance Modeling Software - CogTool" (2009). *LSU Master's Theses*. 1401.
https://digitalcommons.lsu.edu/gradschool_theses/1401

This Thesis is brought to you for free and open access by the Graduate School at LSU Digital Commons. It has been accepted for inclusion in LSU Master's Theses by an authorized graduate school editor of LSU Digital Commons. For more information, please contact gradetd@lsu.edu.

ASSESSING THE ACCURACY OF TASK TIME PREDICTION OF AN
EMERGING HUMAN PERFORMANCE MODELING SOFTWARE -
COGTOOL

A Thesis

Submitted to the Graduate Faculty of the
Louisiana State University and
Agricultural and Mechanical College
in partial fulfillment of the
requirements for the degree of
Master of Science

in

The Department of Mechanical and Industrial Engineering

by
Oluwakemi D. Adio
B.S., University of Ibadan, 2009
December 2013

ACKNOWLEDGMENTS

I would like to thank my major professor Dr. Laura Ikuma for her guidance, support and painstaking effort throughout this research. I wish also to thank the members on my committee, Dr. Craig Harvey and Dr. Isabelina Nahmens, for their direction and support throughout my research. My gratitude goes to Christina Koffskey and the faculty members, who conducted the previous research that was referred to throughout this study, and who willingly allowed me access to the data collected.

TABLE OF CONTENTS

ACKNOWLEDGMENTS	ii
LIST OF TABLES	iv
TABLE OF FIGURES	v
ABSTRACT	vii
1. INTRODUCTION	1
2. LITERATURE REVIEW	4
2.1 Current Regulations and Problems with Interface Design in the Petrochemical Industry	4
2.2 Human Error, Operator Capability and HSI Design	6
2.3 Cognitive Theories of Human Computer Interaction	10
2.4 CogTool	14
2.5 Other Automated Evaluation Tools for HCIs	15
2.6 Why Choose CogTool?	17
3. METHODOLOGY	21
3.1 Software/ Interface Display	21
3.2 System failures and how to address them	23
3.2.1 Pump Failure	23
3.2.2 Cascade Loop Failure	25
3.3 Dependent Variable	26
3.4 Independent Variable	26
3.5 How Speed was Measured and Recorded in Previous Study	26
3.6 Modeling the System Failures on CogTool	29
3.7 Statistical Analysis and Comparison	31
3.7.1 Research problem and Statistical test	31
3.7.2 Null Hypothesis, $H_0: S_1=S_2$	32
3.7.3 Alternative Hypothesis, $H_1: S_1 \neq S_2$	32
3.8 Overview of Experimental Variables	33
3.9 Student Participants vs. Control Room Operators	33
3.10 Guideline for Ultimate Decision Making	34
4. RESULTS	37
5. DISCUSSION AND CONCLUSION	50
5.1 Potential Limitations of Study	58
5.2 Conclusion and Future Research	60
REFERENCES	63
APPENDIX	72
VITA	82

LIST OF TABLES

Table 3. 1 Differences between the two interface designs.....	23
Table 4. 1 Summary of experimental results for the “improved” interface, in terms of means and standard deviations (shown in parenthesis)	38
Table 4. 2 Summary of experimental results for the “poor” interface, in terms of means and standard deviations (shown in parentheses)	38
Table 4. 3 Amount of data used and excluded from analysis in terms of count and percentage excluded (shown in parentheses).....	39
Table 4. 4 Summary of results of confidence interval test showing cogtool’s prediction against the 95% CI.....	46
Table 4. 5 Summary of null hypotheses.....	48
Table 5.1 Comparing participant's navigation actions against CogTool's overall estimate	53

TABLE OF FIGURES

Fig. 3. 1 Gray (“improved”) interface	22
Fig. 3. 2 Black (“poor”) interface	22
Fig. 3. 3 Steps involved in addressing pump failure	27
Fig. 3. 4 Steps involved in addressing cascade loop failure	28
Fig. 3. 5 Overview of experimental variables for 3 failure types	34
Fig. 3.6 Guideline for making final decision	36
Fig. 4. 1 Sample experimental document from previous study.....	41
Fig.4. 2 Comparing results on the “improved” interface	42
Fig. 4. 3 Comparing results on the “poor” interface	42
Fig. 4. 4 Timeline visualization for pump I failure.....	43
Fig. 4. 5 Timeline visualization for pump II failure	44
Fig. 4. 6 Timeline visualization for cascade failure.....	45
Fig. A. 1. Start cogtool and specify input devices	67
Fig. A. 2. Name the task to be modelled.....	68
Fig. A. 3. Import corresponding background images	69
Fig. A. 4. Background displaying the p-125 alarm banner blinking is shown	70
Fig. A. 5. Import unique background images for each frame	71
Fig. A. 6. All 8 frames needed for this demonstration are imported	72
Fig. A. 7. An alarm is triggered - p-125.....	73
Fig. A. 8. Select the appropriate widget for the "click" action	74
Fig. A. 9. Create another widget to represent "starting the pump"	74
Fig. A. 10. Creating transitions (movement from one widget to another).....	75

Fig. A. 11. How cogtool denotes transitions.....	76
Fig. A. 12. Each frame has been assigned a unique name; transitions are complete.....	76
Fig. A. 12. Each frame has been assigned a unique name; transitions are complete.....	77
Fig. A. 14. Beginning from the first frame, "walk through" the widgets	78
Fig. A. 13. To begin demonstration, double-click the highlighted cell as shown above.....	78
Fig. A. 15. Cogtool automatically writes the klm script.....	79
Fig. A. 16. A snapshot of the klm script on cogtool	80
Fig. A. 17. Run the simulation by clicking "compute"	81

ABSTRACT

There is a need for a human performance modeling tool which not only has the ability to accurately estimate skilled user task time for any interface design, but can be used by modelers with little or no programming knowledge and at a minimal cost. To fulfill this need, this research investigated the accuracy of task time prediction of a modeling tool – CogTool - on two versions of an interface design used extensively in the petrochemical industry – DeltaV. CogTool uses the KeyStroke Level Model (KLM) to calculate and generate time predictions based on specified operators. The data collected from a previous study (Koffskey, Ikuma, & Harvey, 2013) that investigated how human participants (24 students and 4 operators) performed on these interfaces (in terms of mean speed in seconds) were compared to CogTool’s numeric time estimate. Three tasks (pump I, pump II and cascade system failures) on each interface for both participant groups were tested on both interfaces (improved and poor), on the general hypothesis that CogTool will make task time predictions for each of the modeled tasks, within a certain range of what actual human participants had demonstrated. The 95% confidence interval (CI) tests of the means were used to determine if the predictions fall within the intervals.

The estimated task time from CogTool did not fall within the 95% CI in 9 of 12 cases. Of the 3 that were contained in the acceptable interval, two belonged to the experienced operator group for tasks performed on the improved interface, implying that CogTool was better in predicting the operators’ performance than the students’. A control room monitoring task, by its nature, places great demand on an operator’s mental capacity. This also includes the fact that operators work on multiple screens and/or consoles, sometimes requiring them to commit information to memory that they have to revisit a screen to check on some vital information. In this regard, it is suggested that the one user mental operator for “think time” (estimated as 1.2sec), should be revised in

CogTool to accommodate the demand on the operator. For this reason, the present CogTool prediction did not meet expectations in estimating control room operator task time, but it however succeeded in showing where the poor interface could be improved by comparing the detailed steps to the improved interface.

1. INTRODUCTION

There has been considerable effort in recent decades to develop complex computer-based tools that automate management of power systems, especially in the petrochemical industries. In spite of increasing automation and sophistication across process and distribution plants in this industry, the role of human operators remain essential to ensure safe, effective and efficient system operation (Avouris, 2001). Human responsibilities in many modern automated systems, according to Meshkati (2006), span such tasks like operation planning, supervision of the automated process, and intervention in unforeseen circumstances.

However, human operators have received comparatively little attention during the design process and their role in the complex task of supervision and management of pipeline networks. The need for continuous improvement in quality and productivity has increased drastically and this trend has caused plants to become more complex and more automated - a trend that is sure to increase in the future. (Thurman & Mitchell, 1995) observed that increasing the levels of automation helps the operator to successfully accomplish required tasks, and generally handles anticipated situations quite well. Unfortunately, when situations not anticipated by designers are encountered, even the most sophisticated automation often fails miserably. In such cases, operators are expected to take action, often times without fully grasping exactly what the automated system is doing (Avouris, 2001).

A good approach to address this concern incorporating operators' monitoring role is to implement performance modeling in the early design stage of complex systems. Performance modeling and analysis has been and continues to be of great practical and theoretical importance in research labs in the design, development and optimization of computer and communication systems and

applications. However, the cost of access to human performance modeling tools such as the Improved Performance Research Integration Tool (IMPRINT), Man-machine Integration Design and Analysis System (MIDAS) etc. is often a deterrent for small-scale developers and beginning modelers. Thus, this research seeks to measure performance of specific tasks on interface designs of a virtual plant called DeltaV, using a low cost modeling tool - CogTool.

A previous experiment – which forms the foundation for this study - investigated how the complexity and efficiency of the DeltaV virtual plant interface affects user performance. Two versions of the user interface were developed: “poor” and “improved” and participants were recruited, trained and then tested to determine their performance on select tasks (representative of core tasks usually carried out by operators in control rooms) in terms of its component steps and overall time. Participants completed one subset of the experiments; either performing tasks using the ‘poor’ interface or on the ‘improved’ interface.

The present study evaluated these two interfaces using CogTool - an emerging predictive human performance modeling tool that allows system designers to predict the mean time to accomplish a task. CogTool is part of ongoing research by Bonnie John, a founding member of the HCI Institute at Carnegie Mellon University. The underlying psychological theory upon which CogTool bases its prediction is known as the Keystroke-Level Model (KLM) theory – developed in the 1980s and validated through decades of research in over 100 research publication (Trewin, Richards, Bellamy, John, Swart, & Sloan, 2011). CogTool’s prediction has been reported to give predictions of task time within about 10% of empirical data (John et al., 2004), but this has not been widely validated in academic and journal articles. In the current study, CogTool was used to conduct a comparative analysis of task time predictions from the modeling tool against the human participants’ actual results to establish the validity and accuracy of its prediction.

The emphasis of this research was to assess the accuracy of task time prediction of CogTool using the virtual plant interface design as a test bed. To accomplish this objective, it was necessary to understand how the interface design encountered in operator control rooms impacts operator performance and accuracy to respond to predominant situations. For this purpose, experimental data from observations recorded using human participants to test performance, a function of time taken to complete an event consisting of several sub-tasks on DeltaV, was compared to results from CogTool's task time prediction for the same tasks.

Based on the accuracy of prediction, (i.e. within the boundary of the allowable range), CogTool will be an effective tool to identify and eliminate poor or sub-optimal workflow paths or user interface design choices in the design process, making it possible to recruit human participants in software interface design only to test optimal design choices. The 95% confidence interval of the means of the human participant data guided the acceptance (or otherwise) of the hypothesis that task time prediction on the CogTool software is accurate. The results led to recommendations for improving the virtual plant interface based on analyses of results from CogTool's performance.

The objectives of this study were:

- To assess the validity of task time predictions calculated by CogTool by comparing its predicted overall time to the actual results of the previous study (Koffskey, Ikuma, & Harvey, 2013) conducted with human participants.
- Based on the timeline visualizations generated in CogTool, to identify possible bottlenecks in the DeltaV user interface (UI) design.
- To suggest improvement/extension towards CogTool research efforts.

2. LITERATURE REVIEW

2.1 Current Regulations and Problems with Interface Design in the Petrochemical Industry

Today, pipelines and oil tankers are the major means of transporting petroleum products, with millions of barrels conveyed every day through pipelines. Oil and gasoline are the largest volume products transported nationwide, with oil extraction increasing from under 3 million tons a day in 1960 to 10 million tons daily in 2005, where it has since remained (EIA, 2011). According to the U.S. Environmental Protection Agency - EPA (Profile of the Petroleum Refining Industry, 1995), petroleum refining is one of the leading manufacturing industries in the United States, in terms of its share of the total value of the U.S. economy. As a result of high demand for petroleum and related products, increased competition among key players, and pressures to reduce operating costs while driving performance; more complex technologies are being introduced into the oil industry to meet new frontiers and develop new projects.

The question of liquid and gas pipeline safety in the petrochemical industry is a major concern, with its consequences spanning decades across several boundaries such as fatalities, property damage and environmental effects - from endangering aquatic life and vulnerable species, wildlife, death of mangroves as well as oil on the surface are being washed up onto beaches impeding bathers and tourists. Several billion dollars have been lost in the petroleum industry to major pipeline accidents because of delays in finding problems and taking appropriate corrective action (NTSB, 2005). A famous example is the April 20, 2010 BP oil spill in the Gulf of Mexico – the second-most publicized environmental catastrophe in decades (after the 1986 Chernobyl nuclear power plant) - which while concerned with oil and gas extraction rather than distribution, shares

many of the same safety and reliability issues as distribution systems, and demonstrates the significant potential for major disasters in the pipeline industry.

In the United States, the Department of Transportation (DOT) Pipeline and Hazardous Materials Safety Administration (PHMSA) published a final rule in the Federal Register (74 FR 63310) on December 3, 2009 and became effective on February 1, 2010. Tagged RIN 2137–AE64: “Pipeline Safety: Control Room Management/Human Factors”, the rule amends the federal pipeline safety regulations to address human factors and other aspects of control room management for certain pipelines where controllers use supervisory control and data acquisition (SCADA) systems (DOT, 2010). Amongst the PHMSA mandated procedural guidelines is the requirement for pipeline operators to assure that new, expanded, or replaced SCADA displays meet the provisions of the consensus standard governing such displays, API RP 1165: Recommended Practice for Pipeline SCADA Displays (Byrd, 2010). Displays for petrochemical pipelines are required to meet only some provisions of the standard. Operators are required to validate the accuracy of SCADA displays whenever field equipment is added or moved and when other changes that may affect pipeline safety are made to field equipment or SCADA displays. Pipeline operators are also required to test any backup SCADA systems and to test and verify a means to manually operate the pipeline (in the event of a SCADA failure) at least annually.

In the past, there have been measures taken to understand the role of the human operator in operation and management systems to ensure the safe transportation of hazardous liquids. SCADA systems are used to collect data from pipeline sensors, and human controllers monitor the data from remote sites for operational and safety problems. Although several factors have been attributed to the causes of accidents in the oil and gas industry, understanding human capabilities and limitations is imperative for creating an effective system design. According to Shahriari

(2006), one of the issues encountered by operators especially in the oil and gas pipeline industry is that between various display modes, there is no overall standard interface design being maintained, which leads to confusion in presentation, where messages and graphics vary from one computer display to another.

In a study to define the potential effects of advanced human-system interface (HSI) on personnel performance and plant safety, O'Hara (1997) identified several general human performance issues, amongst which are display design characteristics and information design organization. They reported that for instance, the systems do not always consider the need for information in the context of the operator's current tasks, goals and objectives or the need for feedback to the operator from computer systems' actions. Errington (2005) also showed that improving the human machine interaction (HMI) in designing the operator's user interface resulted in 41% less time for the operators to deal with events like leaks, power failures, equipment malfunction and equipment failures in an unstable plant. This agrees with a later investigation by Formosa Plastics (2007) into the sources attributable to cause of accidents in petrochemical and refining operations, which revealed that amongst other sources, operator and maintenance errors accounted for the largest cause of accidents (reported to be 41%), followed closely by equipment and design failures.

2.2 Human Error, Operator Capability and HSI Design

Human errors can be caused by many variables, such as poor interface design, lack of operator experience, communication problems, and shift fatigue. Some human-machine interaction (HMI) issues include color, alpha- numeric and text presentation as well as audible alarms which should be well thought-out when designing the graphics display. Performing two tasks concurrently is a common activity of humans in human-machine interaction. For instance, operators in control

rooms in petrochemical industries may operate a device and monitor several displays at the same time. At the core of a refining plant/pipeline system is a refinery/central production facility composed of a group of chemical engineering processes and unit operations with various components for processing crude materials into usable forms. As more automation technologies are introduced into process plants, it has become harder for the operator to know and keep up with the series of events taking place inside the ‘big black box’.

Allender (2000) observed that human performance is probably the noisiest, most variable “component” in the system, as it varies not only as a function of the design itself (e.g., the hardware and software interface, the amount of information displayed, the size or weight of the equipment), but also as a function of the context or outside environment in which the system will be used (day, night, urban terrain, extended operations) and as a function of a myriad of unobservable, internal states (e.g., aptitude, cognitive workload, stress, fear, motivation). In a study by Kim (2012), an increase in complexity led to a concomitant increased need for operators to learn to adapt to new and unfamiliar situations and improvise a solution themselves.

This work focuses primarily on downstream and petrochemical control room systems from which a sophisticated network of oil and gas refining and pipeline systems is controlled – hence, it is a demanding task, especially in the light of potential threat to equipment, the environment, the worker, and public safety – all of which peaks during these unfamiliar and unanticipated events. Predicting human performance under such task conditions at an early stage of system design can save system development teams (engineers, human–machine interface designers and even managers) a significant amount of time and cost in comparison to revising the systems at a later stage of system development (Wu & Liu, 2009).

When things go wrong, the Human System Interface (HSI) is a good place to find out which part of the system is causing the problem, and more importantly, to maintain the plant in a safe condition. Typically, the HSI has different modules for different situations. For instance, each alarm occupies a physical position in space and is directly accessible to the operator. As the complexity and the scale of the system grow, the sheer numbers of alarms become overwhelming to the operator. The design of the HSI must therefore take human capability and operator behavior into consideration (Huang, 2007).

According to Abdeen & Shata (2012), interfaces are the main tool for information hiding in software systems as they represent service contracts between users and system designers. Because of this contract role, interfaces should be designed to be more stable, especially since developing their blueprint is a sensitive task with a large influence on the rest of the product's functionalities. Interfaces should be designed to help reduce the effort required to understand them and maintain the overall functionality of the software system. In the same way, during the evolution of a software system, interface design must be accurately assessed in order to minimize the impact of any required change in the later stages of design.

Jamieson & Vicente (2001) described Ecological Interface Design (EID), a candidate framework for human-computer interface design that has the potential to fulfill these diverse roles: aid the operator's multi-tasking role, provide reliable cues and informative feedback, reduce cognitive and working memory load among others. EID is a type of framework for designing operator-machine interfaces and integrates different kinds of representations into a common interface based on two concepts from cognitive engineering: the abstraction hierarchy and; the skills, rules, and knowledge (SRK) framework. The abstraction hierarchy, a form of multi-level structure, is used to develop physical and functional models of systems as well as the mappings between them, while

the SRK framework illustrates the systematic abilities/descriptions of human performance in total, from distinctions between work situations from daily routine to stressed encounters in accidental events. The human operator draws on all three abilities to perform assigned duties in the control rooms.

SRK is used in EID to guide the design of the visual form in which information should be displayed in an interface, i.e. showing the condition of a plant in the form of balance from various levels, with the aim of decreasing cognitive burden of an operator and understanding unpredictable accidents by monitoring. The idea is to take advantage of the operators' powerful pattern recognition and psychomotor abilities, allowing people to deploy everyday skills that have been honed through evolution. Thus, EID recommends that information be presented in such a way as to promote skill- and rule-based behavior, allowing operators to deal with task demands in a relatively efficient and reliable manner.

Kim (2012) conducted a study to validate whether an EID improves operators' situation awareness in an advanced control room of a nuclear power plant (NPP). The result of their study revealed that the EID as an emerging technology is adaptable to a digitalized control room in an aspect of improving operators' situation awareness. However, beyond this, application of this concept to petrochemical industries has not been widely validated.

Although software development has evolved appreciably with novel innovations and advancement in product designs, incorporating usability of the end user has not been an easy forte. Usability, simply put refers to ease of use and learnability of a product, device or interface. Several research studies among which dates back to Bias & Mayhew (1994), have validated that adding usability to a software development program can provide up to a one hundred-fold return on investment,

with benefits to the software development organization including savings from early discovery of problems, decreased need for user support, decreased training costs, and increased sales. Similarly, benefits to end-users include increased productivity, decreased errors, and greater satisfaction.

However, while the benefits of integrating usability into software development are now recognized, in practice such integration has proven difficult. Usability concerns are often difficult to integrate into real-world software development processes. One reason for this is that usability methods have been developed out of the social sciences, psychology, ethnography and anthropology, and the methods and terminology reflects this history (Bellamy, John, & Kogan, 2011). Common usability methods include cognitive models (e.g. KLM-GOMS), task analysis and modeling, and design methods; each of which is a function of the specific need and preference of the product developer (UPA, 2010).

2.3 Cognitive Theories of Human Computer Interaction

Research in Human-Computer Interaction (HCI) which began in the 1950s has been spectacularly successful, and the applications from its findings have fundamentally changed computing. Interface improvements more than anything else have triggered this explosive growth, which has evolved from direct manipulation of graphical objects to future technologies such as gesture recognition (Myers, 1998). An approach that has been put forward in the history of human-computer interaction (HCI) is “engineering models of human performance”, which is similar in much the same way that physical designs of automobiles are tested using physical crash dummies based on solid science of human anatomy and physiology.

For HCI, an interactive system design would be tested using a computer model of human perception, cognition and motor actions based on solid psychology and data of human behavior.

These “engineering models,” dubbed *cognitive crash dummies* (John et al., 2009), would bridge the gap between the psychology knowledge of the usability experts and (sometimes) the UI designer and the quantitative approaches favored by software engineers. This is because usability experts and software engineers do not use the same terminology, which leads to communication problems. Besides, there is often a lack of integration between the two professionals, resulting in a lack of awareness of each other’s concerns when involved in team work. Because of these, questions about methodology, and participant selection, bias in testing often arises. There is a need for human performance software that is created based on theories of human cognition, yet can easily be used by professionals interested in testing the ease of use of their products.

There are several cognitive modeling methods/theories used to evaluate usability of a product or assess how long it takes to perform tasks on an interface design. These include Keystroke-Level Model (KLM), Model Human Processor (MHP), Goals, Operators, Methods and Selection Rules (GOMS), and parallel design. The KLM, which has its foundation in theory proposed by (Card, Moran, & Newell, 1986) is an approach to assessing quantitative usability and human–computer interaction (HCI). It is basically a method that involves listing the sequence of keystroke-level actions a user must perform to accomplish a task. The original KLM process was for a modeler to itemize the overt actions a user would have to take to accomplish a task (keystrokes, mouse movements, mouse clicks, etc.), place a single mental operator using heuristics derived from user data, add system response time that makes the user wait, then sum up the times associated with these operations. KLM describes task execution in terms of four physical-motor operators: **K** (key-stroking), **P** (pointing), **H** (homing), and **D** (drawing), one user mental operator **M**, and a system response operator **R**(*t*). The mental operator (denoted by “M”) is to represent all the unobservable operations a user would perform, e.g., eye movements, memory retrievals and

decisions, using a set of five heuristics defining where the Ms should appear in the model (John, 2010), although with an eye tracker device, tracking eye movement on an interface is now possible.

Goals, Operators, Methods and Selection Rules (GOMS) is a cognitive modeling method/approach to modeling human computer interaction and have since been extended to include a large family of modeling techniques. It is an engineering analysis and design method based on empirically validated results from cognitive psychology. The GOMS Keystroke Level Model (KLM) is the simplest of the GOMS models and deals only with operators and methods. It assumes that the user's goal has already been formulated and the most appropriate method or approach to execute the goal has already been determined. Therefore, given a method, KLM allows a modeler to compute the performance (time to complete task) of an expert user who makes no errors. An expert user is defined as one who already knows how to complete the task well, and does not need to pause at any point to consider how best to proceed. For instance, if an interface engineer wishes to predict the interactive performance of a particular interface for an expert user, and believes that the probability of error during task performance is low, KLM provides a mechanism for doing this, as it is appropriate for things like comparison of candidate designs in situations where users will complete tasks with high frequency, and where efficiency is a critical usability criterion.

The GOMS-KLM model is designed to be as straightforward as possible and easier to use, and it is usually applied in situations that require minimal amounts of work and interaction with a computer interface or software design. To apply this model, the sequence of operations involved within a task is modeled as a sequence of a small number of operations denoted with specific codes in the form of upper case letters. For instance, a keyboard entry is denoted by K (to represent the operation “Key press and release”) and M represents “Mental Preparation” among others. Each individual operation is then assigned a duration derived from thorough psychology experiments,

and is intended to model the average amount of time an experienced user would take to perform such operation. (Card, Moran, & Newell, 1986) conducted extensive studies and were the first pioneers in investigating the average time durations for specific operations depending on the skill level of the operator and came up with the mean and range values for different operation/parameters. (Jastrzembski & Charness, 2007) introduced multipliers for adjusting for the processing times in older adults forty years and above.

ACT-R (Adaptive Control of Thought—Rational), also known as the Anderson's Theory, is one of the most important and comprehensive theories of skill acquisition. It is a cognitive architecture developed by John Robert Anderson at Carnegie Mellon University, and describes the cognitive and perceptual operations that enable information processing in the human mind (Anderson, Bothell, Byrne, Douglas, & Lebiere, 2004). These operations are basic and have been reduced to the simplest parameters. One of the major contemporary theories of skill acquisition that suggests the importance of practice as a mechanism of performance improvement, ACT-R is based on the assumption that practice provides an effect for refined procedures needed for improving tasks – hence the need to extensively train operators and ensure they are well grounded with thorough practice sessions and continuous refresher courses before and after deployment to the fields. ACT-R has been used to successfully create models in numerous cognitive domains such as learning and memory, problem solving and decision making, language and communication, perception and attention, cognitive development, or individual differences (Anderson, Bothell, Byrne, Douglas, & Lebiere, 2004). Besides being applied in cognitive psychology, ACT-R has also been used in other fields such as human–computer interaction to produce user models for evaluating different computer interfaces, education (cognitive tutoring systems) for predicting and providing help on

student difficulties, computer-generated forces, and in neuropsychology, to interpret functional magnetic resonance imaging (fMRI) data (Oyewole, Farde, Haight, & Okareh, 2011).

In the past few years, one of the interesting developments in the cognitive architecture development scene is a new HPM tool called CogTool. This tool allows interface designers to mock up an interface or computer display, demonstrate the task to be performed, and automatically synthesize an ACT-R based cognitive model of the tasks.

2.4 CogTool

CogTool is a low-cost predictive human performance modeling tool that is able to predict the mean time to accomplish a task on an interface, making it possible to compare interface designs before building them. Developed by Bonnie E. John as the principal investigator in 2004, it was created to allow user interface designers with a graphics design background to use Keystroke-Level Model (KLM) in their normal work process to assess the efficiency of proposed designs. In practice, UI designers using CogTool often express their designs in storyboards, with frames representing the states of the device, widget “hot spots” representing the interactive controls in the device, and transitions connecting widgets to frames representing the actions a user takes to accomplish a task. CogTool therefore provides a tool for making these storyboards and then uses them as a representation of the device being assessed. When a UI designer demonstrates the correct way to do tasks on a storyboard, CogTool turns the storyboards and demonstrations into ACT-R code that emulates the KLM, runs the code, and returns a prediction of the mean skilled performance time for the task on that UI.

2.5 Other Automated Evaluation Tools for HCIs

There are several other human performance modeling tools in the market depending on the purpose of use, and can be divided into three major categories: commercially available environment modeling tools, commercially available human performance modeling (HPM) tools, and academic/research HPM tools. Environment modeling refers to simulating a particular setting or surrounding and incorporating the intended users into such environment e.g. a habitat or space such as work station, vehicle, military fields, games etc. Commercially available environment modeling tools can be used to develop work environment models for comparative design evaluation or prototyping e.g. EB Guide, DI-Guy (Garzon & Cebulla, 2010; Li, 2009).

This study is interested in HPM. Commercially available HPM tools and academic/research HPM tools integrate human performance assessment elements within the modeling tools, allowing for system/interface evaluation. One good example of a commercial HPM tool is the set of three tools developed by Alion Science and Technology - a large defense modeling and simulation contractor with expertise in the development of a suite of tools for modeling human performance. These tools comprise of: (1) a Crew Station Design tool (CSDT); (2) an Integrated Performance Modeling Environment (IPME), and (3) a Total Crew Model. These have been used extensively by the military in aircraft design, ship design, and command and control systems. In addition, Alion Science and Technology has recently been commissioned to develop HPM tools for use in railway system design and provide technical expertise to the Federal Railroad Administration FRA's Human Factors Research Division, with the aim to reduce human factors that cause train, trespasser and grade crossing accidents (Alion, 2010).

Other commercial HPM tools include iGen - a patented artificial intelligence engine developed from Air Force Research Laboratory funding with most of its applications focused on military tasks (Zachary et al., 2005) and MicroSaint Sharp - a general purpose discrete-event simulation tool with an intuitive graphical user interface (GUI) and flow chart approach to modeling (Schunk, Bloechle, & Laughery, 2002). iGEN uses a revolutionary psychological model of human thought and problem solving called COGNET, and together they form a framework that implements a general capability to learn the conditions under which each of a disjunctive set of goals or actions should be taken. MicroSaint Sharp on the other hand offers a wide range of new enhancements such as optimization, flexibility and customization, making it a powerful tool for use in several applications. However, the drawback in general is that these tools are cost prohibitive.

On the other hand, academic/research performance modeling tools are freely available and based on scientific research. However, they do not all have a support system, leaving users to their own ingenuity in understanding and using the tool. CogTool is one such tool; others include IMPRINT Pro – a tri-service tool which uses Micro Saint Sharp and has the capability to examine Army, Navy, Air Force, Marine, and Joint Missions making it totally focused at military settings (Duffy, 2009) and MIDAS (Man-machine Integration Design and Analysis System). MIDAS is a 3-D human performance modeling and simulation environment, jointly developed by NASA and the U.S. Army and Sterling Software, Inc. and has a broad use that has been explored by the Army, Navy and NASA (Li, 2009). While CogTool and IMPRINT Pro are available for evaluation and use via online download, MIDAS is not.

2.6 Why Choose CogTool?

CogTool was borne out of a need to provide an easy to learn and use front end to the powerful and well-validated cognitive modeling engine, ACT-R. It allows UI designers and developers to produce more accurate keystroke-level models in substantially less time (John, Prevas, Salvucci, & Koedinger, 2004), with higher reliability, and having had only about an hour of training (John, 2010).

The modeling-by-demonstration feature within CogTool substantially reduces learning time and increases accuracy for novice modelers with no background in psychology, and shows an order of magnitude reduction in time for a skilled modeler to produce predictions on a new device and task compared to doing KLM by hand. Another advantage is that incorporating the capabilities of ACT-R makes CogTool a distinct tool. This is because an important attribute of ACT-R is that it distinguishes itself from other theories by providing the opportunity for researchers to collect quantitative data which can be compared directly with the quantitative measures obtained from human subjects (Oyewole, Farde, Haight, & Okareh, 2011) – an approach this study is employing to assess the validity of the tool.

The CogTool software package also has an easy-to-use user interface that assists users to simulate tasks including definition of the tasks and interfaces by clicking buttons to select options and filling texts in appropriate fields to name and differentiate between widgets, thus eliminating the need for users to learn a simulation language or new software package in order to model interface designs. Designers only need to scheme the UI mock-ups in a graphic format, store them as images in the computer, and then choose the corresponding widgets and transitions to demonstrate choice tasks. This is achieved mostly by mouse clicks, which significantly decreases the modelers' working

memory load in translating the prospective actions of the users of their interface designs into numerical codes and reduces potential errors in manually inputting the KLM codes. Starting from importing background images of graphic interfaces, users specify widgets on the UI (e.g., the “button”) previously, its input method (mouse click or box check), and a series of actions or operators corresponding to that object (e.g., “look at,” “think” etc.) and then demonstrate the tasks being modeled. CogTool also allows the modeler to observe the information processing state of the model during simulation, especially when demonstrating the actions as CogTool intuitively writes the script. CogTool automatically translates these actions/operators and runs the numerical code to generate a quantitative task time prediction.

In the illustrative case study that (John, 2011) conducted to develop tools for predicting the duration and variability of skilled performance without the use of skilled performers, they showed that naive users without prior simulation language programming experience can model human performance within minutes and that CogTool can save considerable modeling time. The study concluded that CogTool opens the door to predictions of skilled performance time that is both easy to obtain and rich enough to provide more useful information to system designers.

Furthermore, unlike the original research that created the KLM, using demonstration on storyboards in CogTool reduces errors compared to manually listing steps (either just keystrokes or placing mental operators), a crude method in which analysts were inaccurate when making lists of steps to do a task. This often results in the vast majority leaving out necessary steps and/or including unnecessary steps. In addition, the timeline visualization feature offered in CogTool allows UI designers to interpret their CogTool models and enables them to extract design recommendations directly supported by the psychological science underlying the models.

The following are some of the research applications where CogTool has been used to predict human performance speed:

- **Predicting the data entry speed in a flight deck task**, which involves entering the landing in a Boeing 777 Flight Management Computer (FMC) using the Control and Display Unit (CDU); a task which occurs once in every flight and at which commercial pilots become very skilled (John, Patton, Gray, & and Morrison, 2012).
- **The first use of CogTool within IBM is on a project called PERCS (Productive, Easy-to-use, Reliable Computing System)**. IBM's contract with the US Defense Advanced Research Projects Agency's High Productivity Computing Systems initiative requires that IBM demonstrate a ten times improvement in programmer productivity with new parallel programming hardware, language, and programming environment, over productivity levels in 2002. In addition, the demonstration of improvement must be delivered at the same time as the new hardware and software, which provides no opportunity to establish skilled users to test. However, since human performance models do not need running hardware and software, but can be made with information from anything from old documentation to preliminary design ideas, using CogTool as one part of the demonstration seemed a reasonable approach (Bellamy, John, & and Kogan, 2011).

One previous attempt at validating the consistency of creating models on CogTool was in a study by John (2011). The study required 100 novice modelers, who were students in a HCI Methods class at Carnegie Mellon University, to produce 600 models of task simulations on two real-world web-pages for book cataloging and collection sharing (i. e. 3 tasks each on 2 interfaces). This assignment constituted 8% of each student's final class grade. The study showed that the students were able to generate quantitative predictions of skilled time with 7% coefficient of variation (CV)

between each student's predictions. Each modeler could also extract recommendations that will increase efficiency for users of the webpage by studying the timeline comparing both models (John, 2011).

The question now is why does research need this current investigation? The study described above conducted by (John, 2011) as well as another by (Luo & John, 2005), appeared to have validated CogTool with the developer closely involved in the study. While the former compared model consistency between the modelers, the latter studied time prediction of a task performed on a PDA as modeled on CogTool and compared it to 10 (expert) user task times. The task had a specific goal: finding the opening hours of the Metropolitan Museum of Art (MET) on the PDA, and based on the functionality of the application, the authors identified 4 different methods of addressing this task. The results of the CogTool PDA prediction compared to actual users' execution time revealed an average prediction error of 3.7%. Besides, the experimental settings in which these studies were conducted have been mostly web designs and handheld interface designs, thus making it not a robust assessment.

For these reasons - ability to compare expert task time for different interfaces without recruiting participants for a think aloud study, determine task time predictions and as an alternative usability testing, amongst others - this research assessed the capabilities of the CogTool software in order to establish it as a valid HPM tool. Albeit not being widely represented, other considerations such as ease of use, reduction in learning time and extraction of design recommendations, enhanced the decision to use CogTool to evaluate the DeltaV interface by comparing task time based on its prediction to that obtained by using human participants, and to make recommendations on improvement.

3. METHODOLOGY

The foundation for this study was an experiment conducted to test operator performance metrics using an enhanced simulation interface. For this purpose, two interface types - “poor” and “improved” interface designs (Fig. 3.1 and Fig. 3.2) were developed and students and control room operators completed an experiment using the interfaces. This experiment is referred throughout in this study as “previous experiment” and involved 24 students and 4 operators who completed 3 tasks each on either interface type (poor or improved). Performance data from the 28 participants were collected from watching the video capture of each participant’s screen interactions; this formed the benchmark against which results from CogTool’s simulation were compared. The current study (referring to CogTool’s prediction of the 3 tasks on either interface) proceeded to simulate the same scenarios as used in the previous experiment on CogTool, to predict skilled performance times for each of the tasks, and then made statistical comparisons between results from both studies; an endeavor that led to gauging the effectiveness of CogTool as a predictor of human performance for user interface designs. This chapter provides the methodology of comparing CogTool’s predicted skilled time and data from the previous investigation. The details of task simulation in CogTool are provided in the appendix.

3.1 Software/ Interface Display

The DeltaV software is a virtual plant developed by Emerson Process Management. It was used to simulate a pipeline model which represents a crude unit and its supporting components. Two versions of the same interface were developed: one after the current DeltaV used in control rooms and the other was developed to test operator performance while using a poorly designed interface. Snapshots of the two interface designs are shown in Fig. 3.1 and Fig. 3.2. Table 3.1 presents a comparison of the features of the interfaces.

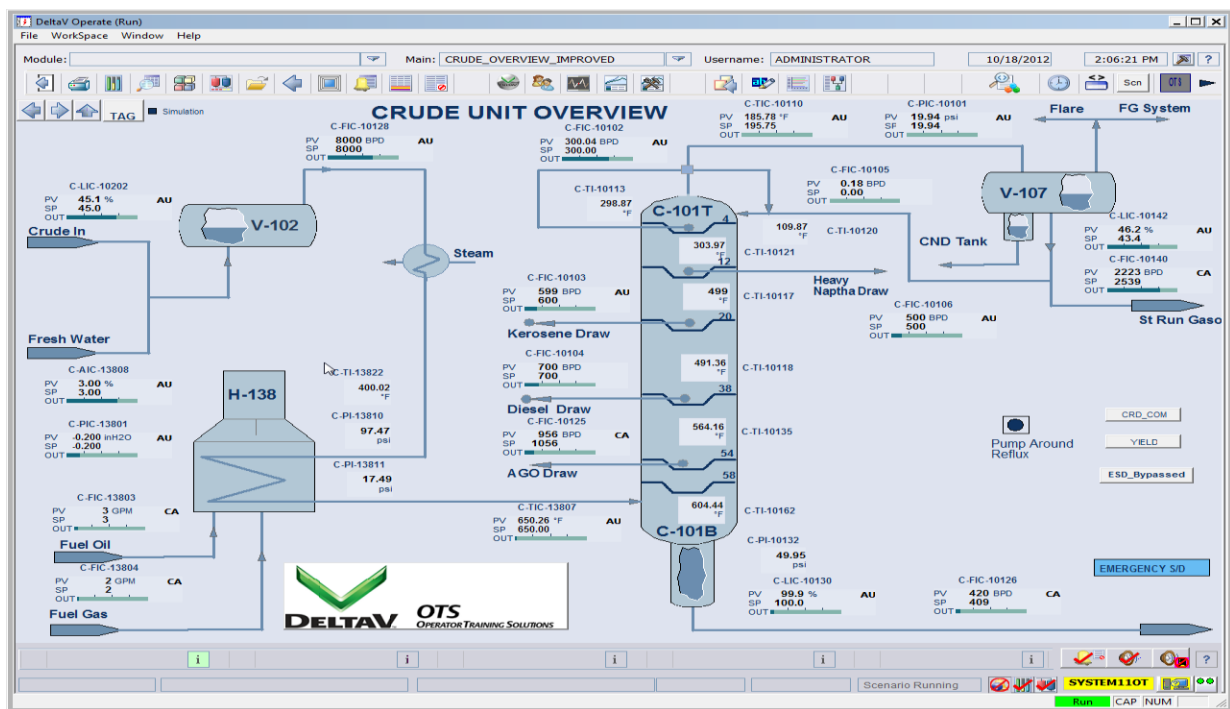


Fig. 3. 1 Gray (“Improved”) Interface

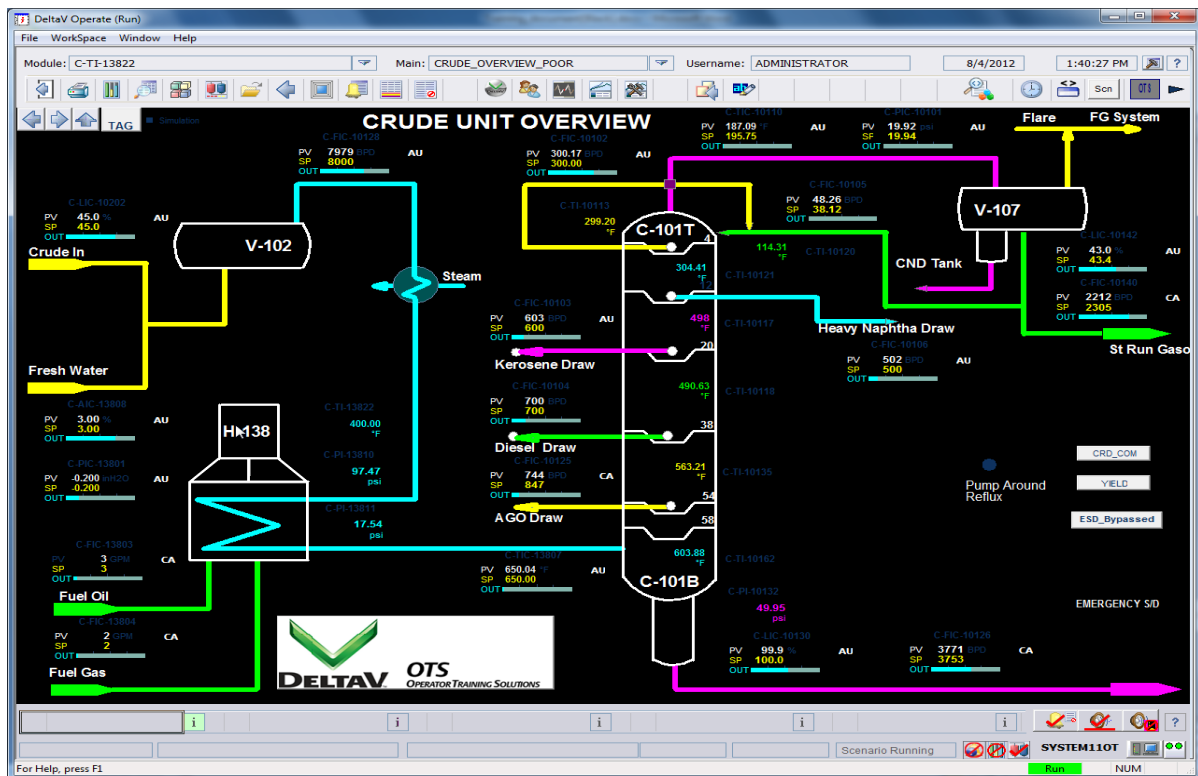


Fig. 3. 2 Black (“Poor”) Interface

Table 3. 1 Differences between the two interface designs

Improved (Gray)	Poor (Black)
Interface is based on current DeltaV with minor updates	Interface was developed for the experiment to test operator performance while using a poorly designed interface
-Minimal colors, better contrast	Color:
-Increased font size and color contrast on navigation buttons	-Color clustering not pertinent to operation -Low contrast between flow lines and background -Low contrast between text and background
Align information when possible	Flow lines not aligned
Eliminate bends in flow lines when possible	Flow lines intersect one another
Clicking on alarm in alarm bar takes user to correct screen and faceplate	Clicking on alarm in alarm bar takes user to correct screen only
Out of mode elements highlighted	No indicators for out of mode elements or pump status
Pumps show green when running, red when stopped	No indication on pump icon

3.2 System failures and how to address them

Users encountered three types of failures in the improved or poor interfaces: two variations of a pump failure (Type I or Type II) and a cascade loop failure. The specific steps needed to correct each type of failure are described.

3.2.1 Pump Failure

When a pump fails, the flow through the pump drastically drops, and the operator has the responsibility of restarting the pump and getting the flow back to the original steady state. The method outlined below describes the steps involved in addressing a pump failure (illustrations in Fig. 3.3). For the purpose of the current study, this failure is categorized into two types – I and II. Type I refers to a failure type that only requires the operator to restart the pump that failed while

Type II is a failure that requires adjusting the flow through the pump back to initial settings. Type II also causes a chain effect, usually affecting another component and making the component “fall out of” mode.

3.2.1.1 Restarting the Pump (Pump Failure Type I):

1. Alarm banner showing the name of the failed component starts blinking. Usually, the first letter of the component denotes a pump or cascade failure e.g. **P**-130A (Pump Failure) or **C**-FIC-10126 (Cascade Failure). Fig. 3.3a
2. Clicking on the alarm banner redirects operator to the correct screen where the pump is located (both interfaces) and the faceplate will appear (gray interface only) (Fig. 3.3b). In the black interface, the operator must find the correct pump.
3. On the faceplate, click on the Start button to start the alternative pump.
4. Acknowledge any alarms shown on the faceplate.

3.2.1.2 Adjusting the Flow (Pump Failure Type II):

1. Once the failed pump has been restarted (as explained in step 3 of the pump failure Type I above), the operator locates the immediate flow indicator.
2. Open the Process History View and find the original steady state value (Fig. 3.3c).
3. If the PV does not return on its own, the operator may have to manually adjust the PV to within +/- 50 of the original steady-state value (Fig. 3.3d).

Manually adjusting the PV:

Within a faceplate, there are three (3) values that are adjustable: the mode, process value (PV) and set point (SP.) The mode may be changed by simply clicking on the desired mode buttons. PV can only be changed while in Manual mode and SP can only be changed in Auto.

In Manual: Adjust the output slider, click on the output value and type in the new value, or click on the buttons with up and down arrows to increase or decrease the output in fine amounts.

In Auto: Adjust the set point slider, click on the set point value and type in the new value, or click on the buttons with up and down arrows to incrementally increase or decrease the output.

3.2.2 Cascade Loop Failure

Usually, cascade failures occur when either a component's mode has been altered from CAS to AUTO, large deviations between process value (PV) and set point (SP), and/or large spikes in SP value. For example, a control valve which works like a faucet to restrict the flow of material, an operator would adjust the opening so that the process variable (PV) matches a certain specified value called the set point (SP).

The following describes the steps involved in addressing a pump failure:

1. An alarm is triggered.
2. Clicking on the alarm banner redirects operator to the correct screen with the failed component highlighted in a bright blue rectangular box and the faceplate will appear - this happens in the grey interface. In the black interface however, controllers would have to search for the component it is in cascade with (usually by matching the label on the failed component). Locate those components; it is there where adjustments must be made.
3. Open the component's faceplate, look at the process history and determine where the original steady-state value was set. To access process history, click on the button on the bottom of the faceplate (fig. 3.4b & c).
4. Switch to manual operation (fig. 3.4d – 1).

5. Adjust output according to process history. The process history shows the steady state value when the simulation began. (fig. 3.4d – 2).
6. Acknowledge any alarms (fig. 3.4d – 3).
7. Wait for the PV to get back to the original steady state value, change mode to Cascade (fig. 3.4d – 4).

3.3 Dependent Variable

The total time operators spent in decision making and taking action, also referred to as speed, that were recorded from the time an alarm goes off (signaling the start of an event) to completion of all actions involved in ensuring a failed component is back to normal operation. These were provided from the previous study (Koffsky, Ikuma, & Harvey, 2013) using human participant. Dependent variables also include the task time predictions made by CogTool upon modeling the tasks the human participants performed, as in the previous study. It should be noted that not all participants completed all steps involved in each of the system failures. In order not to interfere with the internal validity of this study, comparison of results of both studies was based on time to complete event only for those events that were successfully addressed from when the faceplate was opened, to when the simulation was returned to steady-state.

3.4 Independent Variable

The independent variables in this investigation are the two interface displays (improved and poor), 3 task types (pump I, pump II and cascade) and 2 participant groups (students and operators).

3.5 How Speed was Measured and Recorded in Previous Study

To capture each participant's observed performance and responses during each experimental scenario, MORAE – a video capture software was used to record the time durations each operator

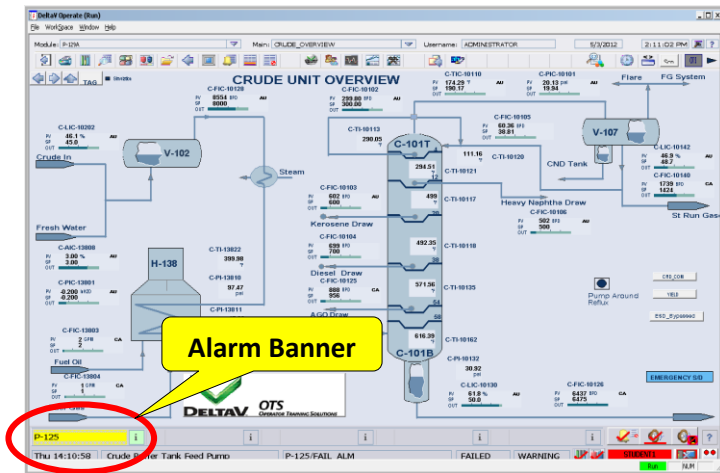


Fig. 3.3a

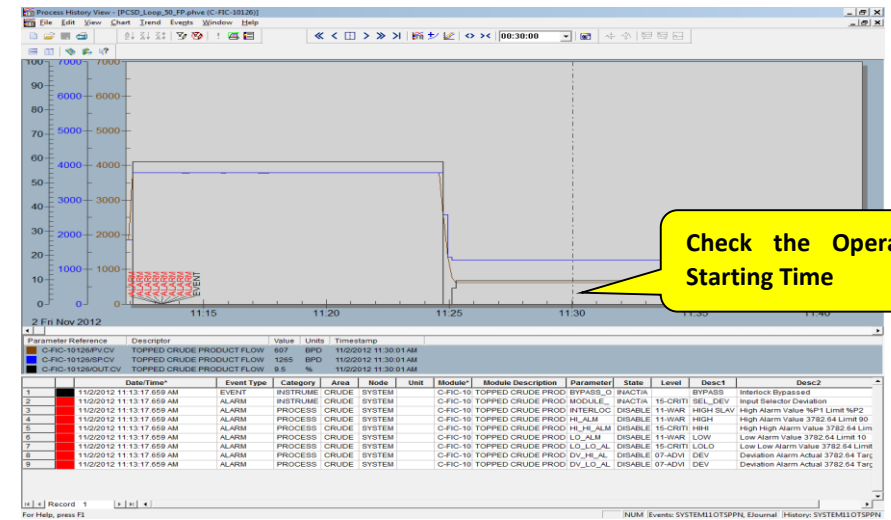


Fig. 3.3c



Fig. 3.3b

Fig. 3. 3 Steps Involved in Addressing Pump Failure

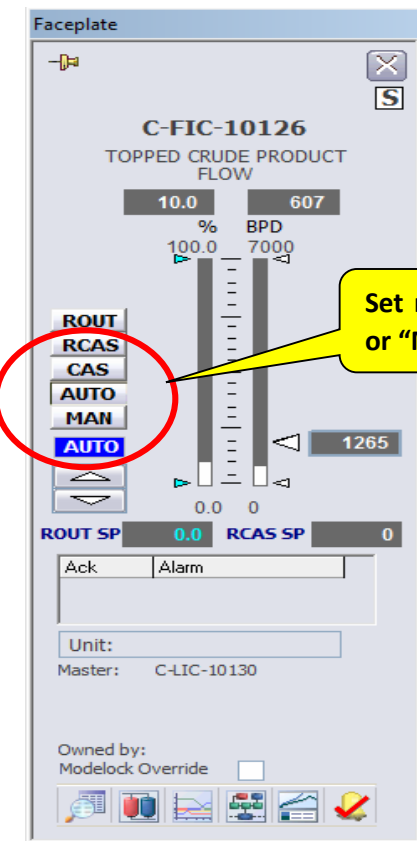


Fig. 3.3d

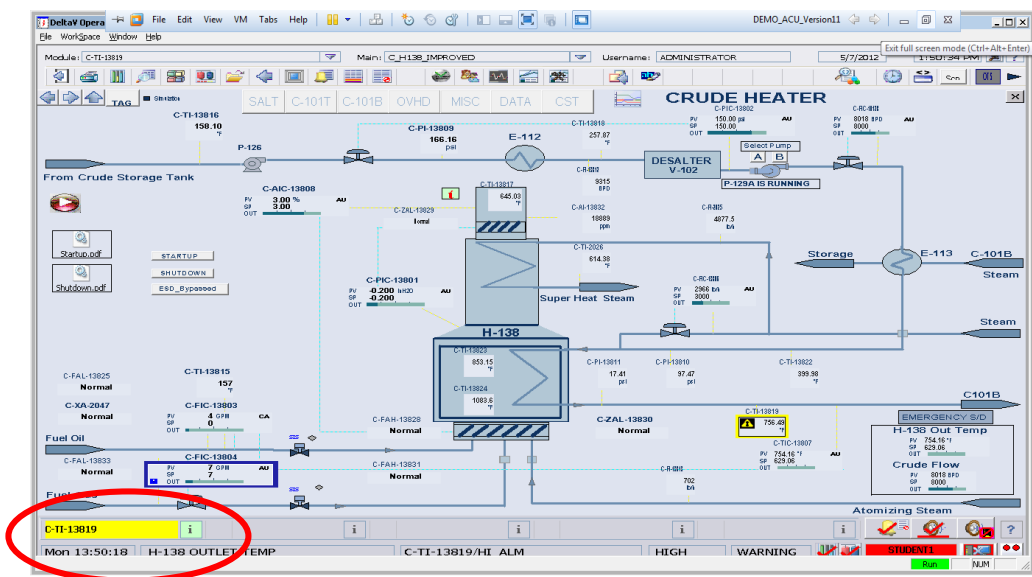


Fig. 3.4a

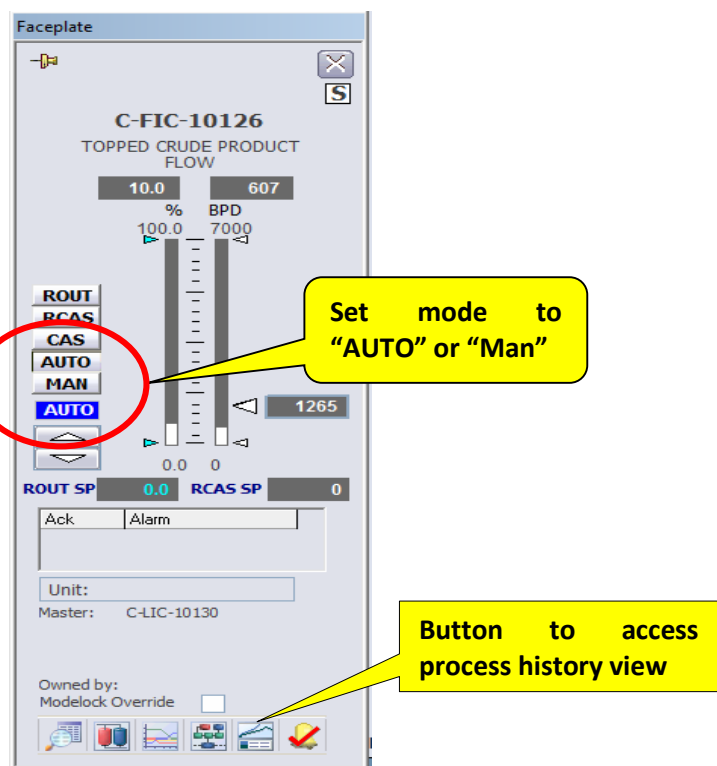


Fig. 3.4b

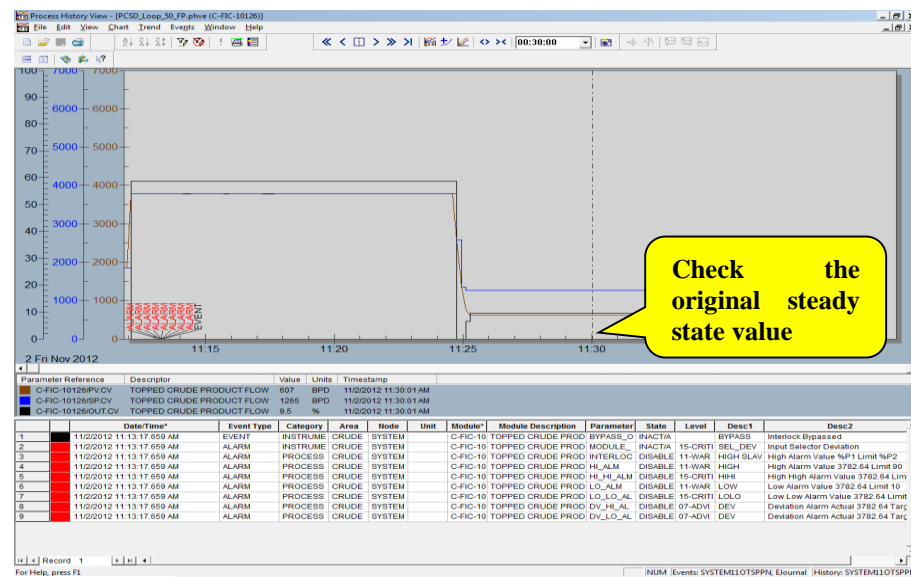


Fig. 3.4c

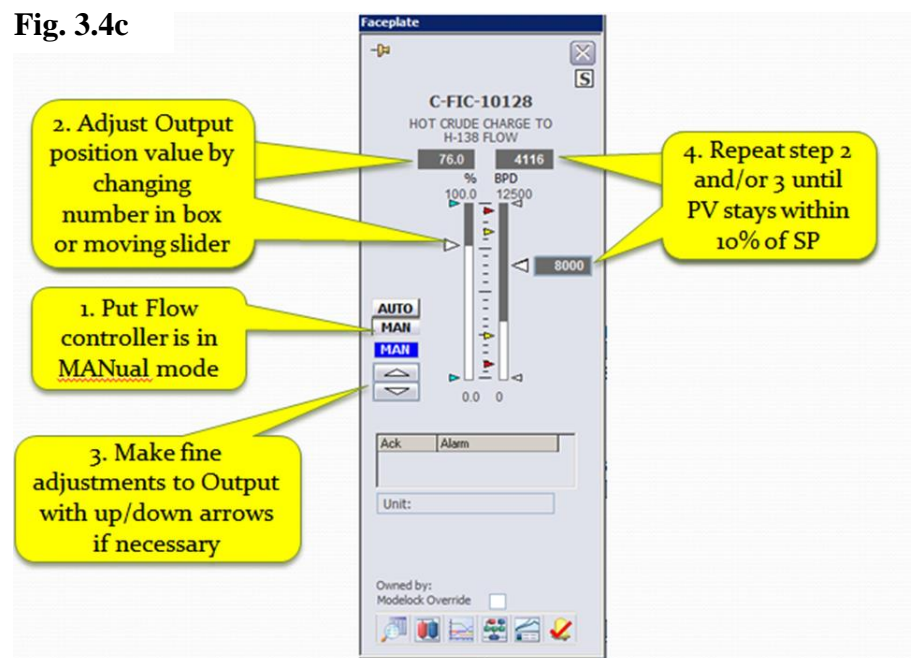


Fig. 3.4d

Fig. 3. 4 Steps Involved in Addressing Cascade Loop Failure

spent between the start of an event and its completion. MORAE (TechSmith) recorded user interactions, capture audio, on-screen activity, and keyboard/mouse input efficiently, and also allowed the experimenter to discover patterns easily.

3.6 Modeling the System Failures on CogTool

Regardless of the combinations of system failure types presented to a participant during the previous experiment, where some failures occurred in rapid succession causing overlap of system failure, this study only considered failure types that occurred in isolation: the pump failure type I, pump failure type II and cascade failure. All actions (for instance button click, window close, drag and drop etc.) required to address a particular failure for a selected interface on CogTool were simulated, by creating storyboards of actions and then demonstrated each of the component actions on the storyboards. Inputs such as the background image(s) for the interface designs, input device type (e.g. mouse click, keyboard entry, tap for touchscreens etc.) and corresponding widget were provided. Afterwards, transitions between the widgets in sequential order were shown after which CogTool runs a demonstration of the task and automatically wrote a script of the task being demonstrated including “think times” where appropriate. At the end of the demonstration, and upon confirming that all the actions required completing the task had been shown, the “Compute” button was clicked and CogTool analyzed and produced a numeric time prediction of a skilled for that task. Snapshots from CogTool that explained the general steps involved in generating skilled user task-time for addressing a pump failure on the grey interface are provided in the appendix.

Experimenter vs. Skilled User: Because of the extra steps and time needed to search for components and details on the black interface, there was a concern on what a skilled user would do, in order to simulate his actions in CogTool. It was agreed that the experimenter who conducted the previous study (Koffskey, Ikuma, & Harvey, 2013) would be a good fit. Therefore, she

completed the scenarios on the black interface and the actions she executed in addressing the system failures were recorded with Morae and subsequently used to simulate the three failure types on CogTool. Of course, there is the question of familiarity with the set-up of the interface and the sequence of the failures. This may influence how fast the experimenter moved through each failures and the time it took to address them. Given the thorough discussions that ensued before the experiment and the experimenter fully understanding the underlying reasons and cautious not to ‘outsmart’ the failures/events on the screen, it is expected that this should be minimal and would not significantly affect the outcomes. Otherwise, the time estimate generated by CogTool could be extremely shorter and outside the confidence intervals.

Simulation Start Time: Recall that to check the original steady state value on the process history view (PHV), the operator is required to retrieve the output corresponding to the time the simulation commenced. In the previous study (Koffskey, Ikuma, & Harvey, 2013), this was done by writing the start time on a post-it given to each participant, which they would quickly look at to recollect this piece of information. The potential drawback here is that this requirement was not designed in the previous experiment to be carried out on the screen, and had to be somewhat ‘revised’ to be considered in CogTool’s calculation. In order to simulate this action in CogTool, a word document was created and the start time is typed boldly in 72pts Calibri font in the middle of the page. This file was kept open throughout the simulation and the operator was simulated to “click” on the appropriate window in the taskbar area of the monitor, “look at” and “click” on the DeltaV window to go back to the experiment. The “look at” action is the same for both conditions and only the two “clicks” required in CogTool would differ. This difference however, should cancel out in the time it takes a participant to shift his attention away from the screen, read off this piece of information and go back to the screen.

3.7 Statistical Analysis and Comparison

3.7.1 Research problem and Statistical test

The purpose of the study was to determine whether there was a difference between average times observed to complete an event on an interface design when the experiment was conducted with trained human participants and when the same event was simulated in CogTool to predict skilled-user time to complete the event. The predictor (or determinant) variable is the data from the previous study (Koffskey, Ikuma, & Harvey, 2013), recorded as speed and denoting the average time from alarm trigger to when the event is completed. The criterion variable, on the other hand, is the predicted time from CogTool's simulation (current study).

Data from the previous study were analyzed to determine the mean and standard deviation of the time values quantifying the operators' speed for the events involved in each of the system failure types – Pump I & II and Cascade Failure. CogTool's simulation also yielded numeric time predictions as demonstrated on the storyboard for the actions involved in addressing these failures. Since the observations obtained from each of these studies (previous and current) were unrelated, unaffected and independent of one another, the confidence intervals test was most appropriate to guide the analysis and comparison of the two data sets.

Careful review of the data from this experiment (previous) revealed that approximately 70%, 65% and 46% of participants completed the pump failure I, pump failure II and cascade failure respectively on the grey interface while for the black interface simulation, approximately 75%, 66% and 34% (in the same order) completely addressed these failures. Watching the Morae video of the DeltaV simulation revealed that this was partly due to the fact that some participants took too much time to address a certain failure, and therefore became overwhelmed with the overflooding of other failures as they appeared in quick succession (especially for the hard difficulty

level). Hence, approximately 31.3% and 53.9% of the data for time to complete events on the grey and black interfaces respectively were missing from the data supplied from the previous study (Koffsky, Ikuma, & Harvey, 2013), solely as a result of some participants' failure to completely address those failures.

Furthermore, participants that took overly long to attend to a particular failure inadvertently recorded time to complete event that were much higher than the others (some about 300% higher). Such data are outliers, as they fall 3σ away from the mean. Including them in the set of data for calculating the average for similar failure type and interface type resulted in artificially inflating the time to address such failure. Hence, they were excluded from data used in the comparison.

3.7.2 Null Hypothesis, $H_0: S_1=S_2$

The null hypothesis tested was that there is no significant difference between results observed in operators' actual speed while addressing system failures - S_1 and the quantitative prediction of skilled user by CogTool - S_2 . The confidence interval statistical test was used to guide the hypotheses; whether CogTool's prediction falls within 95% confidence interval of the mean of the human participants' task times. The alpha level of significance is set at $\alpha = 0.05$.

3.7.3 Alternative Hypothesis, $H_1: S_1 \neq S_2$

The alternative hypothesis tested was that there exists a difference between results observed in operators' actual speed while addressing system failures - S_1 and the quantitative prediction of skilled user on CogTool - S_2 . This implied that the predicted value fell outside the 95% confidence interval, an indication that CogTool's prediction of skilled user time is inaccurate and is thus, not an effective tool for human performance modeling.

3.8 Overview of Experimental Variables

Figure 3.5 shows a graphical representation of the experimental variables. For each interface type, the three system failures: pump I, pump II cascade failure (CAS) were simulated on CogTool to determine the skilled user time to address each failure. Thus, there were twelve (12) variable combinations: two (2) interfaces designs (poor and improved), three (3) system failure simulations (cascade and pump failures I & II) and two (2) classes of participants (students and operators). As a result, the chosen statistics described above – the confidence interval test would be repeated twelve (12) times to ensure an effective analysis of results.

3.9 Student Participants vs. Control Room Operators

Careful consideration was exercised in analyzing the data for student participants who were trained specifically for the purpose of the experiment and control room operators who have had considerable number of years on the job. Because CogTool returns predictions for skilled users, there was some level of concern on whether it would be appropriate and unbiased to compare data from both studies regarding student participants. This concern was alleviated because the students in question were trained and given the opportunity to become familiar with the interfaces and expected scenarios, as well as ask questions during the training.

INTERFACE TYPE	SYSTEM FAILURE TYPE	PARTICIPANT GROUP
BLACK INTERFACE	Pump I, II & Cascade Failure	Participant # 1 - 14
GREY INTERFACE	Pump I, II & Cascade Failure	Participant # 15 - 28

Fig. 3. 5. Overview of Experimental Variables for 3 failure Types

Thus, the students were not complete novices and results from both studies were not expected to show considerable disparities for this class of participant.

However, for clarity, this study analyzed and made comparisons for the student participants and control room operators separately, because it was believed that although there were fewer operators, results obtained from their performance were likely to be consistent with less variability, especially since their accuracy was a little higher but not statistically significant ($p = .082$). They however, had significantly slower time to completion compared to the student participants (322s vs. 207s).

3.10 Guideline for Ultimate Decision Making

Sequel to deriving numerical predictions of experimental scenarios on the CogTool software, making the ultimate decision on accuracy required extreme care and caution. It was therefore proposed that the comparison of results will be conducted within subjects i.e. results from both studies would be separated for each interface type (poor and improved) and for each failure type

(pump I & II and cascade). An example is shown in Fig. 3.6 for comparing time taken to address a pump I failure type on the grey interface design as reported in both studies. The same was repeated for the pump II and cascade failure for both participant groups, resulting in twelve (12) decisions (yes/no). If there were at least four (4) “yes” out of six (6) for either student participant or control room operators on each interface, then CogTool would be considered an accurate human performance prediction tool.

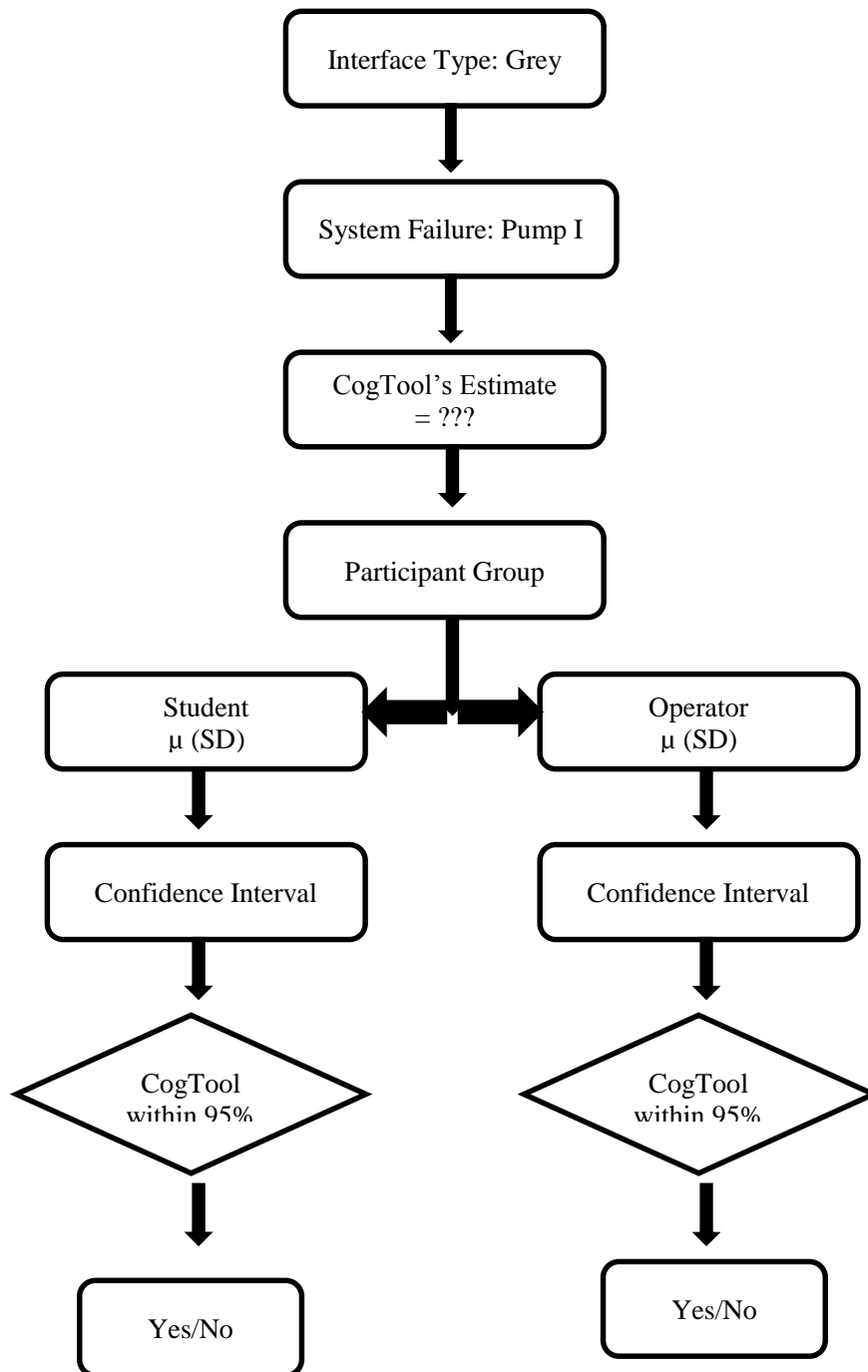


Fig. 3.6 Guideline for Making Final Decision

4. RESULTS

As discussed in the previous chapter, the three system failure types were modeled in CogTool to obtain a numeric task time prediction of what a skilled operator would do and how long it would take him to carry out specific tasks. One of the features of the CogTool software is that it automatically creates a transcript of all the physical actions (clicks, mouse movements etc.), as well as cognitive activities of the user, as the modeler demonstrates those actions required to address an event (in this case, the system failure). Afterwards, it generates and sums up all the individual times taken for each of these actions to present an overall quantitative measure of task time for that event. CogTool also allows the modeler to export the transcript generated in excel file format. The transcript lists the frame or background image on which an action is taken and the type of action, including the type of widget used. These are provided in the appendix.

The mean and standard deviations of the task times recorded for the three task types, on each of the interfaces were computed and classified according to the participant groups. An upward trend was noticed in CogTool's estimate of the skilled user task time (from Pump I to Pump II) in line with actual results from the previous study (Koffskey, Ikuma, & Harvey, 2013), except for the cascade failure's task time prediction, which was lower. However, this was in agreement with data from the human participants. CogTool's estimated task time for the Pump I system failure was 8.4sec for the improved interface in contrast to the time that the participants spent in addressing the same task – 74.8sec (34.8). Interestingly, both the students and operators took approximately 74.8sec on average, although the observation sets have different variations – 34.8secs for students and 16.7sec for operators. On the same interface, the student participants took 120.4sec (76.7) – doubling CogTool's prediction of 64.4sec - to address the pump II failure, while the operators spent 40sec more than CogTool's skilled user task time (104.3sec (89.5)). The operator's mean

time to address the cascade failure was 30sec [9.3] which was 0.5sec less than Cogtool’s prediction and 10sec less than the students, which was recorded to be 40sec (34.1).

Conversely, the variation arising from the observations on the poor interface was higher compared to the improved interface. CogTool estimated 10.3sec task time for the pump I failure while the student participants and operators were recorded to have spent 95.8sec [54.6] and 111.6sec [79.4] respectively to address the same event. In relation to CogTool’s estimate for the pump II failure (78.6sec), there was approximately 58.5sec increase in the student’s task time 126.1sec [101.7], while the operator took much longer – an increase of 48.5% above CogTool’s virtual skilled user 161.9sec [117.9]. Finally, the cascade failure was addressed in fairly the same time by both CogTool and the student participants – approximately 66sec [52.1], while the mean task time for the operators was 132.8sec [38.9]. Tables 4.1 and 4.2 present a summary of the experimental results in terms of means with the standard deviations shown in parentheses.

Table 4. 1 Summary of Experimental Results for the “Improved” Interface, in terms of means and standard deviations (shown in parentheses)

	Pump I (sec)	Pump II (sec)	Cascade (sec)
CogTool	8.4	64.4	30.5
Student	74.8 (34.8)	120.4 (76.7)	40.0 (34.1)
Operator	74.8 (16.7)	104.3 (89.5)	30.0 (9.3)

Table 4. 2 Summary of Experimental Results for the “Poor” Interface, in terms of means and standard deviations (shown in parentheses)

	Pump I (sec)	Pump II (sec)	Cascade (sec)
CogTool	10.3	78.6	67.1
Student	95.8 (54.6)	126.1 (101.7)	66.3 (52.1)
Operator	111.6 (79.4)	161.9 (117.9)	132.8 (38.9)

As seen in Tables 4.1 and 4.2, there was large variation in the data used to make the comparison for evaluating the validity of CogTool's task time prediction. In order to arrive at a reasonably acceptable data set that is sufficient enough to allow for a scientifically viable research conclusion, some observations were eliminated. Table 4.3 presents an overview of the portion of data excluded from the study and the reason for their exclusion. Data that were missing or unavailable can be largely attributed to situations where participants were unable to attend promptly to a particular failure, which led to other failures being triggered. As a result, he/she became overwhelmed with the flooding of alarms and could not address those failures before the simulation ended. For such cases, it is impossible to record task time for those events, thus corresponding observations are concluded 'unavailable/missing'.

Table 4. 3 Amount of data used and excluded from analysis in terms of count and percentage excluded (shown in parentheses)

	Failure Type	# of observations	
		Improved Interface	Poor Interface
Missing data (due to uncompleted tasks)	Pump I	16 (28.6)	16 (28.6)
	Pump II	49 (31.8)	71 (46.1)
	Cascade	40 (31.7)	94 (74.6)
Outliers (observations $\geq \mu + 3\sigma$)	Pump I	10 (17.9)	4 (7.1)
	Pump II	40 (26.0)	19 (12.3)
	Cascade	14 (11.1)	13 (10.3)
# of observations (available for analysis)	Pump I	30	36
	Pump II	65	64
	Cascade	72	19

The cascade failure was carefully analyzed before being modeled in CogTool, in that to address this failure in a timely manner before it escalates and triggers other failures, there were essentially five steps involved. While some participants couldn't attend to this failure at all and therefore no data was recorded for these cases, there were actually a few who resolved the cascade failure in remarkably short time. This they did by skipping one or two steps but ultimately completed the event. Steps skipped could be not going to the PHV screen to check the original steady state value or not waiting for the PV to be within range. But because the participant had initially opened the faceplate, he could switch the mode to manual and adjust the SP on the faceplate, then move on to other things. The previous study (Koffskey, Ikuma, & Harvey, 2013) regarded and therefore counted such method as completing the event and task time recorded.

Fig. 4.1 below shows a snapshot of sample experimental documents from the previous study, displaying the steps taken and missed. There were 18 out of 24 students who missed 2 steps on average but completed the event. CogTool's modeling was fashioned after their pattern to maintain consistency of task. Some students (6 out of 24) however followed the procedure and completed all 5 steps. This is discussed in the confidence interval results. Fig. 4.2 and 4.3 show a comparison of CogTool's estimated time to actual participants' mean task time.

Event 4 Steps:	Correct	Missed	Incorrect	Time
Open faceplate for C-FIC-10126	1			
Open PHV and check the OSS value		1		
C-FIC-10126 mode to CAS OR				
mode to manual, adjust the OUT				
mode to auto, adjust the SP	1			
PV within range, mode to CAS			1	

Event 6 Steps:	Correct	Missed	Incorrect	Time
Open faceplate for C-FIC-13803	1			
Open PHV and check the OSS value		1		
C-FIC-13803 mode to CAS OR				
mode to manual, adjust the OUT OR		1		
mode to auto, adjust the SP				
PV within range, mode to CAS			1	

	A		B		B-A		C		C-A	
Event	Event Start:		Faceplate Opened:		Time Elapsed to faceplate:		Event Complete:		Time Elapsed to complete:	
1	07	14	07	56			08	09		
2	10	22	10	34			12	24		
3	13	24	13	28			21	02		
4	18	26	19	40			19	54		
5	21	25	21	38			22	49		
* 6	26	23	26	28			26	41		
8	33	27	33	31			33	46		

Fig. 4. 1 Sample experimental document from previous study

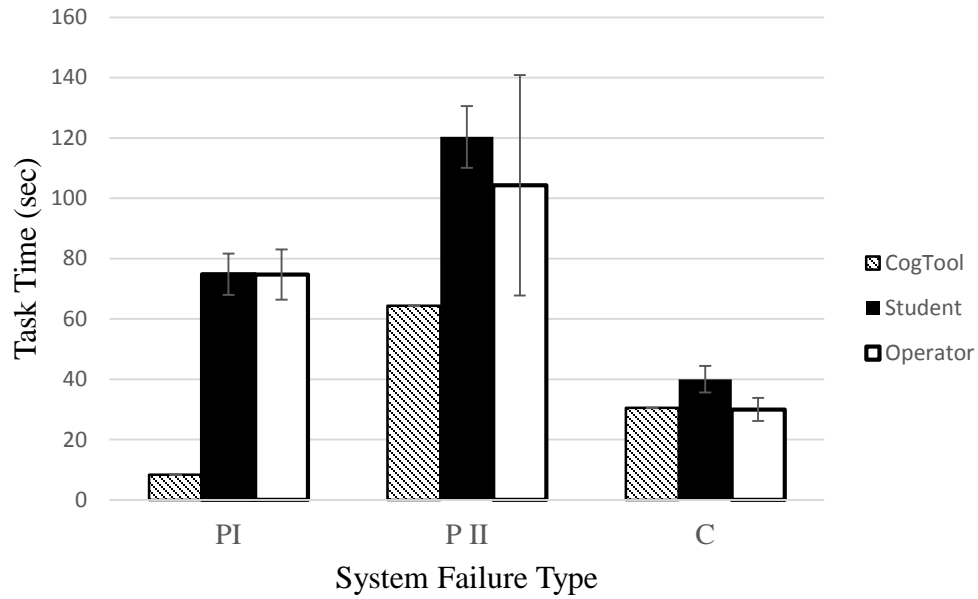


Fig. 4. 2 Comparing results on the “improved” interface

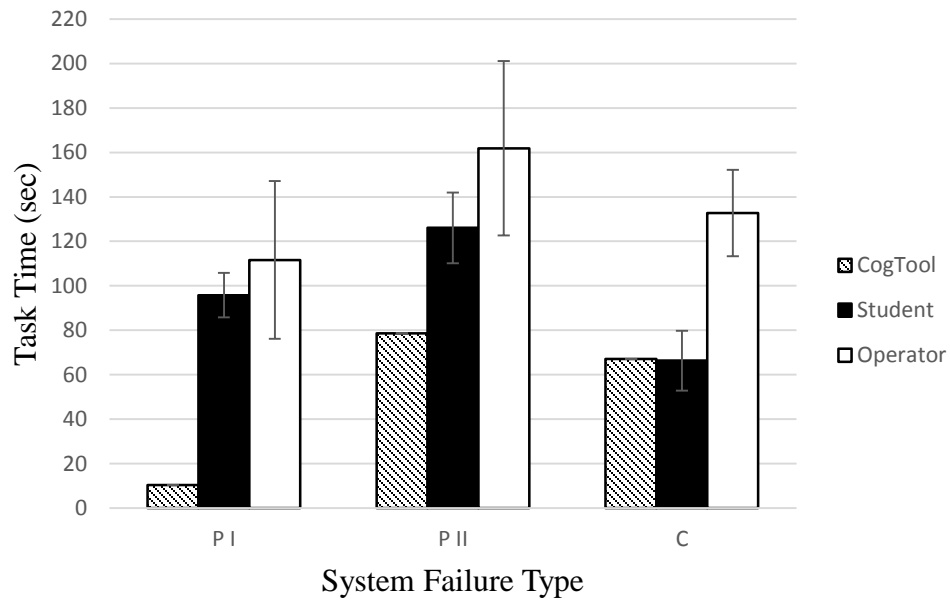


Fig. 4. 3 Comparing results on the “poor” interface

The figures below (4.3 – 4.5) are the timeline visualizations from CogTool models of the three tasks compared on both interface designs. In the three figures, the top row is for the black interface

and the bottom for grey interface. For Fig. 4.3, call-outs represent similarities in actions on both interfaces, and it is obvious that the black interface requires more steps to be taken.

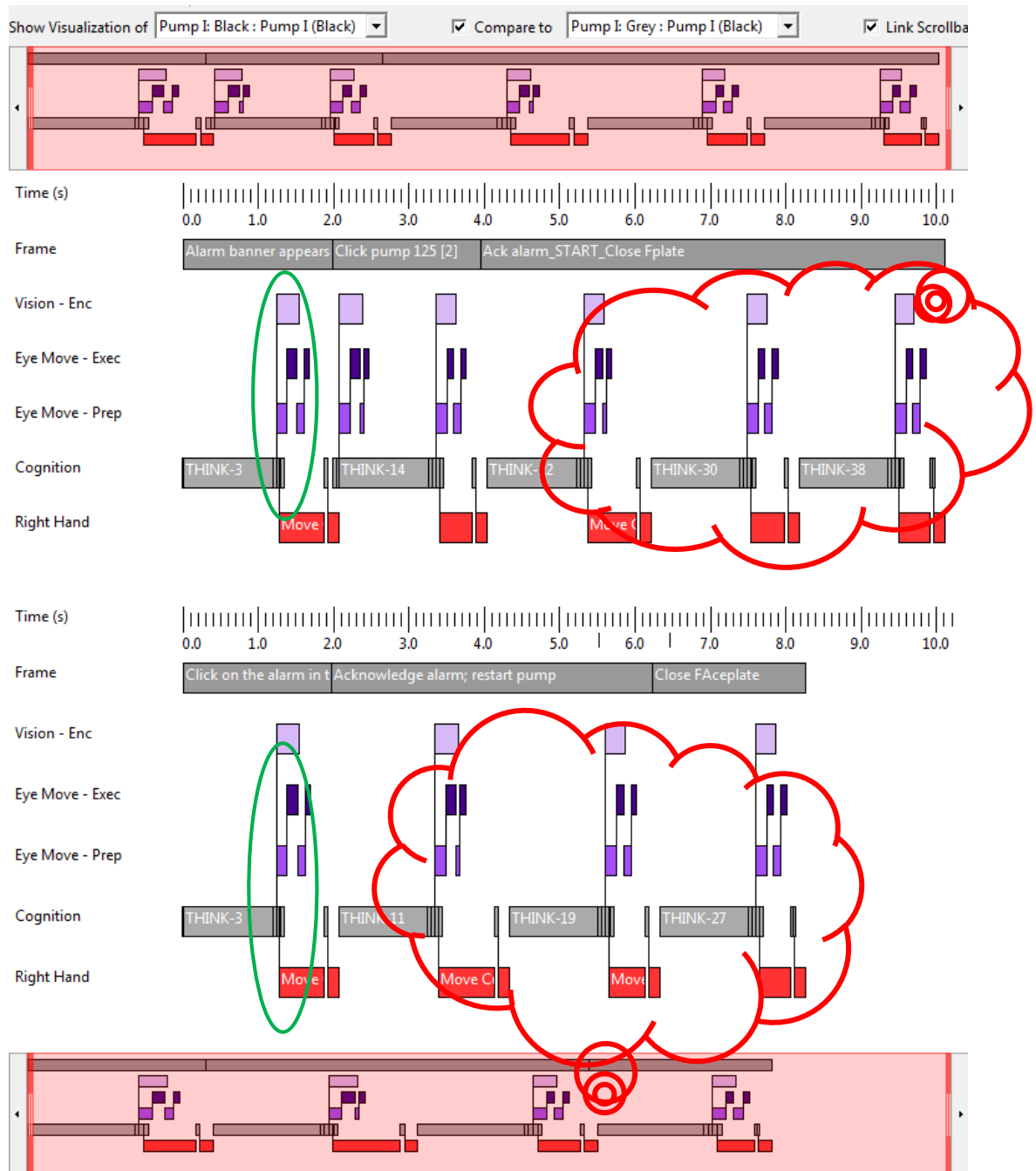


Fig. 4. 4 Timeline Visualization for Pump I Failure

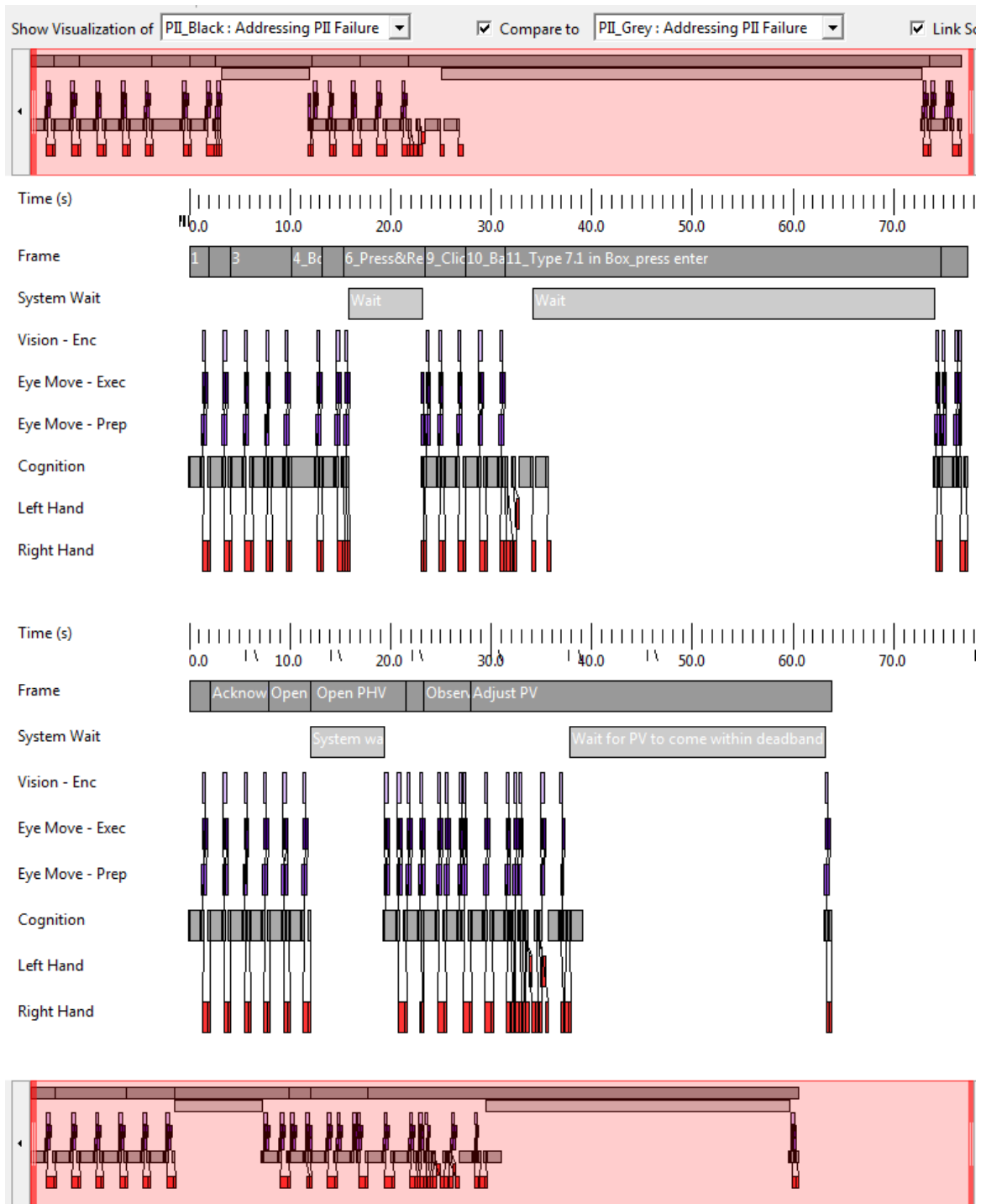


Fig. 4. 5 Timeline Visualization for Pump II Failure

For Fig. 4.5, the grey interface appeared to have more actions taken than on the black interface. Recall that the tasks on the black interface were modeled after the experimenter's actions and she did not visit the word document page to check the original start time, thus, eliminating three steps

(“click” on the appropriate window in the taskbar area of the monitor, “look at”, and “click” on the DeltaV window to go back to the experiment).

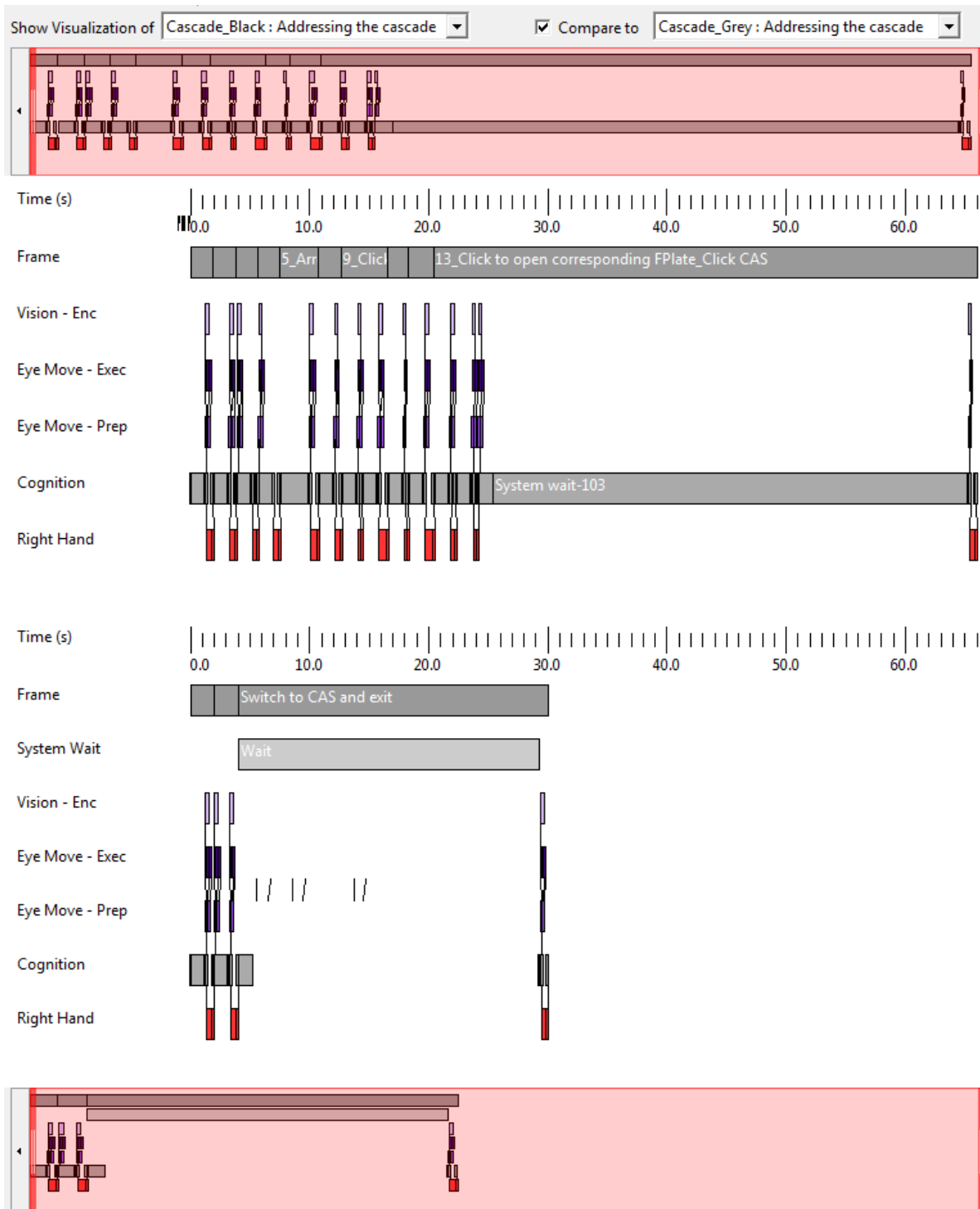


Fig. 4. 6 Timeline Visualization for Cascade Failure

The confidence interval (CI) – a statistical method which gives an estimated range of values within which a certain parameter is expected to fall - was used to test the differences in CogTool’s prediction of skilled task time and human participant’s performance for each task (system failures). For this purpose, the three system failures (Pump I, Pump II and Cascade failure) were modeled on CogTool for each of the two interface types to determine whether CogTool’s prediction falls within an expected range of values (95% CI from the calculated mean). Because each participant group had different degrees of interaction and familiarity with the software interface (DeltaV), data for each group were compared separately and independently with CogTool’s task time prediction. Table 4.4 below presents a summary of the results of CI tests.

Table 4. 4 Summary of results of confidence interval test showing CogTool’s prediction against the 95% CI

	Pump I (sec)		Pump II (sec)		Cascade (sec)	
	Improved	Poor	Improved	Poor	Improved	Poor
CogTool	8.4	10.3	64.8	78.6	30.5	67.1
Student	[61.4, 88.20]	[76.3, 115.3]	[100.3, 140.5]	[94.9, 157.2]	[31.4, 48.7]	[40.0, 92.7]
Within Interval?	No	No	No	No	No	Yes
Operator	[58.4, 91.1]	[42.0, 181.2]	[32.7, 175.9]	[84.9, 238.9]	[22.6, 37.4]	[94.7, 170.8]
Within Interval?	No	No	Yes	No	Yes	No

A quick review of the results shown in Table 4.3 showed that CogTool’s prediction fell within the 95% confidence interval one-fourth of the time. Two of the three accurate predictions belong to the operator participants who possess some level of familiarity with the interfaces, as well as had

a somewhat better understanding of what is expected of them in real world applications due to experience. Now, both of these two predictions in agreement with the operators' mean task time were on the “improved” interface and represents the task times for addressing the pump II and the cascade failure. It is worthy of mention that the cascade failure task time estimate for the student participants also fell within the 95% CI for the “poor” interface – these constitute the three CogTool's accurate task time prediction in this study.

When the task times for those few students (6 out of 24) who completed the total 5 steps out of 5 for the grey interface (mean=59.6, SD=33.0) were compared to CogTool's estimate (30.5sec), this estimate fell within the 95% CI ($51.30\text{sec} \leq \text{CogTool} \leq 67.98\text{sec}$). One would have expected their task time to be higher because of the increase in the number of steps, but the reverse was the case. This may be due to the fact that these students had a good sense of the task and what is required of them, as a result, they had no need for long meditations or thinking, drastically reducing their overall task time. In the same fashion, CogTool's KLM model is for an expert operator whose information retrieval is in the working memory, to address tasks that are pre-meditated, thus average think times have been established as 1.2sec. However, if a participant was able to eventually get to any one of the failures flooding the system, having lost considerable time, it is obvious that such task time recorded will be much higher than the average of other participants who were able to address that particular event promptly. Adding these extreme observations (task times) in calculating the means and standard deviations would result in artificially high values that are incongruent to the actual experiment. Thus, as mentioned earlier, values that were equal to or exceeded 3 standard deviations were excluded from the data analysis in order to maintain uniformity.

In retrospect, there were twelve null hypotheses that this study set out to investigate. In view of the results described above, we failed to reject three. This implies that the CogTool software did not precisely predict real world situation or conditions of operation for the two interface designs under investigation, as witnessed in the scenarios played by both human participant groups. Further discussions on this are provided in the next chapter. Table 4.5 below presents a summary of the hypotheses tested.

Table 4. 5 Summary of Null Hypotheses

	Description	Observation	Conclusion
Hypothesis 1	No difference exist between CogTool's skilled time estimate and student task time for PI system failure on "Improved" interface	Rejected	Difference Exists
Hypothesis 2	No difference exist between CogTool's skilled time estimate and operator task time for PI system failure on "Improved" interface	Rejected	Difference Exists
Hypothesis 3	No difference exist between CogTool's skilled time estimate and student task time for PII system failure on "Improved" interface	Rejected	Difference Exists
Hypothesis 4	No difference exist between CogTool's skilled time estimate and operator task time for PII system failure on "Improved" interface	Failed to reject	CogTool's prediction within 95% CI
Hypothesis 5	No difference exist between CogTool's skilled time estimate and student task time for Cascade system failure on "Improved" interface	Rejected	Difference Exists

(Table 4.5 continued)

	Description	Observation	Conclusion
Hypothesis 6	No difference exist between CogTool's skilled time estimate and operator task time for Cascade system failure on "Improved" interface	Failed to reject	CogTool's prediction within 95% CI
Hypothesis 7	No difference exist between CogTool's skilled time estimate and student task time for PI system failure on "Poor" interface	Rejected	Difference Exists
Hypothesis 8	No difference exist between CogTool's skilled time estimate and operator task time for PI system failure on "Poor" interface	Rejected	Difference Exists
Hypothesis 9	No difference exist between CogTool's skilled time estimate and student task time for PII system failure on "Poor" interface	Rejected	Difference Exists
Hypothesis 10	No difference exist between CogTool's skilled time estimate and operator task time for PII system failure on "Poor" interface	Rejected	Difference Exists
Hypothesis 11	No difference exist between CogTool's skilled time estimate and student task time for Cascade system failure on "Poor" interface	Failed to reject	CogTool's prediction within 95% CI
Hypothesis 12	No difference exist between CogTool's skilled time estimate and operator task time for Cascade system failure on "Poor" interface	Rejected	Difference Exists

5. DISCUSSION AND CONCLUSION

This project sought to evaluate the accuracy of task time predicted by CogTool – a human performance modeling tool – that generates a numeric estimate for any simulated task on different interface designs. The tasks considered were essentially those carried out by control room operators in oil and gas pipeline industries, on two versions of DeltaV pipeline interface - one of the leading petrochemical software programs in the industry. CogTool's time predictions were compared to results from a previous experiment that used human participants (one group of college students in training to become control room operators, and the other, experienced control room operators), and afterwards determined if those predictions fell within the 95% confidence interval of the means from the previous experiment.

This investigation illustrated that when tasks were performed on the improved interface, time to address events were shorter. On this interface, the operator is usually notified of an out-of-mode component by placing a dark blue box over the component. This cue may significantly reduce cognitive workload and increase available working memory, allowing participants to focus on other aspects of the current task. In addition, the contrast provided by the blue colored box on a grey background eliminated the search time it would have taken the user to locate the same component on the black interface. Also, clicking inside the box automatically opens the corresponding faceplate, allowing for easy access to acknowledging the alarm and performing other actions as required via the faceplate interface. This was observed in both groups of participants' results from the previous experiment and in CogTool's estimate.

Furthermore, the results revealed that for the two pump failure types on both interfaces (pump I and pump II), neither participant groups exhibited the skilled level shown by CogTool's prediction. The estimated time generated was less than the means recorded for the students and operators,

sometimes as high as 9 times the predicted CogTool estimate (e.g. pump I, improved: 8.4sec vs. 74.8sec). This is probably because the “think time”, estimated as 1.2sec, is less than the time participants require to address these tasks. Also, CogTool only accounts for the actions demonstrated when tasks are modeled on it, such as mouse movements, clicks etc., and does not include time when an operator stops midway to attend to a distraction. Any other activities relevant to the task being modelled that consumes time would have to be input manually. For instance, if the system delays before switching screens, this would be entered as “System Wait Time” and its duration specified in seconds, for CogTool to add to its prediction.

Alternatively, in order to assess the participants’ navigation actions against CogTool’s numeric estimate, and disregarding higher order cognitive processing, 3 participants were drawn randomly from the grey interface and their screen activities as captured on the Morae manager were analyzed. Specifically, the average number of mouse clicks these participants executed was counted, and compared to the number of clicks CogTool estimated a skilled user would require. This was done for each of the 3 task types. On the average, the participants demonstrated more mouse clicks than CogTool estimated (as seen in Table 5.1), signifying a level of task execution lower than that of a skilled operator. As well, the participants’ number of mouse clicks was multiplied with the standard KLM time for such action (left-mouse-button using the right-hand, estimated as 2.09sec on CogTool) to determine if overall, the product would be closer to the CogTool’s total estimate. The corresponding result revealed that for both the pump II and cascade failures, the product of the average number of clicks and CogTool’s 2.09sec estimate for such action is close to the overall skilled time estimated by CogTool. However, this is not so for the pump I event, as the resulting product (mean click * 2.09) is about 4.3 times higher than anticipated. This is not surprising in that, of the three mean task times recorded for both participant groups on the 2 interfaces, pump I

mean task time is the farthest from CogTool's estimate (see Tables 4.1 and 4.2). Also, the same trend as observed when comparing the overall task times was noticed; mouse click/navigation time increased from PI to PII events and then dropped with the Cascade failure.

Nonetheless, the cascade failure recorded operator mean time for the improved interface and student participant's mean time for the poor interface, to be approximately equal to that of CogTool's estimate. For the cascade system failure, it was possible to skip opening the PHV screen and/or wait for the PV to be within range. 18 of 24 students did this, thus eliminating a couple of steps from the procedure. The reason for this is not clear and could possibly be negligence, distraction or confusion on how to proceed in the task. In order to maintain homogeneity of task, this was the case adopted for CogTool's simulation (eliminated the PHV step). If a component however, goes out of mode without the participant detecting and addressing it, other alarms may be triggered resulting in a flood of failed components, and the participant became overwhelmed and task time is unusually high (reaching as high as 1191sec), in cases when the failure was eventually addressed. Although, the student participants were trained and allowed to practice using the interfaces before they completed the experiment, it may be unfair to consider them "skilled" in the task of addressing the system failures. This judgment arose from reviewing their steps and time to complete events, of which their mean task times was consistently higher than CogTool's prediction - five times out of six. However, when the task times for those few students (6 out of 24) who completed the total 5 steps out of 5 in the cascade failure for the grey interface (mean=59.6, SD=33.0) were compared to CogTool's estimate (30.5sec), this estimate fell within the 95% CI ($51.30\text{sec} \leq \text{CogTool} \leq 67.98\text{sec}$).

Table 5. 1 Comparing participant's navigation actions against CogTool's overall estimate

Grey Interface (Medium Scenario)	# of Clicks				Average time taken/event (sec)	CogTool's estimate (sec)
Cascade failure	Participant #3	Participant #5	Participant #10	CogTool		
Event 1	13	3	12			
Event 4	7	2	10	3	2.09*8=16.72	30.5
Event 6	3	4	18			
Average # of clicks		8 (5.57)				
Pump II Failure						
Event 3	13	78	45	14		
Event 8	3	7	10		2.09*26=54.34	64.8
Average # of clicks		26 (29.6)				
Pump I failure						
Event 5	18	15	-	4		
					2.09*17=35.53	8.4
Average # of clicks		17 (2.12)				

*CogTool's estimated time for mouse clicks (1.2+0.05+0.05+0.59+0.05+0.15)=2.09sec corresponding to think (cognition), look-at widget (cognition), move mouse (cognition), move cursor (right-hand), click mouse (cognition), click mouse (right-hand). Words in parenthesis refer to the resource type corresponding to that specific action.

This implies that some students actually had a good grasp of the simulation and knew what was expected of them in addressing the system failure as it appeared on the screen. They had no need for prolonged meditation or thinking and therefore moved systematically from one step to the other, resulting in a shorter task time. This was how CogTool's actions (based on KLM) were programmed: very simple decisions about where to point rather than trying to retrieve information from memory or some form of complex cognitive brainstorming (Sears & Jacko, 2007). Thus, probably if given further training and more time to practice, the number of such students may increase. The operators supposedly took extra care on the poor interface, perhaps due to their extensive training and experience to project consequences of hasty actions, thus spent longer time in addressing the cascade failure compared to CogTool's estimate.

In addition, it takes a user some search time or "lucky guesses" to locate an out-of-mode component on the poor interface before an alarm goes off, mostly as a result of the poor contrast between letterings and the interface background. Total task time becomes a function of how fortunate the user is in locating the out-of-mode component early rather than how skilled he/she is in responding to the switch in mode. This is mostly a shortcoming regarding the features and layout of the poor interface and may have influenced the poor performance of the participants, in terms of longer search times (which culminates into longer overall task time) and omission of some steps within the procedure.

These findings relate back to strategies for dealing with uncertain information where the participants had to monitor the system while searching for out-of-mode components, to restart or switch them to their appropriate status. It relates also to a state where no information/cue is provided, as on the black interface, therefore placing a greater demand on the participants to make decisions based on their own knowledge or assumptions of the screen display. The outcomes of

this study support the argument by (Abdeen & Shata, 2012), that interfaces should be designed to help reduce the effort required to understand them while still maintaining their overall functionality. The improved interface supports this requirement as it provides better color contrast and improved font sizes making it easier to read information off the screen. Another feature of this interface is that it provides feedback on the status of the components; for instance a pump in running condition is displayed as green but shows as red when stopped. As well, clicking on an alarm takes user to the faceplate and correct screen unlike the poor interface that only displays the correct screen. Also, according to (Allender, 2000), human performance is expected to vary not only as a function of the interface design itself (e.g., the amount of information displayed, the color variation, presence/absence of cues), but also as a function of other unobservable, internal states such as aptitude, cognitive workload, and motivation. The student's poor performance (5 times out of 6), could probably be attributed to less aptitude/motivation to project/relate their actions to real-life situations, unlike the operators who may have embraced the experiment like they would on their jobs, having possessed more knowledge of the potential implications of their actions. Besides, students often have considerable academic workload (assignments, quizzes, exams), in addition to other curricular activities, being invited to take part in experiments that would place an extra demand on their cognitive abilities, may not be given 100% attention. The experiment (by its nature) demands a substantial amount of users' mental processing resources (the students being less experienced and also lacking the many years' worth of training that the operators had), and this demand may have impacted their performance. In (John, 2011) experiment to test the similarity of models generated by 100 students on CogTool, the experiment was used towards the students' final grade.

Moreover, the interfaces employed in this study required users to switch between several screens that are lacking in hierarchical structure and whose navigation methods are often not logical or consistent. The displays are 2D versions of a 3D system, with parts of a component in spatial isolation; not necessarily depicting how a novice would picture a petrochemical refining facility. This differs substantially from one of the previous studies conducted by CogTool's creator to validate the software by using the screen display of a Palm PDA, which has a simple interface and structured menus (Luo & John, 2005). The study investigated the time 10 participants who were classified as expert PDA users would take to find the opening hours of the Metropolitan Museum of Art (MET) on the PDA. The participants had the liberty to choose any preferred method of the four allowed, and their task execution times were then compared to CogTool's model. Participants for this study on the other hand, had to follow a set procedure, often times did not know what to expect and were assigned a monitoring task unlike a defined and goal-based type of task. The (Luo & John, 2005) study recorded an average prediction error of 3.7%, while for those 3 predictions within the 95% CI, average prediction error was 78.3%.

Furthermore, literature on CogTool involved studies where the tasks simulated were goal-specific, for instance, (John, 2011) which investigated the similarities between CogTool models of 2 websites, made by 100 college students. The websites are for cataloging books and sharing collections and the tasks involved were to: (a) sign-in and add a book to your collection, (b) tag the book you just added, and (3) rate that book and sign-out; very specific tasks requiring low cognitive demand. In comparison, the tasks demanded of the participants in this study are of a monitoring nature, and are cognitively demanding. Operators work on multiple screens and/or consoles and often times are required to commit to memory that they have to revisit a particular display to check on some critical point. This is even more heightened when an abnormal situation

presents itself, unlike tasks modeled in the past which have been relatively simple and required less mental stimulation. The aforementioned drives home the point that in such abnormal situations, it would be erroneous to assume a constant mental operator (or think times) as proposed in the KLM theory on which CogTool was created.

Additionally, the change in screen display and associated colors was not automatically recognized by CogTool. This is because the KLM model (in CogTool) assumes that the five operators: Ks, Ps, Hs, Ds, and one user mental operator M, take constant time for each occurrence, thus would not change when the display and its features changed. As discussed earlier, search times or wait times were input manually during the modeling to take them into account; otherwise, CogTool's estimate would only be artificial and not represent the actual DeltaV simulation. The wait times are average times as seen taken by a random number of participants for the PV to come within 5% of the original steady state value while the search time (mostly on the poor interface) was modeled after the experimenter's actions.

Other than the aforementioned, CogTool provides a simple yet impressive front-end to the powerful and well-validated cognitive modeling engine ACT-R. This front-end is easy to learn and use and can easily be used to train novice modelers. Rather than simply generating a single numeric estimate of task time, CogTool researchers may also consider providing such results as (average) number of clicks to success (meaning to complete task) and/or a distribution of task times for users succeeding without error and those with varying degrees of error. This would assist experimenters in deciding where a typical participant falls on the distribution. As well, CogTool does not seem to account for any other thinking besides that which is required for navigations on the interface, thus decision making or higher order processing times need to be integrated into the models generated on CogTool.

The other objectives of this study – identifying bottlenecks in the DeltaV user interface by investigating CogTool’s timeline visualization were also considered. Although CogTool’s predictions were mostly outside the 95% confidence interval of expectations, comparing the visualization models of each task per interface, revealed some drawbacks on the poor interface. This interface compelled participants to spend more time searching for information and required extra step(s) to take users to the desired screen when an alarm/component is clicked. It was interesting to note that the drawbacks identified were similar to those listed in the differences between both interfaces, thereby reinforcing the earlier conception about the poor interface. The same findings were also seen on the videos of the participants’ sessions recorded with the Morae software.

Thus, albeit estimating task times that are mostly different from the actual participants’ data, the timeline visualization feature of the CogTool software provides a very viable tool for quickly assessing the two interface designs. At a glance, it showed all the actions a user took in addressing each of the tasks on each frame, making it relatively easy to foresee and eliminate potential drawbacks in an interface design that may likely affect skilled user performance. Therefore, companies or organizations looking to purchase new (interface) products or complex systems or to design improved interfaces to aid their processes, would find CogTool useful in making optimal choices amongst an array of selections and then aid them in choosing the most suitable from amongst those choices.

5.1 Potential Limitations of Study

Between vs. Within Subjects: While the previous study (Koffskey, Ikuma, & Harvey, 2013) was an experiment between subjects, with each participant going through only one session on an interface type, CogTool’s simulation can be likened to a “within” subject test i.e. for a typical

skilled operator, the think time is constant regardless of the interface type or order of scenario. However, an operator who was tested only on the improved interface may be different from the other who tested on the poor interface in terms of how well each understood the modalities/features of the interface and the tasks presented on the screen. A participant may perform better (or worse as the case may be), if he had experienced one interface first, and then the other. In contrast, CogTool is just one ‘user’, who saw both interface scenarios.

Nature and size of sample: The majority of participants used in the previous experiment were students while the study’s objective was to determine metrics for operator performance. In spite of this, the number of operators was too small at only $n=4$. Considering that CogTool’s prediction was slightly more successful for the operators, and in order to increase the external validity of this study, a larger operator sample size may be considered. It is expected that a larger n should eliminate the possibility/effect of chance that may be present in the four operators selected, because the average task time recorded with a larger size would be approaching that of the population for control room operators and would provide a better perception of how much time they would spend addressing similar tasks.

Elimination of some Observations: A general overview of the data collected from the previous study (Koffskey, Ikuma, & Harvey, 2013) showed that some observations were unusually high. For instance, participant #1 took 1175s to address a Pump II system failure on the grey interface. Such observations add up to create an increase in the mean task time calculated for the corresponding system failures. As a result, they were identified as outliers and subsequently removed from those used in the current study ($\geq 3\sigma$ away from the mean), thus reducing the amount of data available for assessment and analysis.

Lastly, the black interface is essentially an interface with high-contrast, featuring bright figures on black background. Whilst it provides good color contrast for text, the black interface also produces too much glare and does not provide a good background for a range of colors (Hexatec Consulting, 2010). A display of this kind is uncomfortable to the eye, making the participants spend longer time staring at a particular screen or trying to figure out the connections between the components or their modes. This may explain why CogTool's prediction fell short of expectations as it was unable to identify the change in interface color and the effect of such change on users.

5.2 Conclusion and Future Research

In order to evaluate how accurate task time prediction on the CogTool software is, this research compared data from a study that investigated the time duration two groups of participants spent working on certain tasks on the interface design of a petrochemical software – DeltaV. Twelve hypotheses were developed and data from the two participant groups were organized into 95% CI of the means in order to check if CogTool's prediction fell within the intervals. CogTool's prediction fell short of expectations as the time estimates generated were mostly lower than those recorded for the human participants. This may be as a result of the one user mental operator M (in the KLM model that CogTool is based on) being less than what actual control room operators working on petrochemical interfaces require. The mental operator used in CogTool may need to be revised for tasks whose nature may imply grave consequences and thus require extra thought before execution, such as a control room monitoring task. However, given the nature of the base data that was used as a comparison bed for evaluating the accuracy of prediction, as well as other factors at play in the previous experiment, such as the sample size, missing observations (and outliers), participant training/experience, it may be reasonable to infer the present CogTool's performance as tolerable only for the operators' performance on the improved interface.

Further, the 3 system failures under investigation (PI, PII and C) were modeled in isolation that is assuming the participants were tested on only one trial for each interface; this was not the case. The effect of several rounds of addressing the failures under different task difficulty levels (3 PI, 11 PII and 9 C each), could have impacted on the participants' performance, unlike if they had only experienced 3 tasks (1 PI, 1PII and 1C).

Finally, the uncertainty of not knowing what to expect or future events on a live situation may introduce boredom, especially after long hours of inactivity, or cause an operator to become conservative, which would in turn result in reduced performance. Lower performance in this case may produce a longer mental operator M, which the CogTool model would fail to recognize, being a 'dummy' version of the simulation.

Recommendations for further research could be:

- Investigate the one user mental operator M, or "think time" in the KLM model for CogTool, to determine the appropriate value for a control room task.
- Consider a live interface simulation of models on CogTool, rather than the present mock-up version of interface designs. It is expected that this should enable CogTool to automatically detect and integrate in its estimate such actions as system wait times, search time (associated with poor interface background/design) etc.
- Employ a larger sample size for the participants; hopefully this will eliminate the large variation in the data used for this study.
- Participants preferably should be actual control room operators and not college students, in order to increase the external validity of the research.

- If students would be used because of availability, training should be more rigorous and/or some sort of motivation be provided (in the form of incentives) to gain their genuine interest. This might reduce the amount of missing data and outliers as seen in the previous data.
- Rather than simply recording a video of the screens, the Morae software should also be used to record time between clicks as this would assist in quickly determining how much time went into the search or wait sessions.
- Instead of conducting future experiment as between subject, it may be more acceptable to investigate the outcomes of a within subject test in line with the CogTool model.
- Consider integrating the effect of background color in CogTool's simulation to determine whether its numeric prediction of skilled task time would differ.

REFERENCES

- Abdeen, H., and Shata, O. (2012). Metrics For Assessing The Design Of Software Interfaces. *International Journal Of Advanced Research In Computer And Communication Engineering*, 1(10), 737-745.
- Alion Science and Technology. (2010). <http://www.alionscience.com/Top-Menu-Items/News-Room/Publications-White-Papers>
- Allender, L. (2000). *Modeling Human Performance: Impacting System Design, Performance, And Cost*. Paper Presented At The Proceedings Of The Military, Government And Aerospace Simulation Symposium, Advanced Simulation Technologies Conference, Washington, D.C.
- Anderson, J. R., Bothell, D., Byrne, M. D., Douglas, S., Lebiere, C., et al. (2004). An Integrated Theory of the Mind. *Psychological Review*, 111(4), 1036-1060.
- Avouris, N. M. (2001). Abstractions For Operator Support In Energy Management Systems. *Electrical Power And Energy Systems*, 23, 333-341.
- Bellamy, R., John, B., & Kogan, S. (2011). *Deploying Cogtool: Integrating Quantitative Usability Assessment Into Real-World Software Development*. Paper Presented At The International Conference On Software Engineering (Icse).
- Bias, R. G., & Mayhew, D. J. (1994). *Cost-Justifying Usability*. Boston: Academic Press.
- Byrd, W. R. (2010). New Control Room Management Regulations Require Structured Management Approach. *Pipeline & Gas Journal*, 237(2), 1-5.
- Card, S. K., Moran, T. P., & Newell, A. (1986). *The Model Human Processor: An Engineering Model Of Human Performance*. In K. R. Boff, L. Kaufman, & J. P. Thomas (Eds.), *Handbook Of Perception And Human Performance*. Vol. 2: Cognitive Processes And Performance.
- Dot, Department Of Transportation -. (2010). Pipeline Safety: Control Room Management/Human Factors. <http://www.reginfo.gov/public/do/eagendaviewrule?pubid=201010&rin=2137-ae64>
- Duffy, V. G. (2009). Research For Applied Ergonomics And Human Factors Engineering. *Handbook Of Digital Human Modeling*, CRC Press: Taylor & Francis Group, Boca Raton, FL.
- EIA, U.S. Energy Information Administration: Annual Energy Review (2011). <http://www.eia.doe.gov/totalenergy/data/annual/index.cfm>
- Environmental Protection Agency, EPA (1995). Profile of the Petroleum Refining Industry, *EPA Office of Compliance Sector Notebook Project*, SIC 2911.

- Errington, J. D. (2005). *Establishing Human Performance Improvements And Economic Benefit For A Human-Centered Operator Interface: An Industrial Evaluation*. Paper Presented at the Human Factors And Ergonomics Society 49th Annual Meeting, Santa Monica, CA.
- Ferre, X. (2003). *Integration Of Usability Techniques Into The Software Development Process*. Presented At Workshop On Bridging The Gaps Between Software Engineering And Human-Computer Interaction, Proceedings Of The 25th International Conference On Software Engineering (Portland, Oregon, May 03 -10, 2003). International Conference On Software Engineering. Ieee Computer Society, Washington, Dc.
- Formosa Plastics, C. (2007). *Investigation Report: Vinyl Chloride Monomer Explosion*. (Report No. 2004-10-I-II). Illinois: U.S. Chemical Safety And Hazard Investigation Board.
- Garzon, S. R., and Cebulla, M. (2010). *Model-Based Personalization Within An Adaptable Human-Machine Interface Environment That Is Capable Of Learning From User Interactions*. Paper presented at the third International Conferences on advances in computer-human interactions.
- Hexatec Consulting. (2010).
http://www.hexatec.co.uk/Consultancy/hmi_display_design_guidelines.aspx
- Huang, F., Lee, Y., Hwang, S., Yenn, T., Yu, Y., Hsu, C., et al. (2007). Experimental Evaluation Of Human-System Interaction On Alarm Design. *Nuclear Engineering And Design*, 237, 308-315.
- Jamieson, G. A., and Vicente, K. J. (2001). Ecological interface design for petrochemical applications: supporting operator adaptation, continuous learning, and distributed, collaborative work. *Computers and chemical engineering*, 25(7-8), 1055-1074.
- Jastrzemski, T. and Charness, N. (2007). The model human processor and the older adult: parameter estimation and validation within a mobile phone task. *Journal of experimental psychology: Applied*, 13(4), 224-248.
- John, B. E., Prevas, K., Salvucci, D. D., & Koedinger, K. (2004). *Predictive human performance modeling made easy*. Paper presented at the proceedings of the sigchi conference on human factors in computing systems, 6(1), 455-462.
- John, B. E. (2009). Cognitive crash dummies: user interface prototyping with a difference. Paper presented at the 31st International Conference on software engineering, Washington, DC.
- John, B. E. (2010). Reducing the variability between novice modelers: results of a tool for human performance modeling produced through human-centered design. *Proceedings of the 19th annual conference on behavior representation in modeling and simulation* (Brims) (Charleston, SC).
- John, B. E. (2011). Using predictive human performance models to inspire and support UI design recommendations. Proceedings of the CHI 2011 conference, session: predicting & modeling human behaviors, Vancouver, BC, Canada.

- John, B. E., Patton, E. W., Gray, W. D., Morrison, and D. F. (2012). Tools for Predicting the Duration and Variability of Skilled Performance without skilled Performers. Proceedings of the Human Factors and Ergonomics Society, 56th annual meeting.
- Kim, S. K., Suh, S. M., Jang, G. S., Hong, S. K., and Park, J. C. (2012). Empirical Research On An Ecological Interface Design For Improving Situation Awareness Of Operators In An Advanced Control Room. *Nuclear Engineering And Design*, 253, 226– 237.
- Koffsky, C., Ikuma, L. H., & Harvey, C. (2013). Performance metrics for evaluating petrochemical control room displays. Proceedings of the Human Factors and Ergonomics Society 57th Annual Meeting. September 30 – October 4, 2013, San Diego, CA.
- Li, Z. (2009). Digital Human Modeling Packages. In V. G. Duffy (Ed.), *Handbook Of Digital Human Modeling*. Boca Raton, FL: CRC PRESS.
- Luo, L. and John, B. E. (2005). Predicting Task Execution Time on Handheld Devices Using the Keystroke-Level Model. *Extended Abstracts Of CHI 2005* (Portland, Oregon, April 2-7, 2005) ACM, NY, 1605 – 1608.
- Meshkati, N. (2006). Safety and Human Factors Considerations in Control Rooms of Oil and Gas Pipeline Systems: Conceptual Issues and Practical Observations. *International Journal of Occupational Safety And Ergonomics*, 12(1), 79-93.
- Myers, B. A. (1998). A Brief History Of Human Computer Interaction Technology. *ACM Interactions*, 5(2), 44-54.
- NTSB (2005). Supervisory Control and Data Acquisition (Scada) in Liquid Pipelines. *Safety Study*, NTSB/SS-05/02, Washington, DC.
- O'Hara, J., Stubler, B., and Kramer, J. (1997). Human Factors Consideration In Control Room Modernization: Trends and Personnel Performance Issues. Paper Presented at the IEEE Sixth Annual Human Factors Meeting.
- Oyewole, S. A., Farde, A. M., Haight, J. M., and Okareh, O. T. (2011). Evaluation of Complex and Dynamic Safety Tasks in Human Learning Using the Act-R and Soar Skill Acquisition Theories. *Computers in Human Behavior*, 27, 1984–1995.
- Schunk, D.W., Bloechle, W.K., & Laughery, R. (2002). Micro Saint: Micro Saint Modeling and the Human Element. Paper Presented at the Paper Presented at the Proceedings of the 34th Conference on Winter Simulation: Exploring New Frontiers.
- Sears, A. and Jacko, J. A. (2007). The human computer interaction handbook: Fundamentals, evolving technologies and emerging applications. Laurence Erlbaum Associates, Taylor & Francis Group, 2nd Ed., 270 Madison Avenue, NY, pp 107-109.
- Shahriari, M. A., Shee, A., and Örtengren, O. (2006). The Development of Critical Criteria to Improve the Alarm System in the Process Industry. *Human Factors in Ergonomics & Manufacturing*, 16(3), 321 - 337.

- Thurman, D. A., & Mitchell, C. M. (1995). A Design Methodology for Operator Displays of Highly Automated Supervisory Control Systems. Paper Presented at the Proceedings of the 6th IFAC/IFIP/IFOR/SEA Symposium on Analysis, Design and Evaluation of Man Machine Systems, Boston, Mass.
- Trewin, S., Richards, J., Bellamy, R., John, B. E., Swart, C. and Sloan, D. (2011). Extending Predictive Models of Exploratory Behavior to Broader Populations. *HCI, Part I, HCII 2011, LNCS*, 6765, 149–158.
- Usability Professionals Association. (2010). <http://www.usabilitybok.org/>
- Wu, C. and Liu, Y. (2009). Development and Evaluation of an Ergonomic Software Package For Predicting Multiple-Task Human Performance and Mental Workload in Human–Machine Interface Design And Evaluation. *Computers & Industrial Engineering*, 56, 323–333.
- Zachary, W., Ryder, J., Stokes, J., Glenn, F., Mentec, J., & Santarelli, T. (2005). A Cognet/Igen Cognitive Model That Mimics Human Performance and Learning in a Simulated Work Environment. In K. A. Gluck & R. W. Pew (Eds.), *Modeling Human Behavior With Integrated Cognitive Architectures: Comparison, Evaluation and Validation* (Pp. 113-175). Mahwah, NJ US: Lawrence Erlbaum Associates Publishers.

APPENDIX

1. To start creating a storyboard, open the CogTool program on a PC, give the storyboard a unique design name and select the input devices; for this task we will only need the mouse and keyboard. Note that output device is automatically checked as display and greyed.

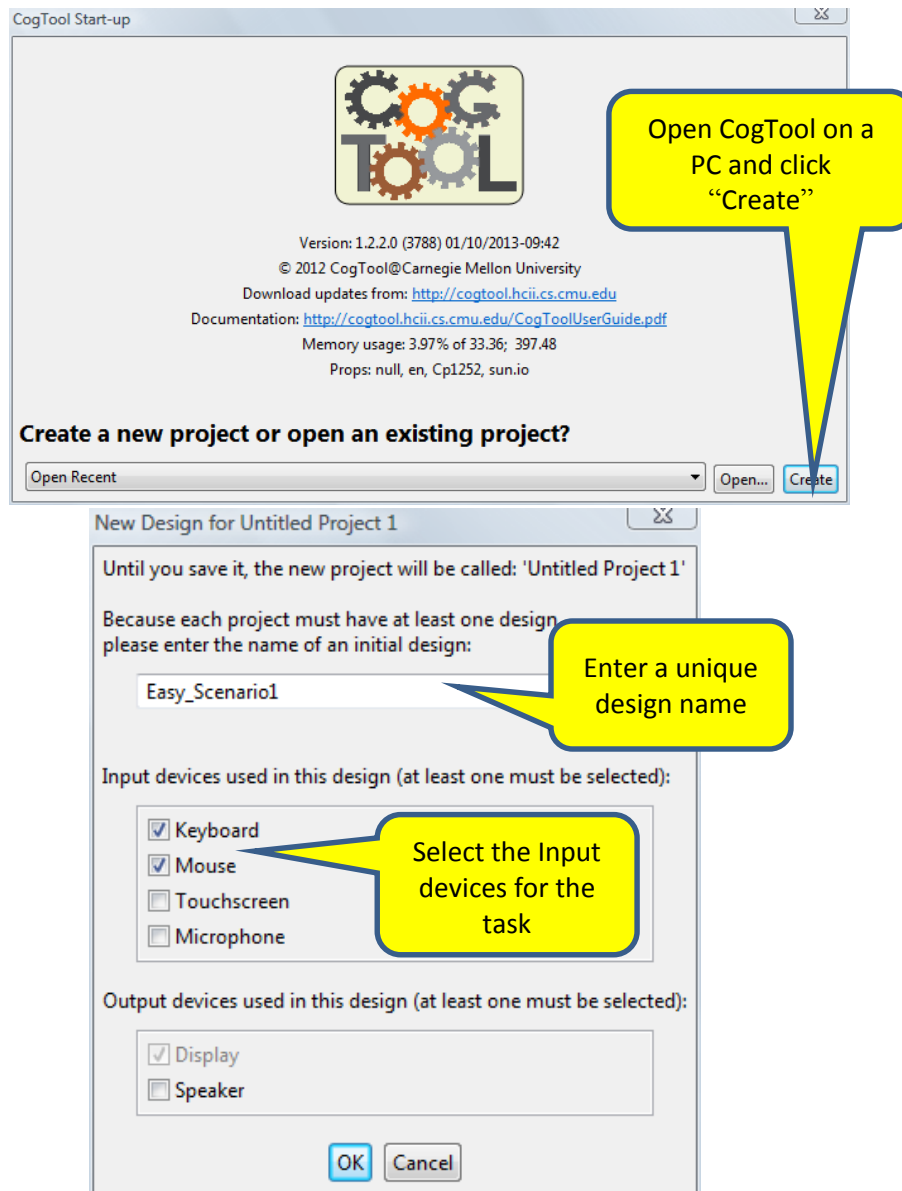


Fig. A. 1. Start CogTool and Specify Input Devices

2. CogTool recognizes the sequence of actions for the design as a task and automatically tags it as Task 1. This can be renamed as desired. Another window pops up, showing a frame; a frame

is used to represent what a user sees and is often blank allowing a modeler using CogTool to import background images.

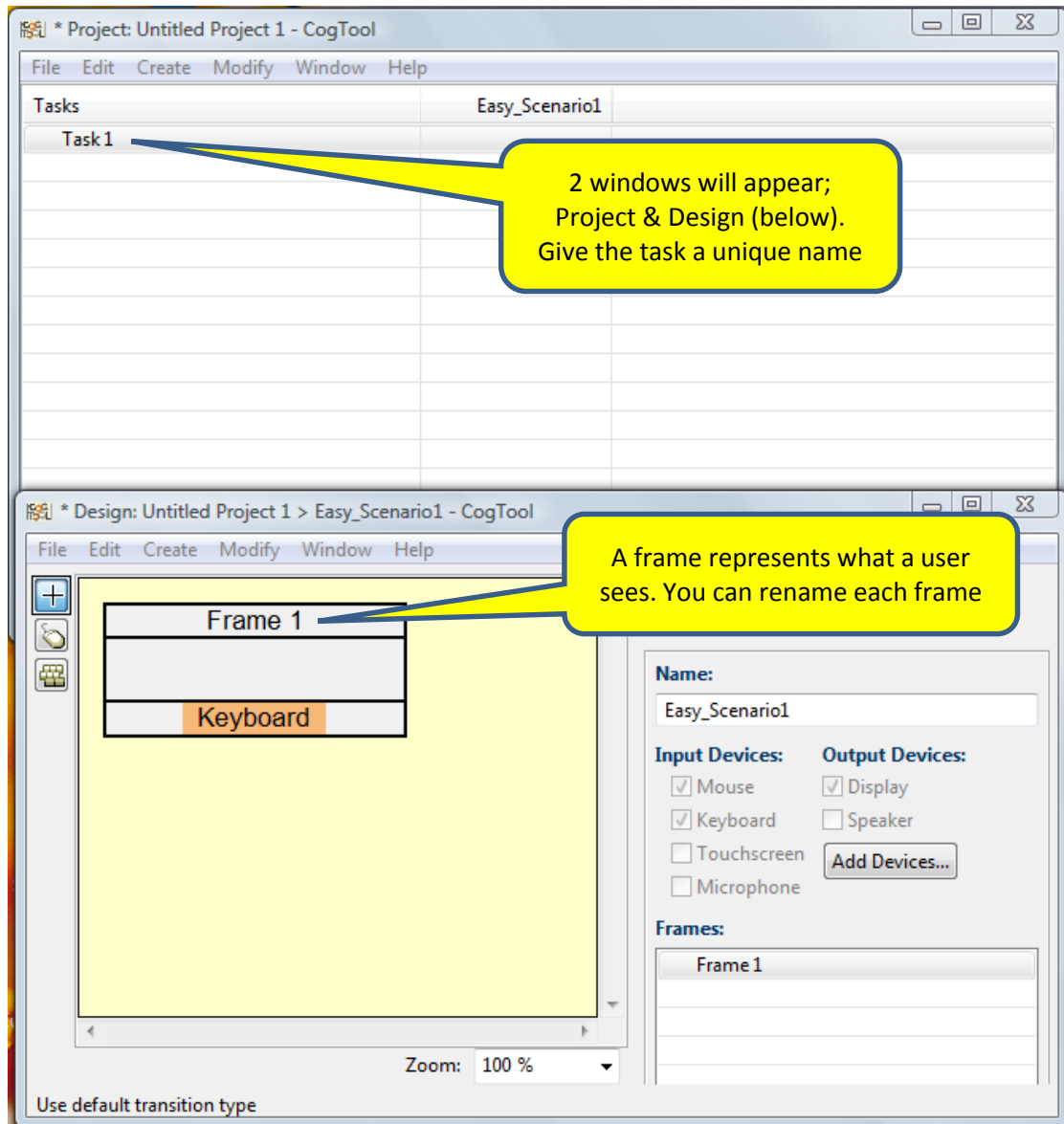


Fig. A. 2. Name the Task to be Modelled

3. At this point, the first screen that signals a pump failure, is imported as background image for the first frame.

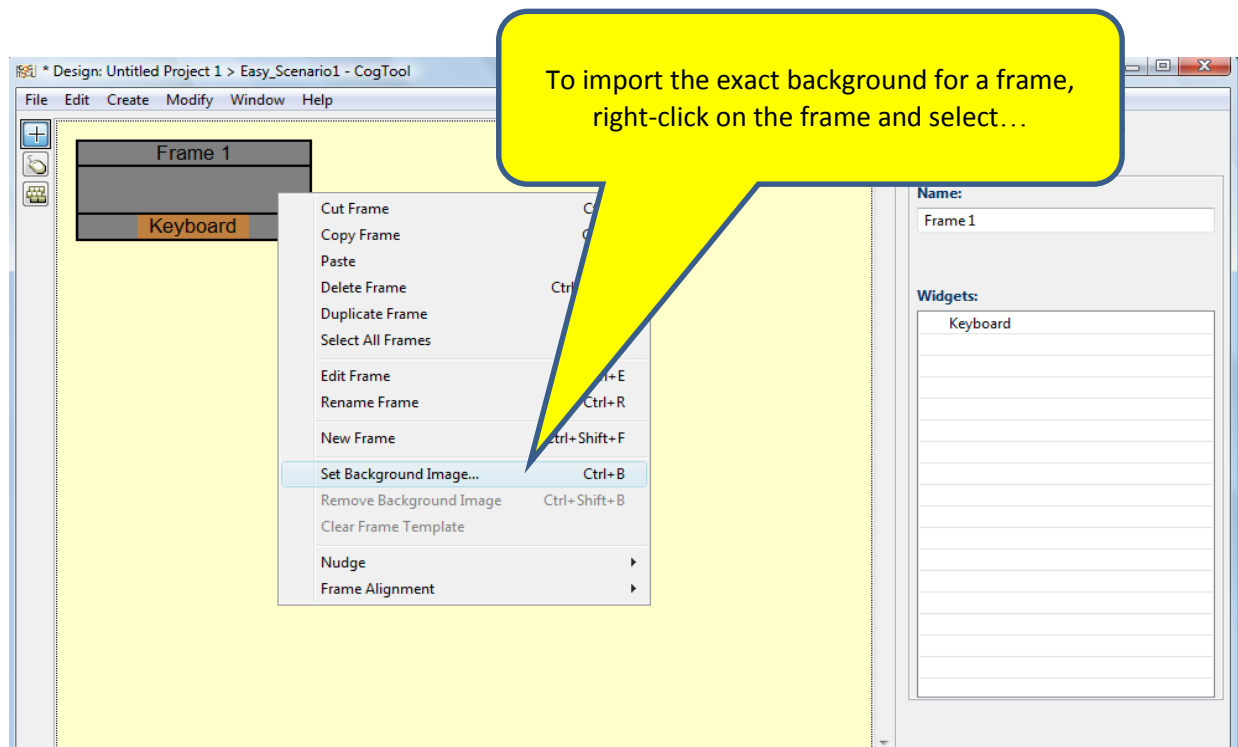


Fig. A. 3. Import Corresponding Background Images

4. For the first frame (with the heading “Frame 1” as shown below), the grey interface displaying the P-125 alarm banner blinking is imported.

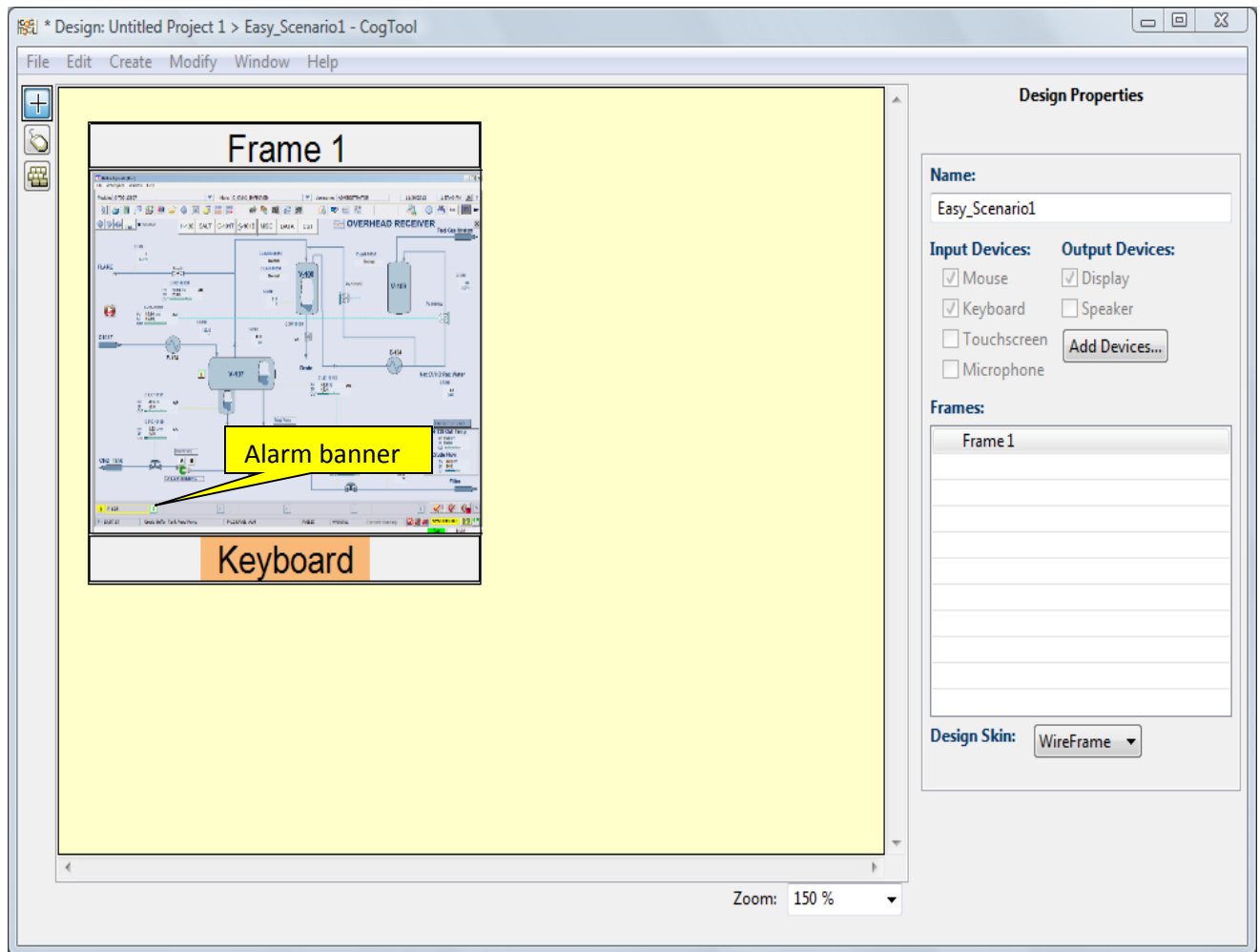


Fig. A. 4. Background Displaying the P-125 Alarm Banner Blinking is shown

5. Several frames can be added as required and the corresponding background interface image would also be imported. Click on “Create” and select “New Frame” (see image below). It is a good idea to either tag each frame sequentially in the order of actions to be performed or use specific names to represent the actions to be modeled.

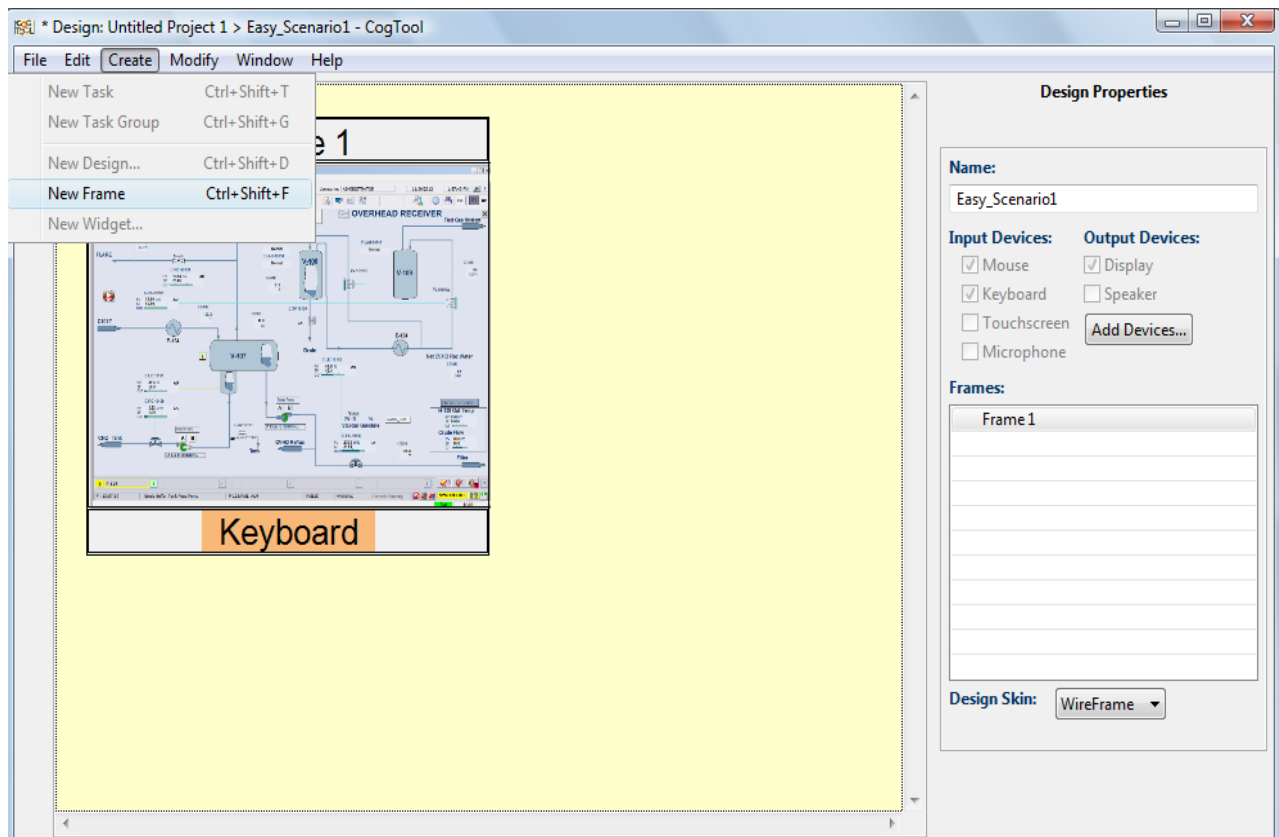


Fig. A. 5. Import Unique Background Images for each Frame

6. For this illustration, a total of eight frames have been imported, representing the interfaces the operator will see for the actions he need to execute in addressing this particular failure. There should be at least one action per frame.

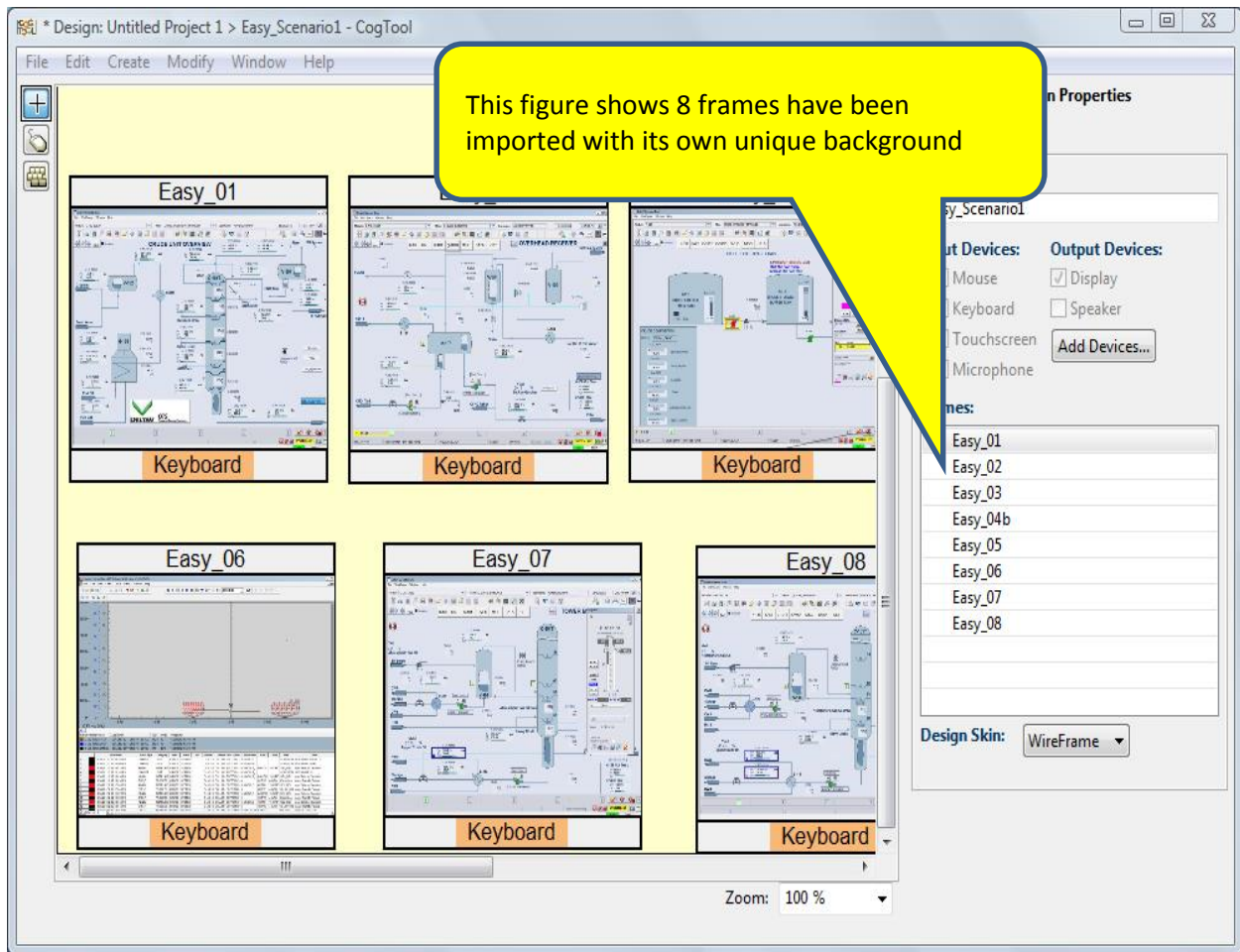


Fig. A. 6. All 8 frames needed for this demonstration are imported

7. To begin assigning widgets on each of the frames, double-click on a frame and select the most appropriate from the widgets tiled on the extreme left. Hovering the mouse over each widget

shows a description of what it does. In the next figure (Fig. A. 7.), P-125 alarm is triggered and the operator must respond by *clicking* on the alarm bar.

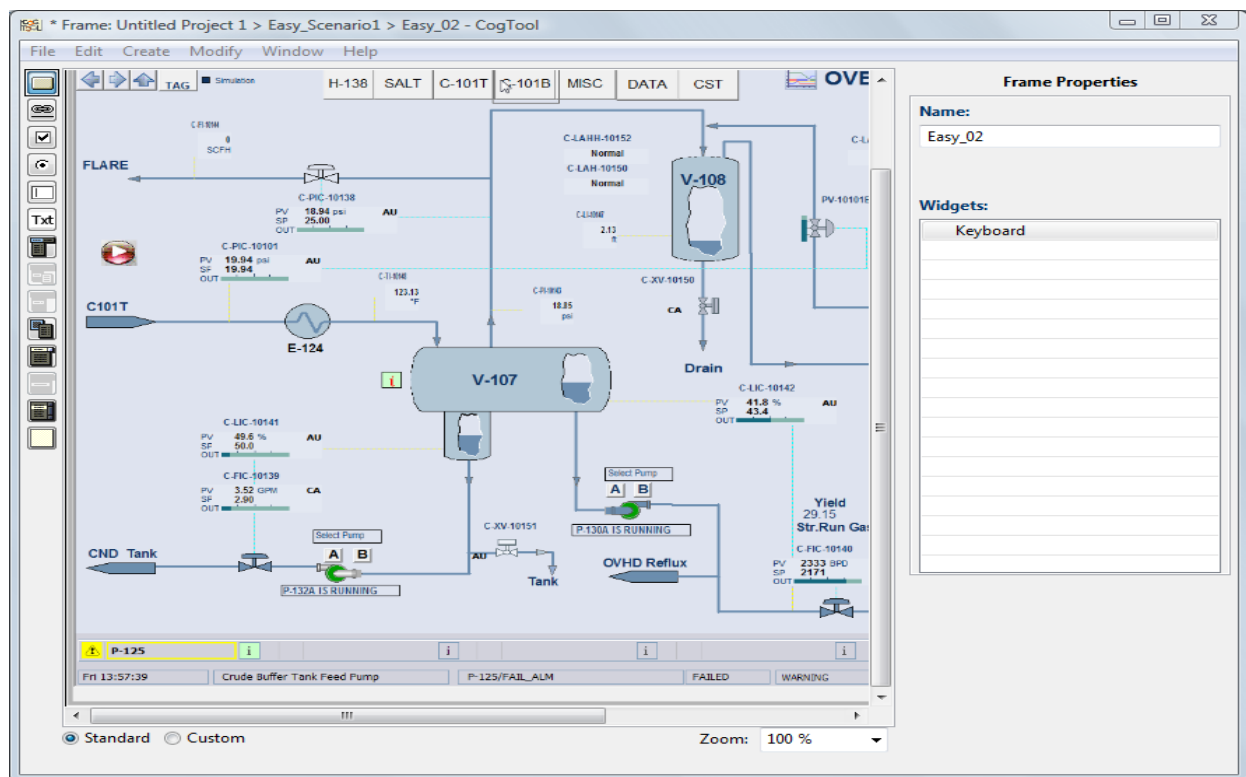


Fig. A. 7. An Alarm is Triggered - P-125

8. As mentioned earlier, when an alarm is triggered as a result of a failed component, an operator is required to first click on the alarm banner in order to go to the faceplate. CogTool has a widget for this kind of action – the “button” widget. It is fashioned after the action of left-click on the mouse. Window over the area to click on, and label the button appropriately.

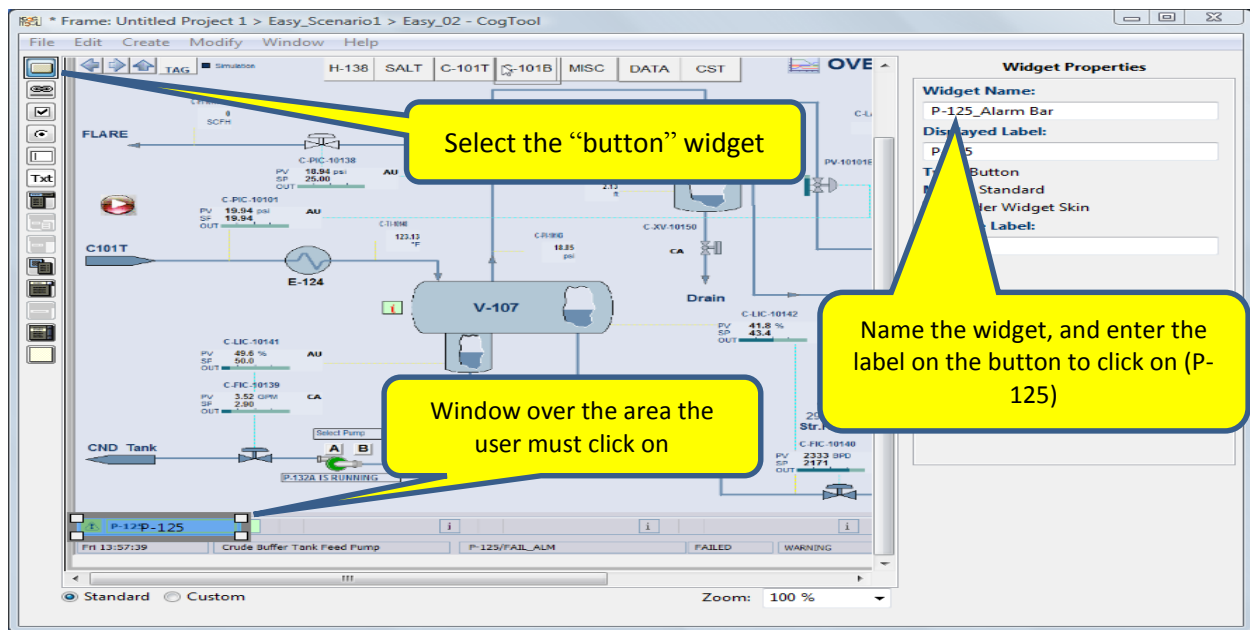


Fig. A. 8. Select the Appropriate Widget for the "Click" Action

9. When the operator clicks on the alarm bar, the faceplate is displayed. He is then required to re-start the pump. Another button widget is needed to demonstrate this action and would be created on the next frame (see Fig. A.9. below). This way, one can create widgets for all the actions on each of the frames.

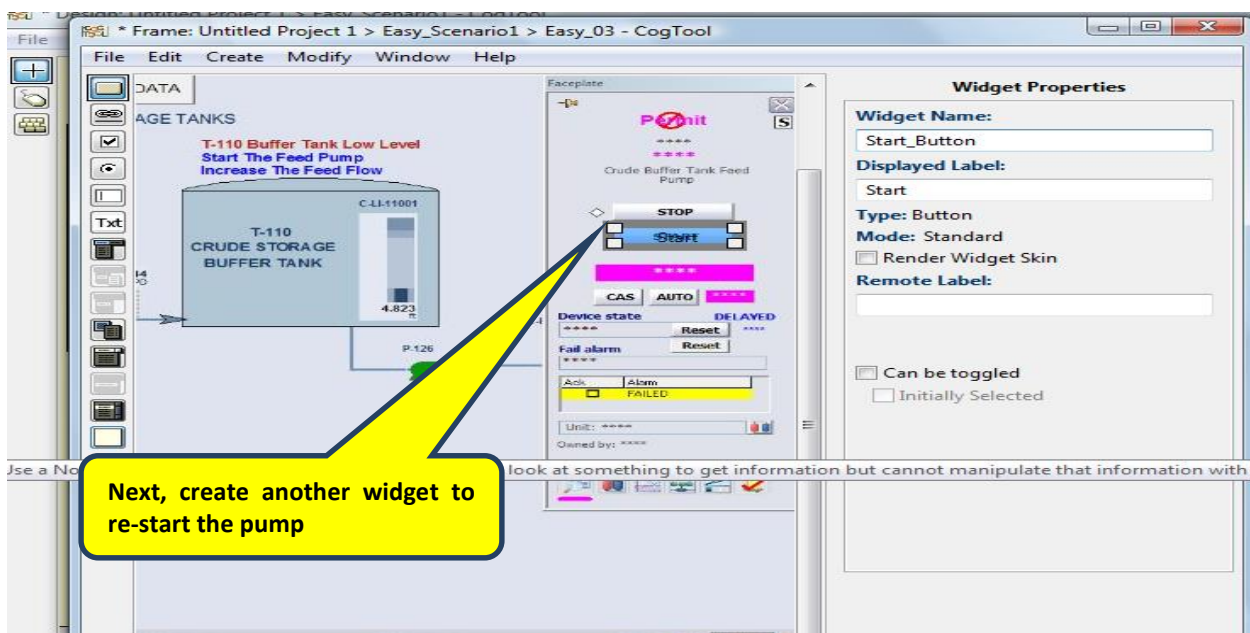


Fig. A. 9. Create another Widget to Represent "Starting the Pump"

10. Upon assigning the appropriate widget type to all actions, the next step is to create transitions between widgets (actions) and frames (background images). This simply means to show in sequential order, which actions precede which, and on which frame is the action executed. Transitions are a way of representing what a user does and in simple terms involves clicking on a widget and dragging the mouse to the next widget in the hierarchy of tasks.

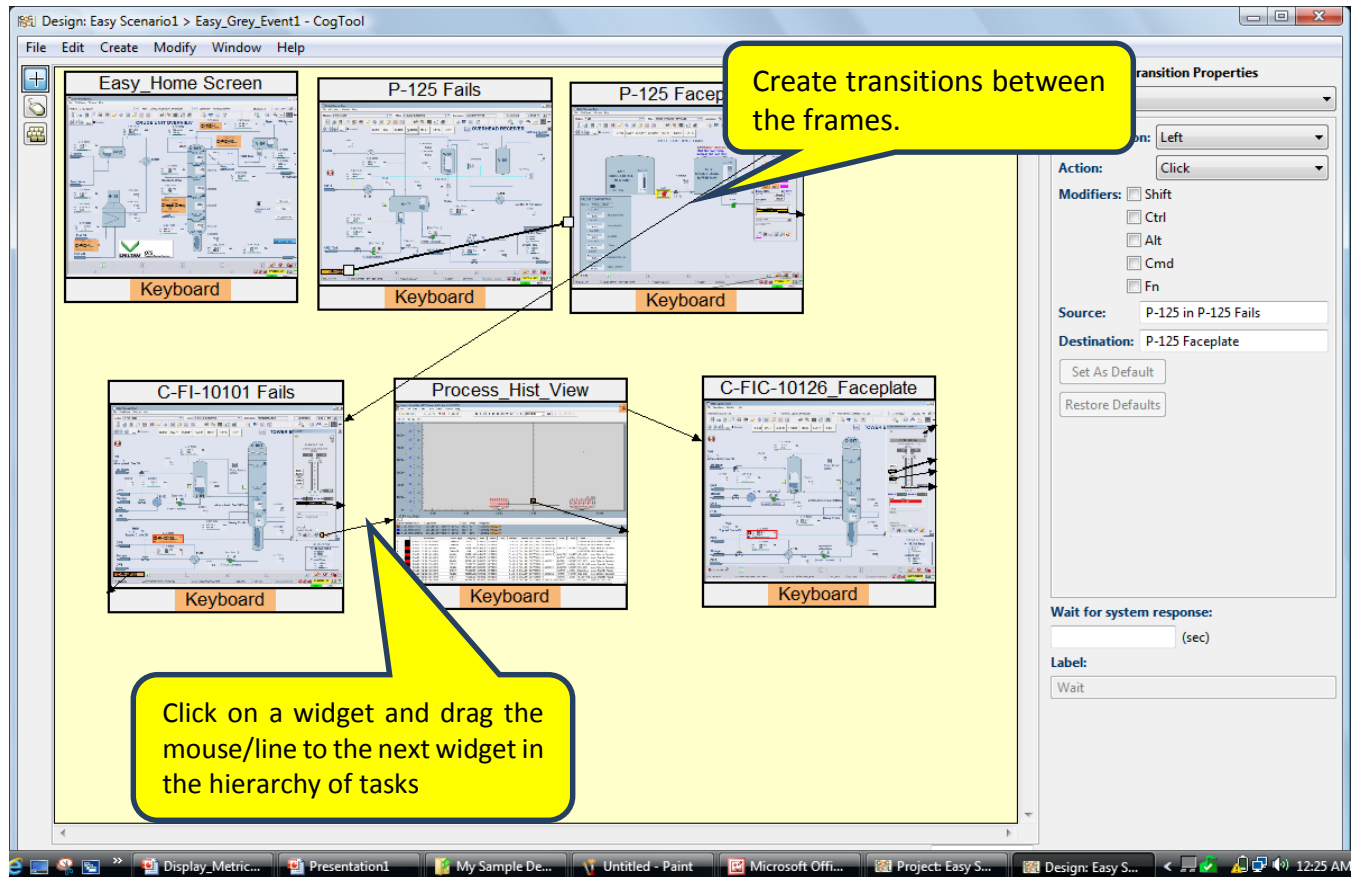
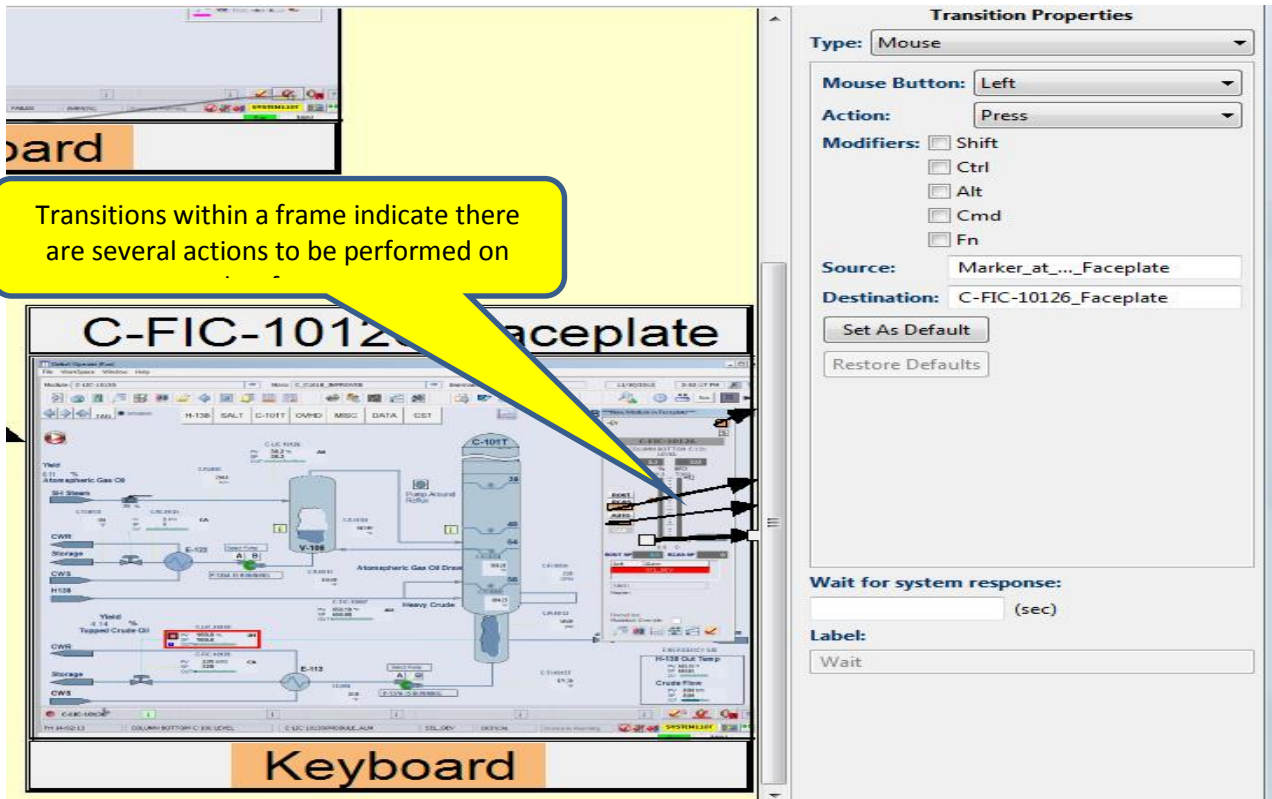


Fig. A. 10. Creating Transitions (Movement from one Widget to another)

11. Sometimes, it may be necessary to have several widgets on one frame, representing more than one action to be executed. CogTool denotes transitions within a frame with arrows/lines from the widget to the perimeter of the frame (as shown in the next illustration).



12. The screenshot below is a sample picture showing transitions completed and frames uniquely named. This is a storyboard, where each state of the UI is represented as a frame, each actionable interface item as a widget (e.g., link, button) or device (e.g., keyboard), and each action on a widget or device (e.g., mouse click, keys typed on the keyboard) as a transition tween frames.

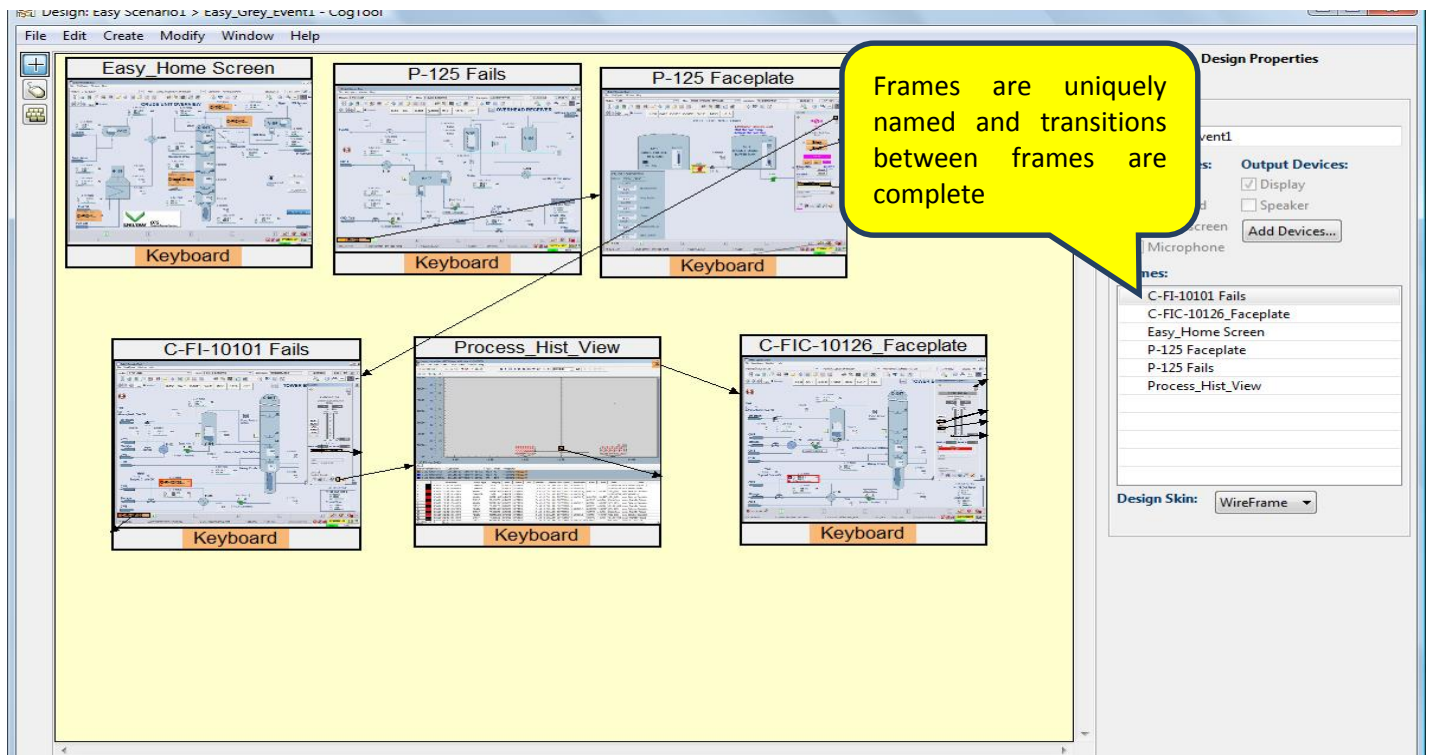


Fig. A. 13. Each Frame has been Assigned a Unique Name; Transitions are Complete

13. The next step is to demonstrate the tasks above on CogTool. Demonstration simply means walking through each action (beginning from the first frame) as a skilled user would do, and CogTool automatically builds the entire KLM for the task being simulated, by tracking and recording the movements or transitions between actions (mouse clicks and keyboard entries) and injects think times and wait times where necessary.

14. During the demonstration of actions, CogTool writes the KLM script as shown below:

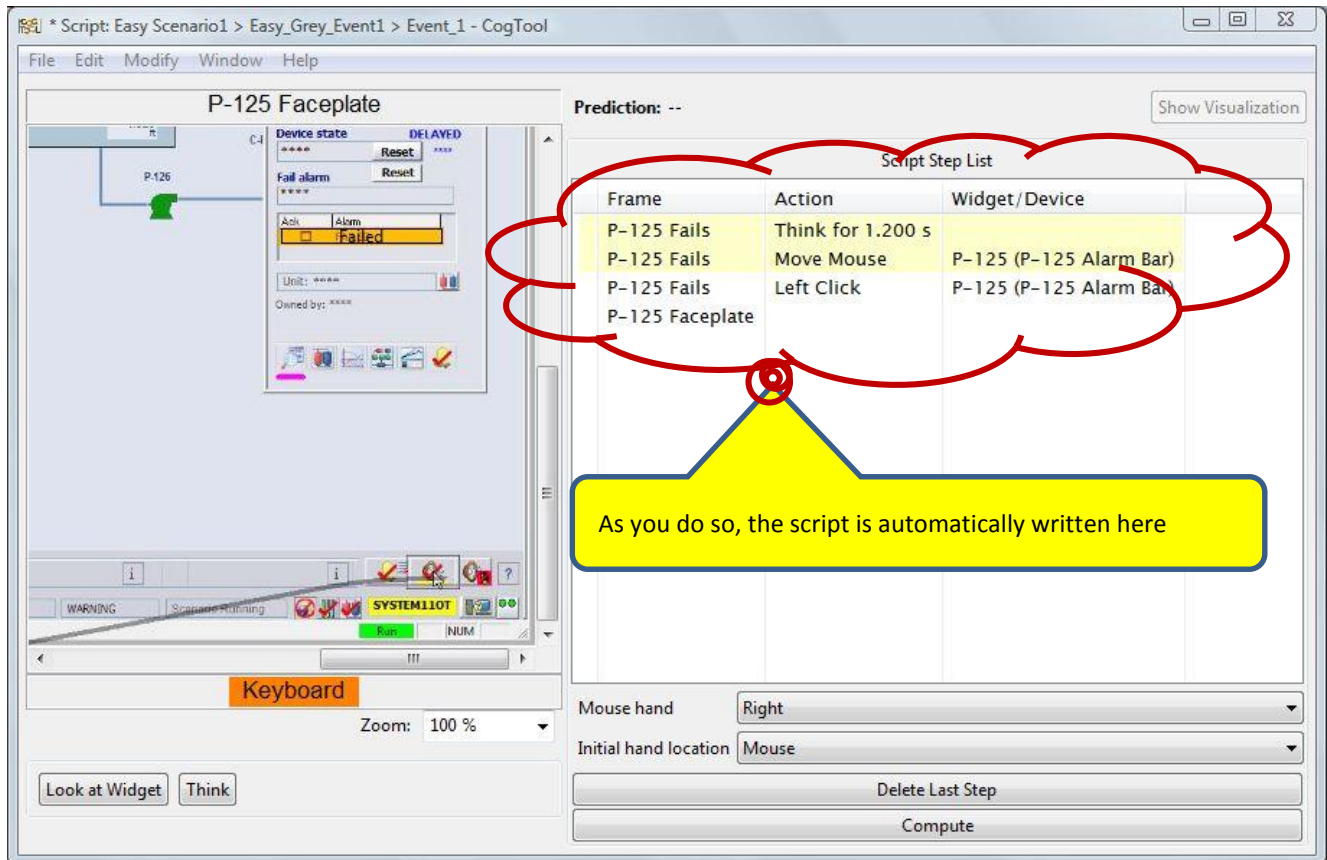


Fig. A. 16. CogTool Automatically Writes the KLM Script

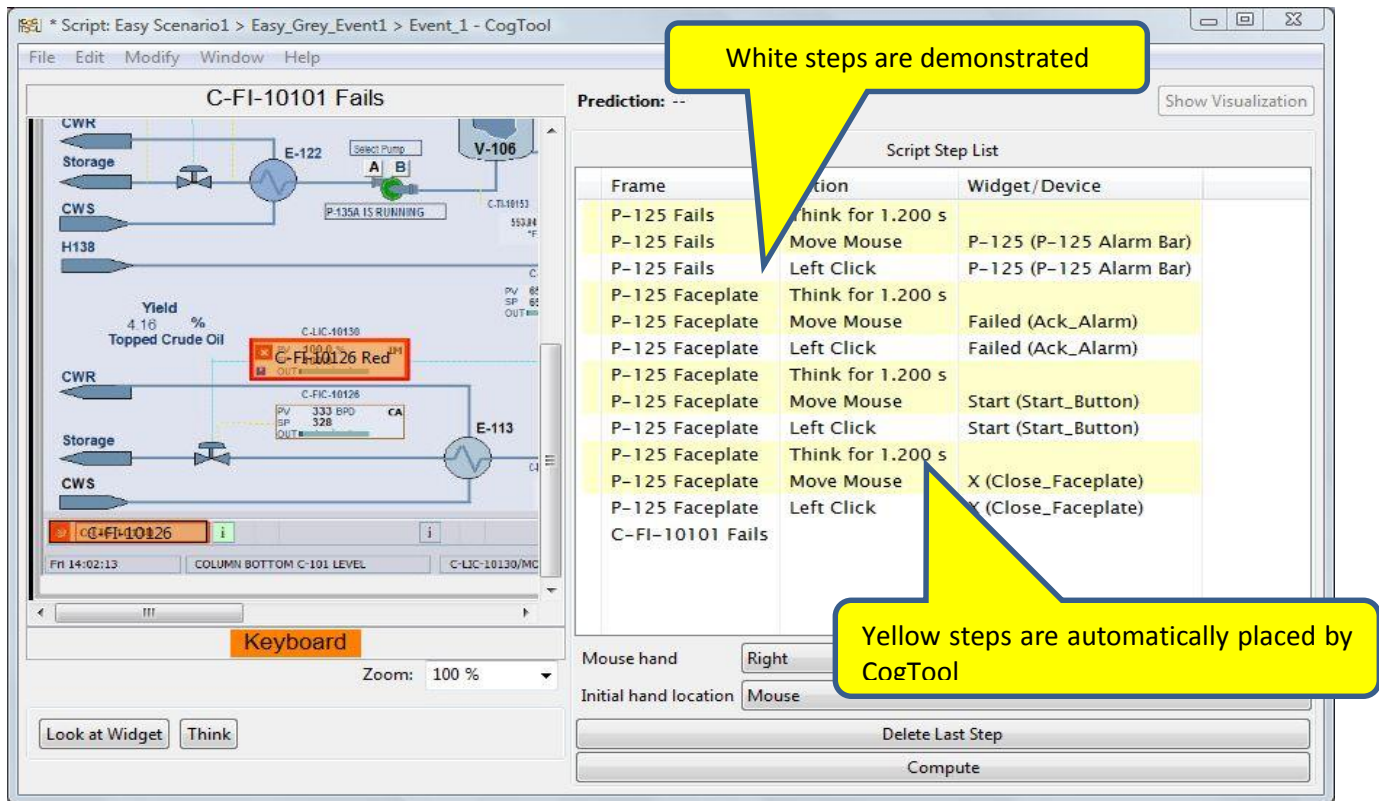


Fig. A. 17. A Snapshot of the KLM Script on CogTool

15. To perform the simulation and determine prime time to complete each specific task on an interface (simulated expert), CogTool runs the cognitive model to produce quantitative predictions of event(s) execution by skilled users from the demonstrations. After the last widget is written, click “Compute” to generate the skilled user time.

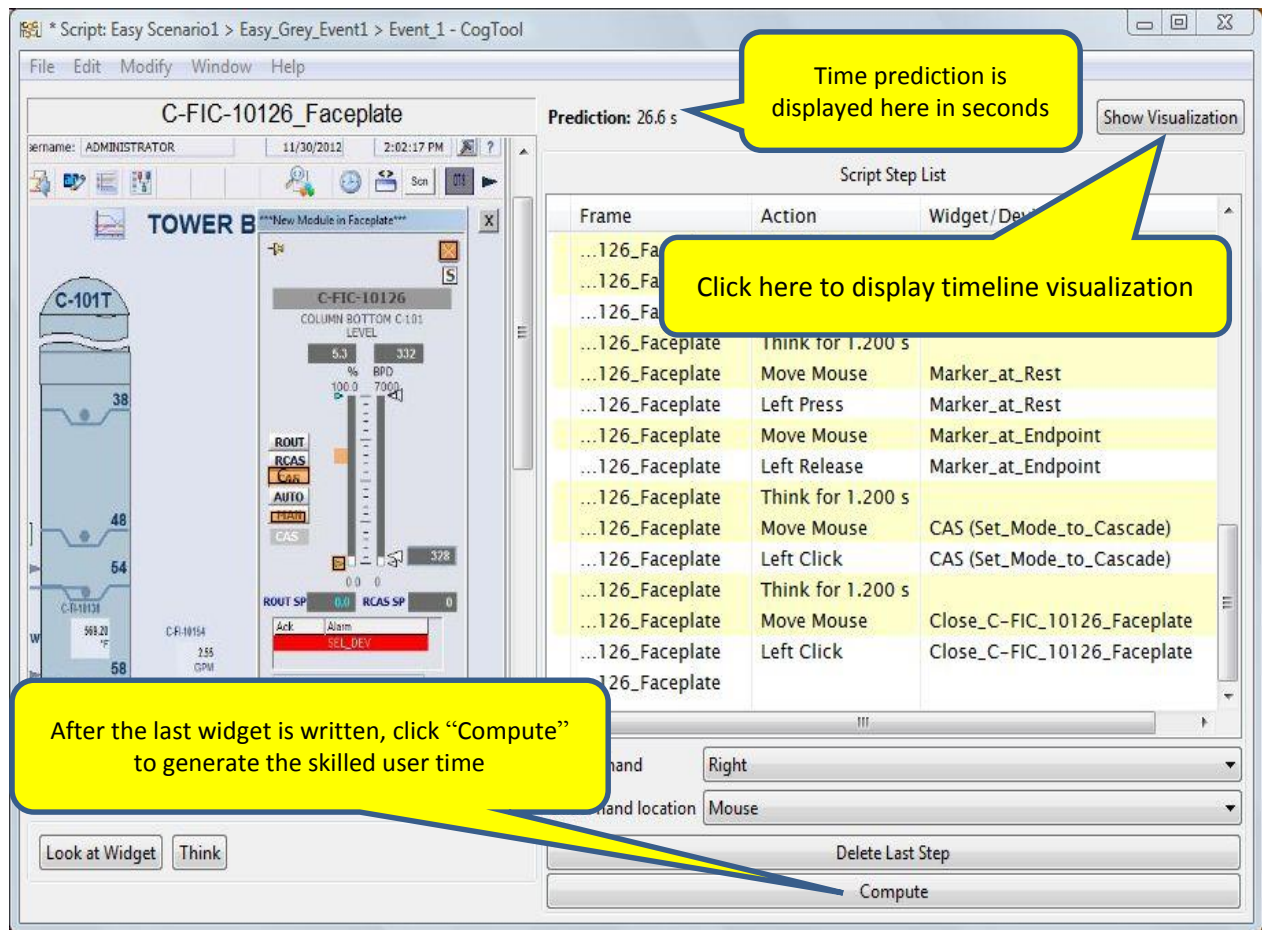


Fig. A. 18. Run the Simulation by Clicking "Compute"

VITA

Oluwakemi D. Adio was born in Ogun State, Nigeria in 1985. She received a bachelor's degree with honors in Civil Engineering at the Nigerian premier university, University of Ibadan, in November 2009. She started her work towards the master's degree in Industrial Engineering in 2012, after working briefly in the construction and banking industry. She also worked as a graduate administrative assistant during her time as a master's student in the Department of Mechanical and Industrial Engineering.