COLLEGE OF INFORMATION SCIENCES AND TECHNOLOGY

THE PENNSYLVANIA STATE UNIVERSITY

UX Design Guidance for Operation Center Design

Literature Review for Critical HMI in Op Centers project with Harris Corp.

Jacob D. Oury
Oury.Jacob@psu.edu

Frank E. Ritter
frank.ritter@psu.edu

Technical Report No. ACS 2018-1
Revised 10 October 2019

Applied Cognitive Science Lab
365 IST Building
College of Information Sciences & Technology
University Park, PA 16802
acs.ist.psu.edu

Phone +1 (814) 865-4455 Unsecure Fax +1 (814) 865-6426

UX Design Guidance for Operation Center Design

Jacob D. Oury (Oury.Jacob@psu.edu) Frank E. Ritter (frank.ritter@psu.edu)

College of Information Sciences and Technology

Penn State, University Park, PA 16802

Last Updated: 4Oct19

Abstract / Executive Summary

Operation centers rely on an interconnected process involving complex technological systems and human operators. Designers should account for issues at possible points of failure, including the human operators themselves. Compared to other system components, human operators can be error-prone and require different knowledge to design for. They also typically exhibit a wider range of performance than other system components. This document provides a design approach, methods, and design principles to use when designing to improving human performance within a complex system that is an operation center, and thus to improve the performance of the system itself. The principles are supplemented by text offering a primer on relevant literature on human cognition relevant to operation centers and example design documents. This report is designed to be useful to operation center designers and implementers.

Acknowledgements

This report is produced under a contract to L3Harris Corporation. Mark Foster was the primary designer of the Water Detection System used as an example in this report. We thank Mark Foster, Melissa Rhonemus, Jim Ringrose, Alison Sukolsky, and Tom Wells for useful discussions during the development of this report. We thank Raphael Rodriguez, Pooyan Doozandeh, Chad Chae, and Cesar Colchado for comments on the report. The opinions are those of the authors and do not necessarily represent those of L3Harris.

op centers HCI reviewV15.doc

Table of Contents

1. Introduction

Operations (Op) centers are a vital communication hub for the transfer of information to and from the frontlines of any situation. Within any given op center, there are going to be different priorities, tasks, and stakeholders that must be considered in their design. This book primarily examines operation centers that manage autonomous, asynchronous systems.

This report is designed to be useful to op center designers and implementers. Managers can use it to adjust the process to take account of a wider range of risks due to not supporting users and their tasks. Designers can use it to manage the process and be aware of useful types of shared representations. Implementers can use it to provide context for small decisions within an interface that are too small be described formally or have not been specified. Where we can, we also identify design principles. Design principles are aspects of the operator or interface or process that suggest prescriptive actions to create better interfaces.

This introduction will first make the case for including users as part of the system and then part of the design process. It will then briefly describe a way to include them (the Risk-Driven Spiral Model) and how it could be applied to operation centers. The rest of the document will use an example system called the Water Detection System (WDS) to help illustrate the principles, concepts, and practical implications derived from the material covered. The introduction concludes with some example guidance that can be noted as an executive summary or a summary for readers who might not have time to read the whole book. The remainder of the book provides support for the guidelines. The appendices include a worked example.

1.1 The role of operators

Operators can greatly influence operation center success. In a study of errors in Air Traffic Control, a type of control room, Jones and Endsley (1996) found that seven out of ten times system failure is due to operator error. The error analysis for aviation disasters organized the contributing errors into the Endsley's (1995) theory of situation awareness. When the errors were organized into their stage of situational awareness, they found that misperception or non-perception of the necessary information was the primary cause about 75% of the time. Going up in complexity, failures to successfully comprehend meaning or importance of information was the primary cause in only about 20% of air disasters. Finally, at the lowest error rate, projection of system state in the near-future is key in less than 5% of disasters. Breaking down these failures into more specific categories of failure cases shows that attentional failure (35%; operator has information but fails to attend to it), working memory failure (8.4%; operator attends to information but forgets it), and mental model failure (18%; operator's understanding of the situation does not match reality) account for the most common causes of operator error in op centers.

Operators of complex systems use a set of cognitive mechanisms that are fallible in predictable ways. System engineers, developers, and designers can begin mitigating the risks associated with fallible cognitive behavior by learning about the many factors and mechanisms that influence operator performance and reliability. Not all these mechanisms can be ameliorated by system design, but they do suggest areas where systems could support operators. This report will suggest ways to do that.

Modifying op center designs could help reduce these types of system failures by providing the information more clearly, more comprehensibly, requiring less attention (perhaps by reducing other less useful information), and appropriately matching and supporting the operator's mental

model and tasks. How can these issues be addressed throughout the development cycle of complex systems? We propose a design process based on understanding the operator, their tasks, and the technology.

1.2 How to improve designs

The variety and complexity of work being performed in op centers prevents strict design guidelines from being a "silver bullet" for every system design issue. The different goals, priorities, and tasks across op centers will likely add up being nearly equal to the number of op centers itself. However, the common element across op centers is the role of human operators. Operators serve as the interface between the wide range of information sources and the higher command structure. This can involve a vast variety of tasks ranging from call intake and prioritization within an emergency response center to monitoring a radar for airborne threats. Furthermore, the task variety is compounded by having a single operator be responsible for multiple tasks. For example: an operator at a 911 dispatch center will often be simultaneously responsible for (a) providing emotional support and guidance to the caller, (b) recording crucial information about the situation, (c) alerting appropriate emergency responders, and (d) answering questions for emergency responders while en route.

Simply providing a set of design guidelines will not suffice because one size does not fit all. Due to the varied nature of tasks and systems across operation centers, we will need to provide a suitable foundation for designers to guide their decision making when there is no clear solution. Thus, this report summarizes a best practice process and design issues to keep in mind when designing operations centers. This report goes further, however, by providing a worked example of design and design steps for an example system.

This report spends more time defining a useful interface design process than giving simple guidelines for design. This user-and-task-oriented process should lead to better interfaces that support operators and do this in a better way than simply providing and following a set of 10 'rules' about font size, which might need to vary, and which will conflict at times with rules about how many objects to put on a screen. And, yet, in providing background knowledge about operators and their tasks, there will inevitably be nuggets of results that look like and work like guidelines.

The complexity and variety of tasks within an op center means that the system designers will need to know their users, their users' tasks, and the technology and then combine these using their own judgment within the design process. They will have to use judgment when aspects of the users and their tasks are not fully known. They will also have to use judgement to prioritize tasks or user types and to balance different design requirements.

The design recommendations offered by this report will often provide "safe" recommendations for designers. Design recommendations will be accompanied by brief supporting details meant to substantiate the information. This self-contained document will provide system designers with a framework for improving user experience and performance by incorporating human-centered design principles into the design and implementation of critical systems.

System designers will benefit greatly from understanding the foundational concepts and literature that support this guidance. This report provides a simple review of the literature to support this guidance. This review serves several purposes: (a) offering motivation for including the topics chosen, (b) describing the related research that has contributed to the high-level guidance, and (c) providing readers with a convenient method to learn more about a topic if needed. While not every system developer will choose to read this document, it provides

interested readers with a more condensed treatment than available from reading several books on user-centered design and users. The final review and guidance document will be detailed enough to provide further guidance in a standalone format.

1.3 Risk-Driven Design

The design and performance of an operation center will depend on financial considerations, task constraints, and the goals of the designers. However, clearly there are limitations to what is possible for any given design process (e.g., deadlines, access to user testing, ambiguous information). In an ideal world, every project would have ample time, personnel, and funding to be able to create the best product possible: clearly this is an unrealistic scenario. Thus, designers and other stakeholders must make decisions about how to ensure project success throughout the design process.

We propose that the Risk-Driven Incremental Commitment Model (RD-ICM) provides the best framework for creating systems including assessing the risks associated with design choices (Pew & Mavor, 2007). Implementation of RD-ICM involves assessing the risk associated with a given decision. Boehm and Hansen (2001) define risks within the RD-ICM as "situations or possible events that can cause a project to fail." RD-ICM uses an iterative, flexible procedure to prompt the stakeholders to make candid assessments of what the risks are at each stage of the project. Implementing RD-ICM effectively may lead to decisions contrary to the dogmatic idea that user experience should be a priority at every stage. The RD-ICM in spiral form is shown in Figure 1.

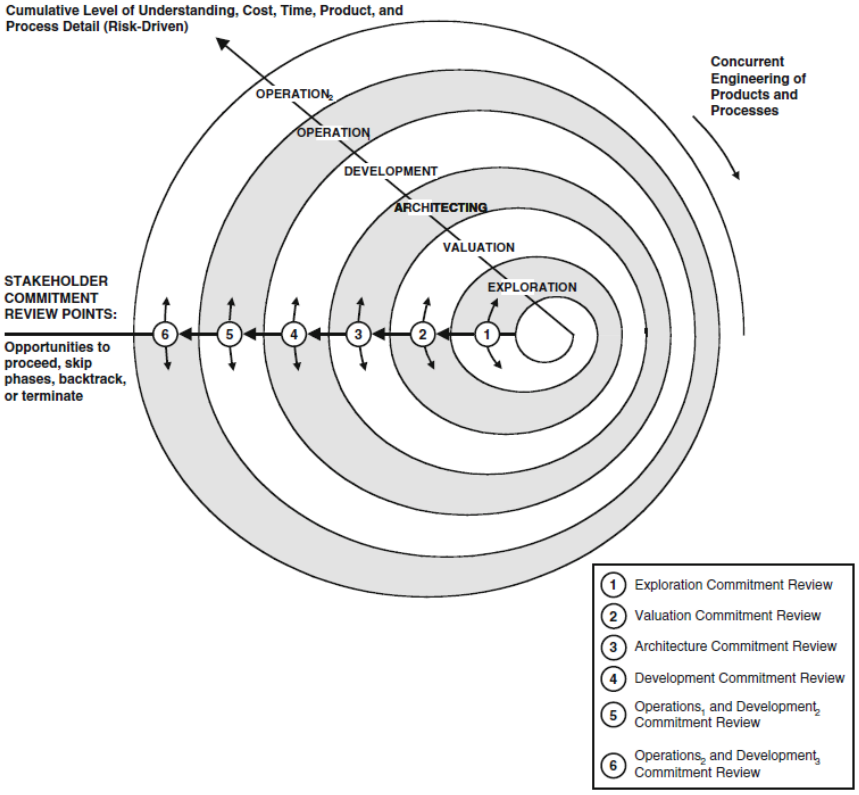

Figure 1. The RD-ICM model as a spiral of development.
Reprinted from Pew and Mavor (2007).

The RD-ICM and risk-driven design require four key features:

1. Systems should be developed through a process that considers and satisfices the needs of stakeholders, that is, provides a good and achievable but not necessarily best solution.
2. Development is incremental and performed iteratively. The five stages (exploration, valuation, architecting, development, and operation) are performed for each project's lifecycle.
3. Development occurs concurrently across various project steps through simultaneous progress on individual aspects of the project, however effort towards each aspect varies over time.
4. The process explicitly takes account of risks during system development and deployment to determine prioritization for resource deployment: Minimal effort for minimal risk decisions; high effort for high risk decisions.

In addition, each stage has phases of (a) stakeholder evaluation and valuation, (b) determining objectives and alternatives, and constraints, (c) evaluate those alternatives and identify and resolve risks, and (d) develop and verify next level product. This approach allows work on risks to proceed in parallel and comes back to value the alternatives with the stakeholders.

Here is an example of how the RD-ICM could shape design choices. During the early design process of a complex system, the risks of not getting the system up and running (e.g., perceived project failure by management, or technical connection issues) may outweigh the risks associated with having a non-ideal interface design (e.g., frustrated users). Instead, the UX design choices could be pushed down the pipeline and then reassessed at a later stage. This would enable the engineering team to focus on creating something that "works." However, once a functional system is formed, the team would reassess the risks. If the interface fails to convey critical information in a consistent manner to most users, now the risks of a user misinterpreting a signal may outweigh the benefits of adding further features to the system. Each stage has its own iterative assessments of how to successfully complete the project. Further information on this approach is available from the National Research Council Report (Pew & Mavor, 2007), a special issue of the *Journal of Cognitive Engineering and Decision Making* (Pew, 2008), and an overview in the *FDUCS* textbook (Ritter, Baxter, & Churchill, 2014).

So, if you accept a risk-driven process that includes human operator related risks, you still must be able to recognize and reduce these risks. This report seeks to provide background knowledge to help developers judge and ameliorate the risks to system success that developers face during the design and implementation process of control rooms. We hope to provide knowledge and guidance that can help designers understand how their design choices may affect task performance throughout the lifetime of the system.

Thus, we suggest following a risk-driven spiral model. This includes formal reviews with stakeholders at each cycle to assess risks, and work focused to reduce risks, not just build a system. This approach uses a range of design documents as shared representations between the stakeholders and the designers and implementers. We include an example set in Appendix 1.

1.4 The design problem space for op centers

This document reviews how risks for failures due to human performance can be alleviated throughout the design process of interfaces within operations centers. Thus, designing an interface for an op center is the design problem. We briefly review this design space and provide an overview of an example before addressing further risks and issues that are common based on users.

Op centers act as the nervous system within a larger body directed to monitor or respond to a set of events. The op center aggregates information input and output to facilitate a rapid response to changing conditions. The specific procedures used are typically guided by senior staff while operators themselves will be responsible for interpreting information, transmitting orders, and following preset procedures for specific situations.

There are three components to this design problem: the technology to support and implement the system, the users, and the users' tasks. The first item is briefly noted as an important component that will support and constrain designs. The final two are the focus of this report so we address them together.

1.4.1 Know your technology

The most important aspect, as noted elsewhere, is perhaps that the system itself works. That technology allows the information associated with the op center to be obtained and presented to an operator. Most designers of technology will be familiar with these aspects. So, the first issue in design is to know what the technology can and cannot do. This technology includes sensor and communication systems. This technology also needs to include interface design and display systems. These systems (technology and interface) are likely to be different from other technology systems and may require different designers and implementers. In addition, the interface tools need to be able to support the designers to create usable interfaces, which not all tools support well (Pew & Mavor, 2007; Ritter et al., 2014).

Knowing these two technologies will help with the inevitable choices about fitting the man to the machine vs. fitting the machine to the man. Sometimes, it will not be possible to produce fits in one of the two directions. Knowing the technology will help reduce problems in both directions.

1.4.2 Know your users and their tasks

The focus of this report is to explain how to know the users of the op centers, the operators, and how to know their tasks. Human operators and their tasks will in many cases be as complex as the technology. The only difference is that many technology designers have been trained in technology, but not trained in the science of how operators think, learn, and do their tasks. This report notes some of the literature, results, methods for understanding operators to help in design. Similarly, it will describe task analysis, which is a useful tool for specifying, implementing, and checking op center designs.

The technology may be able to deliver, but will the operator be able to understand and use the system at speed? Will the tasks, including their microstructure and dependencies be supported? Or, will the operator have to correct and store information (in a more fragile memory than computer memory)? These types of mismatches between operator and system are a cause of system failure.

The gold standard in design (Card, Moran, & Newell, 1983; Pew & Mavor, 2007; Ritter et al., 2014) is to know the operators, know what tasks they are trying to perform, and then use the technology as best it can be used, to support the tasks based on the operator's capabilities. Designers that use themselves or their use of their own systems as a reference instead of the actual users commit a fundamental error, that leads to operators not knowing how to use a system. This can be called the fundamental attribution error of design (Baxter, Churchill, & Ritter, 2014), which is where the designer assumes the user is like themselves. As we note in our

example case in this report, this would often be a mistake and lead to problems in usability because the designer and the operator have different knowledge, skills, and abilities.

In addition, leaving out tasks or making them less easy to perform, making state information visible only upon query, are all mistakes that are easily avoided, but require knowing the operator and their tasks.

Knowing the frequency and importance of tasks is also important, so that common and important tasks should be more easily and safely accomplished than less common and less important tasks. When these two factors collide, frequency and importance, then that is where design choices are available, and sometimes it is appropriate to ask the stakeholders to answer these more nuanced questions.

There are numerous guidelines on how to create task analyses (e.g., Ritter et al, Ch. 11, 2014). There are tools to support TA (*i.e.,* Cogulator[1]), but often plain text documents provide the best value and are useful enough for most designs. TA is a lot like pizza–while the balance of contents may vary in approaches, most versions are usable and enjoyable.

1.4.3 Test designs broadly and with cognitive walkthroughs

During design and implementation there may be unknown aspects of the user, their tasks, or the interaction of these two components with the system. A way to reduce the risk of system failure is to test the resulting system. This test can be quite simple, for example, simply to see if the tasks can be performed, but it could also include many more formal types of tests. Pew and Mavor (2007) review the range of these tests, and there are multiple textbooks describing them (e.g., Cairns & Cox, 2008; Lewis & Rieman, 1994). These tests reduce a range of usability risks and take a wide range of types and amounts of resources.

The simplest test is to have naïve operators use the interface and observe them. This approach is explained in many textbooks, including Ritter, Baxter, and Churchill (Ch. 13, 2014). They can take from 10 min. and cost next to nothing, when you ask a colleague to use the interface and take away a small insight to $100k when you formally run many users at their own site and prepare a formal report.

We also suggest using what is called a cognitive walkthrough (Polson, Lewis, Rieman, & Wharton, 1992) to see that the tasks indeed be performed. Cognitive walkthroughs are a method for evaluating the learnability and usability of an interface by simulating the cognitive activities of the user during normal tasks. The typical method for performing cognitive walkthroughs begins with describing goals and tasks that are required by the system. First, the goal structure of the model is generated from expert interviews, prior research, and other forms of information gathering. The goal structure, like a task analysis or using a task analysis, is arranged in a hierarchy. The top goals represent the overall task. Each top goal is composed of intermediate level goals (*i.e.,* subtasks), each of which is composed of a set of individual actions.

Cognitive walkthroughs, when performed successfully, should determine whether the operator of a system is making the correct connections between each level of the goal. That is, the analyst applies the goals to the interface to produce the behavior. If the analyst cannot make the mapping, it suggests an area for improving the interface. If the analyst is too familiar with the interface, then they will not see problems users will see, at least novice users. The data collected from cognitive walkthroughs can enable developers to provide supplementary "clues" or signals to the operator at specific locations to ensure that each goal, sub goal, and individual action

[1] http://cogulator.io/

provide a coherent information set capable of being understood and followed by the operator (Blackmon, Polson, Kitajima, & Lewis, 2002; Polson et al., 1992).

Cognitive walkthroughs require a task analysis, and then appear to take between an hour and a short day to perform. The length of time is based on the number of tasks and how difficult they are to perform. They may require domain knowledge, and thus may be done in teams, the analyst working through the task analysis and the domain expert making the decisions.

Where detailed time predictions are useful, we recommend using the Keystroke Level Model by Card, Moran, and Newell (1980; 1983). This approach provides time estimates based on the keystrokes, mouse moves, mental operators, system response time, and other operators. The times are engineering estimates (*i.e.,* +/- 20%), but are basically allow fair comparisons of different interfaces and make suggestions about where time is being spent and could be reduced. The regularity of the interactions across subtasks also suggests how much needs learned by the users and where knowledge may be misapplied.

There are numerous ways to reduce system failure due to usability problems. This section noted a few and how to find more. Next, an example system is introduced to ground this discussion with examples of how potentially abstract principles can be put into practice.

1.5 Example task: The Mars Water Detection System

This paper will use a hypothetical use case to provide context for readers. The scenario is based on designing an op center for command and control of a remote Water Detection System (WDS) to accompany a manned mission to Mars. The WDS will arrive alongside the mission team and begin operation following its assembly by the team. Following its activation and an initial system check, the op center on Earth will take over sole command of the WDS for a 10-year mission. Scientists in the program office will make large-scale decisions to support the mission of finding water, while the Earth-based operators implement action-plans and monitor the various systems for any current or upcoming issues. We next detail select information to offer an overview of the example scenario. A more detailed description is found in Appendix 1.

1.5.1 Operation center organization

The WDS is one part within the larger structure of an op center hosting dozens of systems that require constant oversight. While the WDS is important for the mission, it may not be the primary focus for the workers at any given time. The command structure of the op center involves bi-directional communication between scientists from the Program Office who funded the WDS and the operators responsible for direct interaction with the systems. Figure 2 shows example possible interface designs for a system like the WDS. While the design will vary depending on the needs of the system, these systems often will present many different metrics for system performance. Operators will monitor the system, pass along alerts, and update the alerts depending on their risk assessment for a given situation. Scientists will take this information and pass back commands for the operators to transmit. Certain tasks will be able to be completed without direct contact with a supervisor, while others will need direct response from supervisors prior to action.

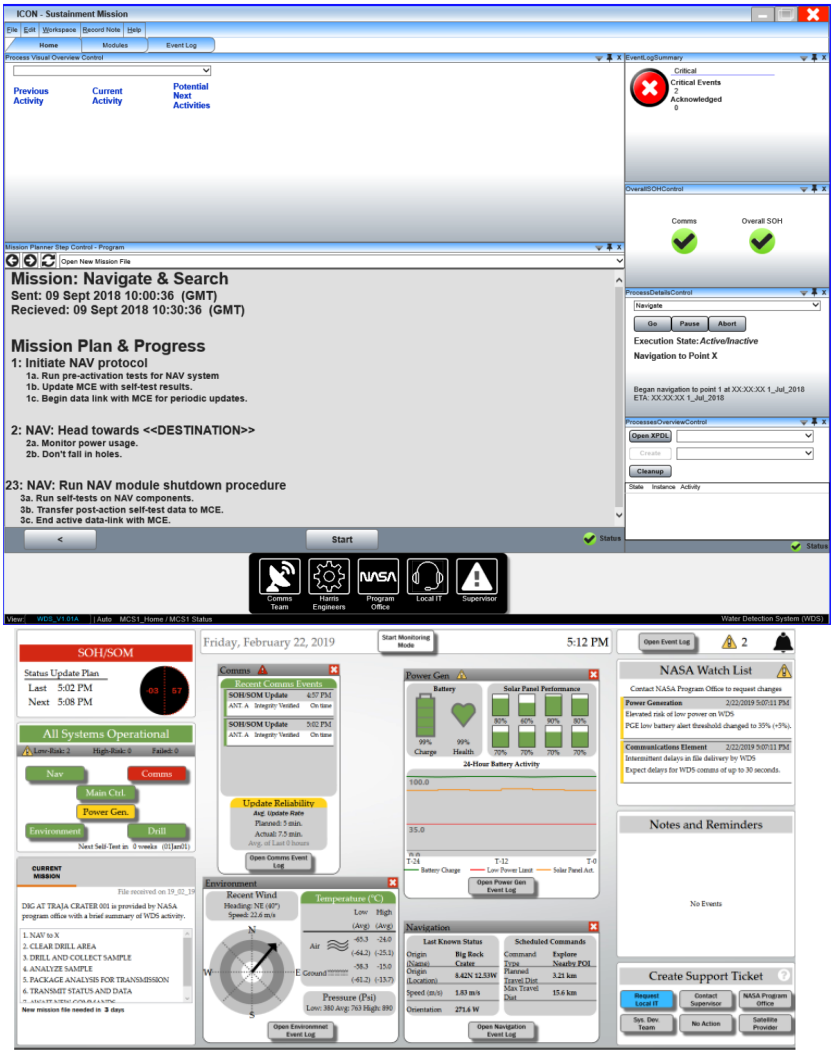

Figure 2: Two example interface designs for the Water Detection System monitoring screen.

1.5.2 Water Detection System structure

The WDS is comprised of several subsystems. The core system in the WDS is the Main Control Element (MCE). The MCE acts as the brain in the field by enacting orders from earth, monitors other subsystems, and linking the subsystems together. The additional subsystems each perform specialized tasks (e.g., communicating with Earth, navigating the WDS, or collecting physical samples), however, all subsystems share a set of key features that the operators may interact with over the course of the mission. These features are shown in Table 1.

Table 1. Key features built into each subsystem of the WDS.

Feature	Description
Status	The current state and functionality of the subsystem, subsystem-specific information, and environmental measures. The MCE checks and stores the status of other subsystems until information is passed to earth.
Event Logs	Each subsystem records detailed event logs from all executed commands. Event logs are periodically transferred to the MCE before being passed to earth.
Configuration	Subsystems maintain a set of configuration fields that determine how the subsystem performs its tasks. For example, the MCE will have a modifiable field for checking a subsystem's status that determines how long to wait for a response before initiating troubleshooting procedures.
Commands	Commands for subsystems will include a time reference and may include additional data if needed. Commands are first sent to the MCE before being passed to the appropriate subsystem.
Redundancy	Nearly every subsystem has an A and B side to provide a backup element in case of any issues, however only one side of each subsystem operates at any given time. These redundant systems are an identical copy of the original system.

1.5.3 Example issues

System designers may be unable to anticipate every problem within a system; however, the risk-driven incremental commitment model drives the designers to try to understand what issues are most likely to arise. Table 2 shows some example problems that could arise throughout the lifecycle of the WDS system, the risk of these problems occurring, the solution, and who handles them.

The WDS is designed to autonomously handle most issues that arise, but human interaction is required on a regular basis. Many of these tasks are simple maintenance and acknowledgement of warnings. For example, when batteries are low, the operator is required to acknowledge the low battery threshold. No action is required other than clearing the notification. Occasionally, however, the WDS will face an urgent problem that requires human input. These scenarios are rare, so the operator has limited training in how to address the issues.

Table 2. Example problems faced by the WDS that require operator intervention.

Problem Description	Risk	Solution	Personnel
WDS is navigating in a crater and gets stuck. The operators need to escalate the issue quickly because the WDS witnessed unexpected terrain. The mappings of Mars must be updated appropriately.	High	Operator from Earth takes over navigation and assumes manual control. The typical operator is not trained in this task, so the supervising manager must take control.	Operator, Supervisor
Dust storm prevents batteries from charging. MCE cannot complete all the scheduled commands for the day.	Moderate	CE alerts the NASA operators of the low battery status. Operator must retask the day's commands because the ANE would use all the remaining power.	Operator, Supervisor
Wall of Screens has many other systems represented at the same time. If the WDS has a problem, it might take a few days for the engineers to remote in to fix the issue. Therefore, the overview screen will remain in degraded state. The problem arises when something else goes wrong on the system.	Low	Modify interface to facilitate proper information presentation. While issues may not be initially present, the possibility of other errors being missed due to clutter is increased.	Operator

1.6 Principles for design

Based on the target system description, the example system, and the design process, we can provide an overview of the report as a set of design principles. These principles provide guidance on high-level concepts that the designers can use to improve the systems they create. Though generally directed towards improving performance across the human-machine interface, these principles will often apply to the entire process of designing complex systems.

Principle 1: Don't assume the user is how you think you are.

One of most important considerations for designers is to dispel the assumption that your users are just like you or how you think you are (we make the distinction because you might not think or work exactly like you think you do). Unless your user is a software or systems engineer, you will in nearly every case need to adjust your design to meet the operator's system-related needs, capabilities, and wants (in that order), and how they are different from you.

Designers often (perhaps due to the ready availability of themselves and the unavailability of example operators) make the risky assumption that the operator is just like them—this is almost never the case. What this suggests, then, is to provide designers and engineers the ability to consult users and other stakeholders throughout the design process. Knowing users can include talking with them, watching them work, having them use your interfaces, reading their

biographies, or watching movies about their work environments (whether documentaries or even fictional accounts).

Understanding the operator enables engineers to mold the system design around the capabilities and constraints of its operators. Countless studies have shown that engineers often fail to understand their users. This knowledge is the foundation of user-centered design and leads to increased performance, financial savings, and safer systems (Lewis & Rieman, 1994; Pew & Mavor, 2007; Ritter et al., Ch1, 2014).

Principle 2: All design choices have tradeoffs. Don't go in blind.

Most design choices have tradeoffs. This basic fact will provide engineers with difficult decisions throughout the design process. For example, a larger font means less on the screen; more on the screen is often helpful. How to resolve this design choice requires knowledge of the task and operators. Use of the risk-driven spiral model helps engineers make the best decision given the constraints by consulting with stakeholders and using what has been learned by others.

For example, recognition memory is more robust than recall memory. While searching for files on a system, it is usually easy to point-and-click around a series of folders to find some item. Using a keystroke-based system (like a command line) might be faster but will require either more skilled users or more training.

As another example, speed and accuracy have conflicting methods for improving performance. Emphasizing speed will often require sacrificing accuracy (i.e., more errors). While ideal solutions are not always possible, designers can meet expectations by understanding the expectations for task time and error rate.

Many studies have explored how users' decision making, reaction time, and error rate change in response to changing task decisions. The Hick-Hymen law predicts that choosing between more options (e.g., 3 choices in a menu vs. 5 choices) takes longer, but the menu is more likely to contain the correct choice. Signal detection theory shows a similar tradeoff between hits, misses, false alarms, and correct rejections.

When possible, engineers should make informed decisions about the tradeoffs between outcomes caused by different design choices.

Principle 3: Use multiple designs.

When designing a new display or component, create and consider multiple versions. Get feedback on the possible designs from as objective a source as you can.

When you create a new display, particularly high stakes or main displays, you should consider multiple versions. Considering multiple versions of designs tend to lead to better designs at least in the tasks that have been studied (Dow, 2011). The best objective source for feedback is often actual users.

Research by Steven Dow (originally at Stanford, then CMU and now UCSD), examined design in the egg drop task. In this task, designers were given a set of materials and asked to design a protective cradle for the egg to be dropped in. Groups that designed more examples and that tested more often had reliably higher distances that their eggs could be dropped. He argues that this will apply to other design tasks, and we agree.

1.7 Summary

Throughout the design of an ops center such as the WDS system and interface, the engineers' top priority will be the creation of a working product. However, engineers must account for the

risks associated with all aspects of the project. Often, the risks associated with some module's reliability or function may trump the human element: human error requires a task that can be failed. However, as the iterative design process advances, and the technology itself becomes more reliable, the human operator becomes more likely to be the point-of-failure within a system. System engineers will be neglecting a crucial component of their system if they do not account for the system's compatibility with the human operators. While this process will have any number of constraints and variations in its implementation, the designers should be confident that their system can be effectively used by the target population. The user interface should facilitate high performance without undue stress on the operators.

Table 3 notes some questions that designers might have in mind when designing and implementing control rooms. In the conclusion to this report we will note how the review has answered them. The book now describes the factors that give rise to the principles in Chapters 2 and 3. An example application is provided in the appendices.

Table 3. Questions to be answered by this document for systems like the WDS

Process Performance

1) What user interface features reduce user stress and improve and maintain level of performance?
2) Which user interface design factors mitigate performance degradation (speed, accuracy) during the execution of detailed procedures for trouble shooting?

High Throughput Reaction Times

3) What levels of fast and complex interfaces impair or enhance user reaction time and accuracy?
4) What are the reaction time and accuracy for a user to react to an alert and respond to the alert with the correct actions using the task UI? What are the upper limits of number and speed of alerts before performance degrades?
5) What are the reaction time and accuracy for a user to distinguish between levels of criticality using the task UI?
6) What are the effects of time on reaction time and accuracy for a user using the system?

Interface Generalizable and Individualized Effectiveness

7) Which interface design elements vary and do not vary in effectiveness across various demographics?
8) Which of the above questions are affected by age and prior education?

2. User-Centered Design and Situation Awareness

The full gamut of factors that can contribute to the success of an interface is difficult to describe within a single report. Instead, we will identify some of the most significant factors that can be used by engineers during their design of an interface. For a more comprehensive review, we recommend: (a) *Foundations for designing user-centered systems: What system designers need to know about people* (Ritter et al., 2014), and (b) *Designing for Situation Awareness: An Approach to User-Centered Design* (Endsley, Bolte, & Jones, 2003).

This report offers design guidelines for optimizing the performance of the human component of the operation centers for asynchronous, autonomous systems. User-centered design (UCD) provides the foundation for this task through basic tenets of its design philosophy. Designers can achieve UCD by designing for situation awareness (SA, explained below) in operators. Guidelines developed in these chapters will provide concise takeaways while selected information on related cognitive mechanisms will provide context.

Thus, this paper will follow this logic. First, we describe the tenets of UCD. These provide high-level questions that engineers can apply to their system at any point in the design process. Next, the connection between operator performance and SA is explained. Performance levels of SA correspond with cognitive mechanisms used to perform a task. The final section describes the cognitive mechanisms, their influences, and offers design guidelines for ensuring compatibility between user capabilities and system interface.

2.1 User-Centered Design

The operator is a component of the system just like the sensors or underlying code. High-performance systems will incorporate operator capabilities into their design. This requires creating a system that follows principles of user-centered design. Though UCD is often associated with user-experience, *Designing for Situation Awareness: An Approach to User-Centered Design* (Endsley et al., 2003, p. 5) explains their difference in underlying philosophy:

> *User-centered design challenges designers to mold the interface around the capabilities and needs of the operators. Rather than displaying information that is centered around the sensors and technologies that produce it, a user-centered design integrates this information in ways that fit the goals, tasks, and needs of the users. This philosophy is not borne primarily from a humanistic or altruistic desire, but rather from a desire to obtain optimal functioning of the overall human-machine system.*

The three primary tenets of UCD, shown in Table 4, describe the high-level goals of UCD. Each tenet is expanded over the next few pages alongside some explanation and examples.

Table 4. The Central Tenets of User-Centered Design as summarized by Endsley, Bolte, et al., 2003.

1. Organize design around the user's goals, tasks, and abilities.
2. Technology should be organized around the way users process information and make decisions.
3. Technology must keep the user in control and aware of the state of the system.

To illustrate these tenets, consider driving as an example. Figure 3 shows a car's dashboard. With respect to Tenet 1, what are the primary and secondary goals of the user when using this interface? Design should reflect the importance of each goal. While operating a vehicle, the primary goal is to arrive safely at the location while balancing time as a secondary goal. Consider how the dashboard shown in Figure 3 matches the goals, tasks and abilities of the driver. The speedometer is large and detailed, providing a quick reference of speed while driving. This is the primary gauge that will be used while in motion and prominence in the display. The large tachometer provides instant feedback about the operator's input on the system, but without the same detail as the speedometer. Broad markings and the red line provide simple indicators of system state. Engine temperature, and gas level gauges are small because they have relatively minor or infrequent usage. Red lines indicate when direct action needs taken.

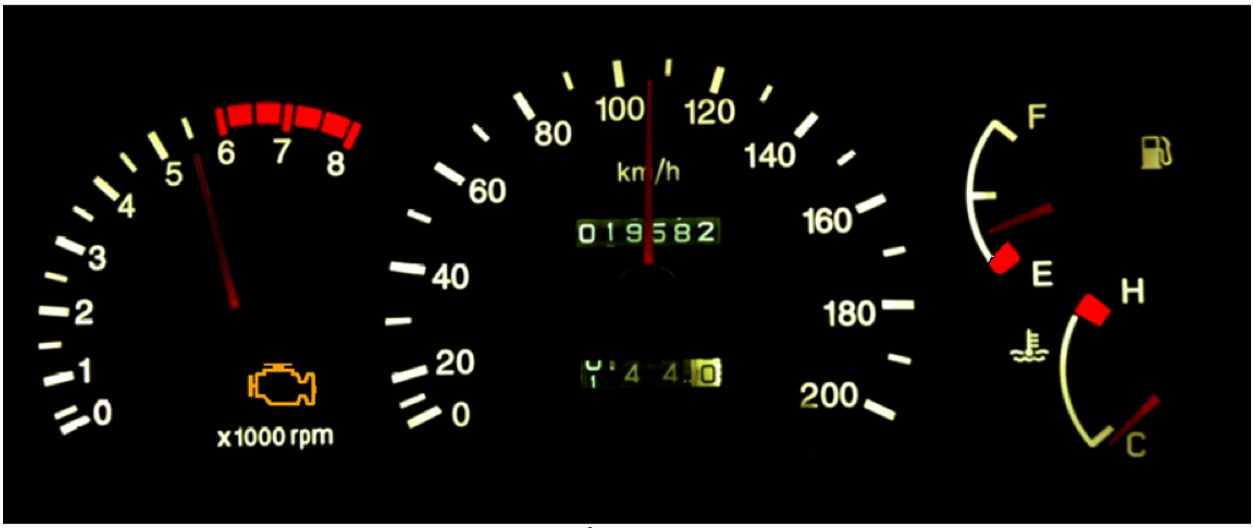

Figure 3. Images of a basic automobile dashboard [2]. The full dashboard shows four gauges from left to right: tachometer, speedometer, fuel level, and temperature.

What are the primary and secondary tasks that a user will perform on this interface? The design should reflect the importance of each task. While driving, the primary task for this interface is checking the speed. Secondary tasks are monitoring the overall state of the vehicle. The speedometer has detailed markings to approximately match speed limits (10 mph increments). The tachometer only provides broad details and a red line indicating a "unsafe state", matching the detail that a user requires for monitoring the state.

With respect to Tenet 2, the information in Figure 3 is makes the speed easy to see, both to find the indicator and to find the speed it represents. The other information for less important tasks is given less room. Where exact numbers are needed, such as miles traveled, this is provided as a number. Would a typical user be able to understand this system? Users and designers are often not the same skill level. In the case of a car, the average driver is not a mechanic, so they often do not need detailed information on issues. An indicator to check your engine may be enough as the layman may not gain any value from additional information. Thus, Figure 3 shows Tenet 2 in practice for the dashboard of a car. For the average driver, the check engine light provides only the necessary information to solve further problems and nothing more.

With respect to Tenet 3, the relevant information is provided to control the system. In this case, a user working through sequential information on a display expects the next area of focus to be

[2] Image from Free Images https://www.freeimages.com/photo/stock-in-car-dashboard-1421520

on a path from left-to-right, top-to-bottom (like when reading). For the state of a car, the water temperature and gas tank level are in a fine order. More complex interfaces may require a different order, and power plant control rooms often order the displays based on their location in the plant.

In Figure 3, if other information not related to driving the car, was presented, such distance from home, type of fuel in the tank, or brand of tire, the driver's ability to drive would be less supported. If the prominence and organization did not match the driver's eye, for example a less clear (or smaller) font, or in a different order, then the driver's performance could suffer. Finally, if the state of the car was less visible, or less appropriately matched to the frequency and importance of goals, performance would suffer.

These tenets are not perfect, however, and do not always give clear guidance. Consider the display in Figure 4. Here, the tenets do not provide direct guidance. The choice between these two designs must be based on the details of the goals and task priorities. If these are not known, they must be obtained from stakeholders (in the best case) or guessed or inferred in the worst case.

Together, the three tenets of UCD provide a foundation for how to frame the system design process around the goals, needs, and tasks of the operators. The various other elements within a complex system have their own design philosophies or guidelines (e.g., modular design, minimal complexity, easy replacement of components). The human-system interface is no different. The tenets of UCD provide an underlying set of principles that should shape the design process for creating complex systems.

Implementing UCD within complex systems requires a method for understanding and assessing operator performance during complex work. Endsley's (1995) theory of Situation Awareness (SA) fills this need by providing a framework for understanding performance and decision making. Describing the SA of an operator means describing the product of relevant cognitive mechanisms that are necessary to perform complex work like decision making and troubleshooting within an operation center.

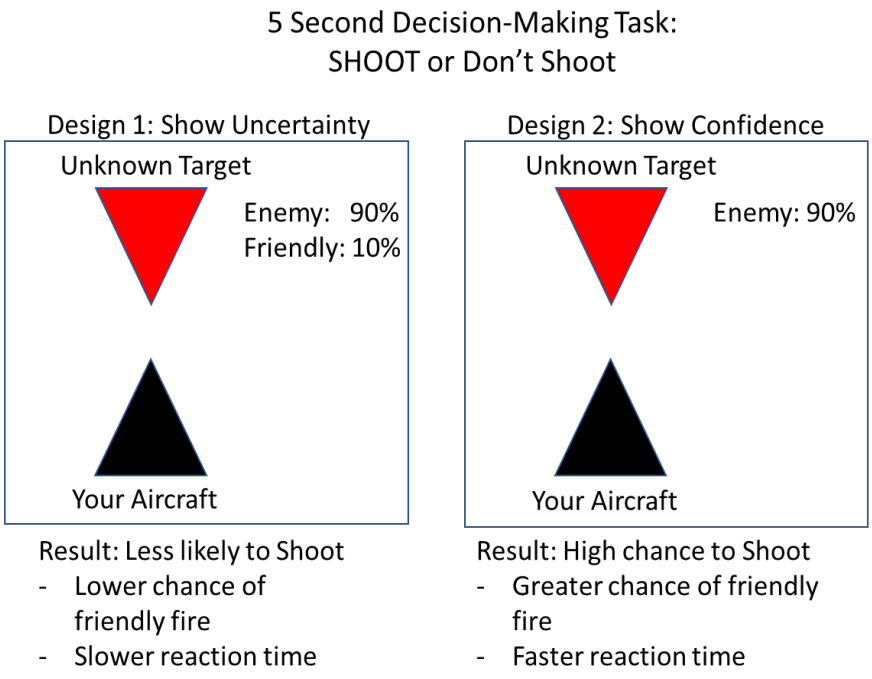

Figure 4. Two ways to present display of an automated target identifier. Each design has trade-offs in operator performance that must be weighed based on the goals and priorities of the system. Image redrawn and modified by authors. Original figure from Banbury, Selcon, Endsley, Gorton, & Tatlock (1998).

2.2 Situational awareness: A key to UCD

Human operators using complex systems must be able to correctly perceive useful information while ignoring other stimuli. Situational awareness (SA) provides a framework for describing human performance at these types of tasks. At its most basic level, an operator with SA understands what the objects are around them, what the objects are doing, and what the objects will do. With these types of knowledge, the operator understands the current state and can project their understanding into possible future states of the system.

Describing an operator's SA performance will use three iterative stages. Though specific performance benchmarks denoting each stage may vary depending on the task, the three stages of SA are typically known as (a) Perception, (b) Comprehension, and (c) Projection. These are illustrated in Figure 4. First, an operator must be able to perceive the useful information from the task environment. Second, they integrate individual cues into a model of the current situation. Third, they use their model of the situation to predict likely outcomes based on their comprehension of the scenario. Figure 5 uses operating a car to provide an example of what types of information are associated with each stage.

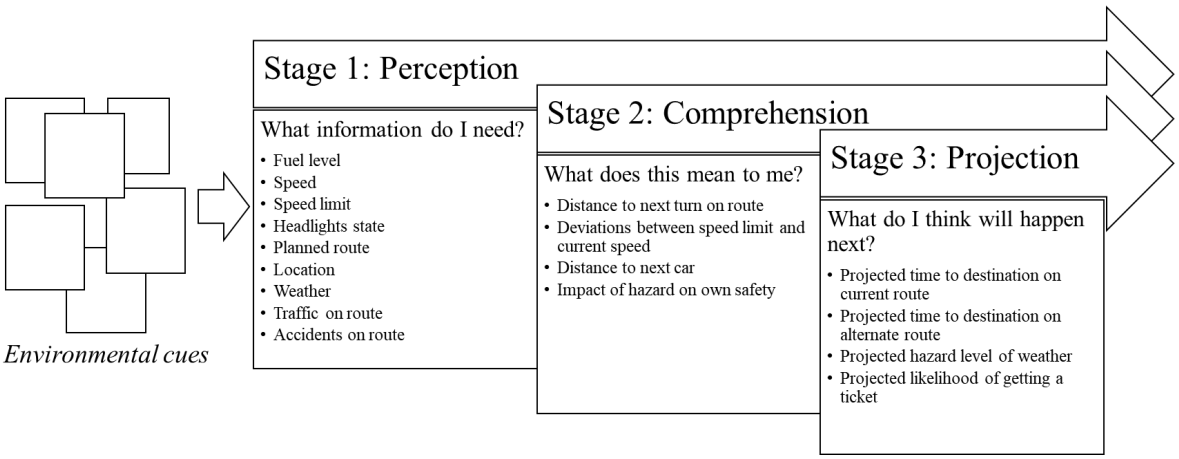

Figure 5. The three stages of SA applied to task of operating a car. Figure redrawn and modified by authors. Figure modified from Bolstad, Cuevas, Wang-Costello, Endsley, and Angell (2010).

Thus, operator performance can be improved through incorporating the tenets of UCD in system design. Improving the UCD of a system requires improving the SA of operators using the human-system interface. The system design will impact how well operators can develop and maintain SA during work. Interface design will affect how quickly and easily operators can advance to each subsequent stage of SA performance and how accurate and complete the operator's understanding is at each stage. Similar to shifting gears in a manual car, the stages progress on a continuous scale where mastery of lower levels of SA is required to advance to the next stage.

The stages of SA provide a framework for assessing performance and identifying task and interface factors that can moderate SA performance. Progression through stages of SA will be impacted by operator characteristics (e.g., fatigue, personal capabilities), environmental effects (e.g., distractions), and task-related factors (*e.g.,* cognitive resources required, task types, complexity; Boff & Lincoln, 1988). Each stage requires significantly more resources (e.g., knowledge, information, time) than the last. Stage 3 SA should not be expected as the norm for every operator or every task, however, it is the most useful.

Next, we describe the stages of SA in more detail and provide principles for design based on using SA as a metaphor for work in op centers. These principles are drawn from Endsley and colleagues (2003) but are created by us to apply SA to the design of op centers. We include motivating examples for each stage. Tasks surrounding aviation were the original focus of SA research before it expanded to include a variety of complex tasks. The discussion of stages will refer to percentage of errors in each stage and these values refer to errors during common aviation tasks for pilots, air traffic controllers, and others. It would be reasonable to assume that similar results would be found for op centers.

2.2.1 Stage 1 – Perception

Perception is the most fundamental aspect of SA. During the common tasks within an op center, operators are bombarded with informational or perhaps more literally, there are many displays that they can view, and on each display, there is a lot of information. The situation and signal content can determine the best course of action regarding how and when to respond to a signal (if at all). Operators with Stage 1 SA will demonstrate the ability to detect important signals while discarding irrelevant ones. Given perception's fundamental role in an operator's

work, it is unsurprising that perceptual issues account for about 75% of errors in common SA work (Jones & Endsley, 1996). Causes of Stage 1 errors can be attributed primarily to human failures (e.g., attentional failure, misinterpretation of a signal), primarily to system failures (unclear or missing information), or some combination of the two.

Some design principles and examples related to Stage 1 SA are shown in Table 5. For example, in the WDS (introduced in Ch1 and explained in detail in Appendix 1), a display can indicate that the battery will be unable to charge at the rover's current position, so the rover will need to relocate. This interface needs to make the point clearly. In a recent example, grades to be inputted were available only through an icon of a man pointing at a screen, rather than the word "Grades". The first icon was not interpretable but would have been clear if labeled "Grades". Often words in interfaces are underused, but are readily interpretable (Chilton, 1996).

Table 5. Design principles related to Stage 1 SA.

1) Make the information available.
2) Make the information interpretable.
3) Ensure the value and salience of each piece of information; eliminate or suppress unnecessary signals.
4) Work around the limitations of human perception and cognition by reducing complexity and workload of the task.

The first principle in this area us to make the information available. This means ensuring the value and salience of each piece of information is appropriate, and to actively drawing attention to important signals while minimizing extraneous stimuli. The second principle in this area is to make the information interpretable by using intuitive, sensible displays. The third principle extends the first two by promoting a hierarchy of signal significance to ensure that the perceived signals are the most useful at any given time.

As an example, reconsider the car dashboard shown in Figure 3. Several design features facilitate Stage 1 SA during typical operation of the vehicle. Compare the prominence of the speedometer and tachometer to the temperature and gas gauges. The operator must update their awareness of speed and engine performance much more often than their awareness of temperature and fuel. Thus, key information is more salient than secondary information.

The design also has a threshold for the gas gauge that provides two indicators of low fuel at dangerously low levels. First, the red line for gauging fuel level compared to the warning point, and second, a light that indicates dangerously low fuel levels. This draws more attention when necessary while always showing some information.

For another example consider the WDS introduced in Chapter 1: when below a certain power threshold, the dashboard interface displaying the battery information will continually flash a red symbol indicating the risk to system power. If this alert continues until the battery is charged, the signal will waste the operator's attention and cause unnecessary distraction. Why does the signal remain prominent, even after the solution has been implemented? Once the solution process begins, there is no need to draw attention to the signal until additional information is received. The signal's visual appearance should be able to be muted until an update is needed.

This principle has further implications for the details of displays. It suggests eliminating or suppressing unnecessary signals and merging compatible signals. Simplify complex signals. For example, in the WDS an interface showing the overall WDS status may include orientation,

geographic location, battery level, and other information. This information is used by operators during periodic checks for unexpected changes in status. Reduce complexity and detail when possible. This can mean only showing integer values for orientation or showing battery level in whole percentage values rather than in the 10,000's that are returned. If warranted, reduce complexity further by simplifying to a binary response for whether orientation is normal or not.

The fourth principle in this area is to work with the limits of human cognition and perception. Human cognition has natural limits in how much it can process at once. Work around the limitations by reducing complexity and workload of the task.

For example, a status update for the WDS may include hundreds or thousands of events in a data log that accompanies the basic system status report. Reserving a space on the interface to indicate critical or alarming events (e.g., imminent power failure) while hiding data related to non-important updates will reduce the amount of information necessary for the operator to perform the most useful tasks.

As another example, some system is rarely interacted with during normal operations. The interface simply provides a status that is checked hourly by an operator. This interface was initially expected to be part of a multiple-monitor display for a seated operator, but now it is checked while standing several feet back. The operator must lean in or squint to read and understand the information.

Consider physical aspects of how the operator uses the system. An operator sitting at a desk in front of the screen can effectively monitor more dense signals than someone five feet away. Ideally, the perceived details of an interface will smoothly transition as an operator views it from different distances.

Books on visual design of interfaces can provide more information in this area (*e.g.,* Kosslyn, 2007; Tufte, 2001, 2006).

2.2.2 Stage 2 – Comprehension

The second stage of SA involves synthesizing Stage 1 cues into a useable mental model of the situation. A practiced operator will detect patterns from various stimuli and form a holistic view of the situation based on their experience with the task and the information presented. Errors arising from comprehension failure account for about 20.3% of errors (Jones & Endsley, 1996). Stage 2 errors are often attributed to misinterpretation of an information set, failure to maintain all the necessary information in working memory, misuse of their mental model, or over-reliance on default settings (i.e., failing to check a status hidden in a submenu).

Some design principles related to Stage 2 SA are shown in Table 6. The first principle is to design the system to prevent misinterpretation of signals. Signals should be unambiguous, consistent, and instantly recognizable.

Table 6. Design principles related to Stage 2 SA.

(1)	Actively design the system to prevent misinterpretation of signals. Signals should be unambiguous, consistent, and instantly recognizable.
(2)	Consider how the actual tasks will be done by the operators. If operators will be expected to multi-task, then build in features to accommodate this

As an example of principle 6.1, the interface that provides the WDS status information may have a variety of information presented on it using textual and visual signals. Icons can help reduce text or provide a more grid-like design but should only be used if the operator

understands the meaning (so make sure that the operator understands the meaning through culture, training, pop-up names, or other means).

Similarly, familiar symbols should have familiar effects. Using an 'X', particularly a red 'X', should close or exit something. Red and green follow cultural norms of stop/exit/bad and go/continue/good respectively. The Apple Design Guidelines[3] give an example set of such guidelines.

The second principle, 6.2, is to consider how the actual tasks will be done by the operators. Interruptions are a major source of error. If operators will be expected to multi-task, then build in features to accommodate this and include handling an interruption as a task in your task analysis. These features can include the ability to postpone the next task so that a task can be completed, or to remember the state of the suspended task until it can be returned to. In a control room, one solution could be to simply include a pad of paper (Trafton, Altmann, Brock, & Mintz, 2003).

As an example, operators may have to multitask while monitoring the WDS. The WDS status interface provides many different pieces of information to an operator, but operators will typically not have any issues responding to routine events. However, once they need to respond to some situation, they must split their attention between the normal monitoring and the new task. This could lead to the operator missing an important warning.

The system could support this task requirement and reduce risk by providing a simplistic view of critical information during times when the operator may be splitting attention across multiple tasks. When an operator pulls up a subsystem view alongside an overall status view, the overall status could have its detail reduced while offering a more salient signal for any changes that occur, or, simple ways to keep track of state like a pad of paper or a sticky note on the screen, that could allow the operator to save partial state before dealing with an interruption.

Further information on how cognition is used to comprehend a situation is available in Endsley's work (Endsley, Bolstad, Jones, & Riley, 2003; Endsley, Bolte, et al., 2003) and other books on human-computer interaction (Krug, 2005; Ritter et al., 2014).

2.2.3 Stage 3 – Projection

The third stage of SA is achieved through projecting the model of the situation into possible outcomes. For example, an air traffic controller could anticipate a dangerous situation based on how two aircraft are likely to maneuver while changing course and act to avert the future situation. Though difficult, this type of expertise is essential for high performance in some complex tasks (Endsley, 2000).

Stage 3 failures account for about 3% of errors in aviation, but the complexity of Stage 3 SA makes generalizable causes of error difficult to isolate. General causes may include over-taxation of mental resources, insufficient knowledge of the domain, or over-projecting current trends (Jones & Endsley, 1996). This type of expertise is difficult to plan around for the engineers during early stages, thus will be given less focus in this document. Obviously, systems that help predict the future of object or systems would help operators. Support in this area could include trend or spark lines showing system state (Tufte, 2006).

One of the most effective ways to design for Stage 3 SA is by eliminating barriers preventing Stages 1 and 2 SA from being effectively supported. Thus, designers are advised to focus on solving issues with perception and comprehension before specifically addressing methods for improving an operator's ability to project into future states. However, further information about

[3] https://developer.apple.com/design/human-interface-guidelines/

supporting projection can be find in Endsley's work (Endsley, Bolstad, et al., 2003; Endsley, Bolte, et al., 2003), and work on mental models (Besnard, Greathead, & Baxter, 2004; Kieras & Bovair, 1984; Moray, 1996; Ritter et al., 2014).

2.3 Summary: Cognitive mechanisms for situation awareness

The three stages of SA provide a broad classification for the performance of operators during complex tasks. This section only briefly describes SA. This overview gives engineers the tools needed to consider how SA applies to the systems they design. Next, the cognitive mechanisms that drive operator performance are described and connected to SA.

This section briefly covers significant cognitive mechanisms used in SA as a way to describe and summarize them. These mechanisms and their role in SA get more comprehensive coverage in Chapter 3. We explain there here because these mechanisms can be simulated in a computer (Anderson, 2007), but can also be productively simulated in the designer's head to make predictions about how that part of the system will be used by the operator will perform a task. Figure 6 shows these mechanisms as they are implemented in the ACT-R cognitive architecture (Ritter et al., 2014, Ch 1). These components can be seen as distinct subsystems with semi-independent operations.

As shown in Figure 6, the process of situation awareness will often start with Perception, the intake and processing of competing sensory cues (or signals) into usable information. In this approach, perception does not necessarily lead to detection of a signal or to understanding. The perceptual process requires attention from cognition. Cognition, the central process, directs processing or focus on the task relevant information while ignoring or not processing the rest. Attention is a limited resource that must be distributed across appropriate features and is probably best seen as a process rather than a single buffer.

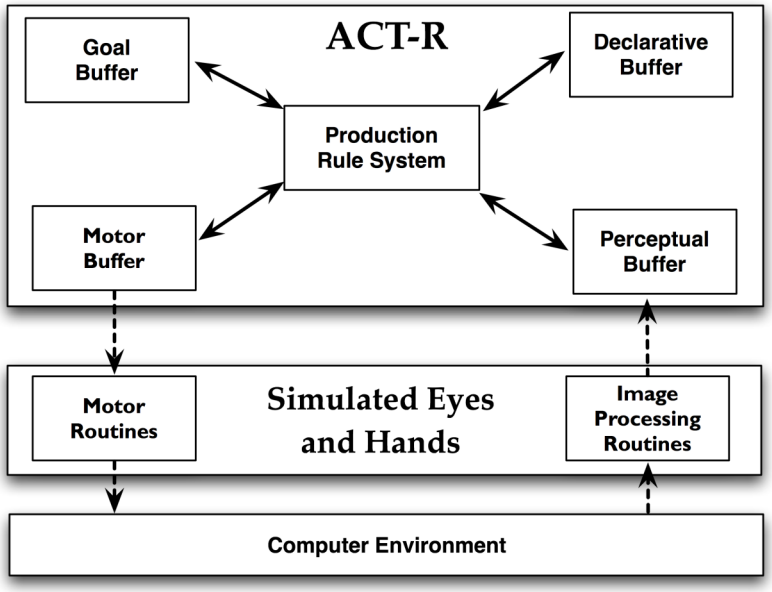

Figure 6. A schematic of the components of a computational model (ACT-R) of the human operator (Taken from Ritter et al., 2014).

Top-Down attention is goal-directed towards some feature(s) for the goal while avoiding focus on distracters (e.g., monitoring speed and position but ignoring billboards while driving).

Bottom-up attention is driven by any of the common features that indicate activity (bright colors/lights, motion, and others).

Memory is used to perform the task, recruited from the declarative memory buffer or activated from long term memory (here, in the declarative buffer and the goal buffer), which might be called Working Memory (WM), which operates as the "RAM" for cognition by storing and manipulating information chunks for short periods. This stored information has to be maintained through use, manipulated, and stored in long-term memory, or it is lost. Human memory is more similar to old drum or plated wire memory, which needed to be continually refreshed, than it is to current solid-state RAM, which can sit without use and without decay.

Directed attention captures information to be stored in WM. Dual-tasks can be performed well if each uses only/primarily one WM type or store. For example, remembering a set of numbers is easier to do while looking at a scene than while solving math problems.

The operator's mental model is the operator's internal representation of an external situation. Their mental model provides the framework that they use handle information for decision making. This model is stored in memory, which means it can be learned, or partially forgotten, and might not match the designer's representation used to understand the system and to create the interface.

The operator's mental model of a situation provides the tools needed to handle large amounts of information. They use their experience from long-term memory to scaffold the intake of new information, noting what to pay attention to, what to discard, and what to remember for a given situation. Mental models also include what to do in a situation.

Thus, situation awareness, the awareness of the state of the world, and what is happening and what will happen, is only possible through an operator's mental model and its use by a set of mechanisms similar to what is in Figure 6.

This approach, when applied to op center design, suggests that each stage of the operator's processing and response is important for successful system operation. The operator needs to be able to see and process the stimuli. They need to be able to have attention and time to understand it, and knowledge that the stimuli is important. They need to have an appropriate mental model to place the perception into relationship, the mental model, with previous perceptions as well as current goals. They need to know what to do, and how to respond.

Situation awareness thus provides a way to organize a designer's model of the operator. It makes strong suggestions about design when combined with knowing the operator's capabilities, their tasks and task priorities, and their mental model of the world, both the longer term and semantic model as well as the ongoing and evolving model of what is happening at any point in time.

The next chapter explains these components in more detail to help a designer understand how an operator might run and apply their mental model.

3. Cognition and Operator Performance

This chapter explains in more detail the primary cognitive mechanisms used by operators to perform their tasks. This section should help designers have a better mental model of operators. These details should help a designer understand how an operator does their tasks and thus support the operator better.

In this approach, based on the cognitive architecture shown in Figure 5, cognition can be described as an emergent phenomenon arising from a collection of mechanisms. The mechanisms can be seen as components of an information processing system in the same way that a computer has components.

The component mechanisms can be described in isolation (visual processing of an object) with some degree of useful truth. However, it is important to understand that this is a practical consideration. In truth, cognition relies on an extremely complex, highly interconnected neurological system.

This chapter explains these mechanisms in detail to help a designer. The mechanisms discussed here include visual perception, attention (which is perhaps emergent from other systems interaction and work), memory, and briefly learning. In each section, we note further design principles to summarize the results to aid design.

3.1 Perception

The most basic level of cognition for operators is the perception of stimuli. While we may be able to receive signals from a variety of sources, visual stimuli provide the proportional supermajority of signals. Auditory comes in second, followed in a distant third by tactile (which does not appear to be used nor needed currently in most control rooms). We will follow this natural system order in our analysis. Thus, we will primarily focus our discussion on visual perception.

3.1.1 Visual processing

Understanding the nuances of visual processing enables system designers to build their interface around the natural capabilities and limitations of the operators. At a basic level, visual processing is the process of capturing light on some visual sensor and transmitting this information to the processing system. For many robotic systems, this is a relatively straightforward process where information only flows one direction. In contrast, human processing is a bi-directional process including feature detection, goal-directed attention, pre-attentive assessment of stimuli, and active interpretation of the signals. The complex system allows us to make a sensible, coherent world out of small snapshots of information without the need for detailed processing. While humans may excel at particular tasks like pattern detection, we also can be easily tricked by unconscious misapplication of the heuristics (visual illusions, misrecognition, not seeing target objects). While some sources of errorful behavior can be inhibited or corrected through conscious effort, others are essentially reflexive actions without any reasonable method for self-regulation.

A classic example of our failure to inhibit automatic processing is the Stroop Task (Stroop, 1935). The task is simple. A subject is presented with a color word (i.e., red, blue, yellow) written in one of those same colors. The task is to name the color of the ink. The experiment has two conditions, congruous and incongruous. When congruous, the ink color and word will match (i.e. “red” written in red ink). When incongruous, the ink color and word will not match (i.e.

"red" written in yellow ink). This task seems simple in the congruous condition, but when the incongruous condition is tested, and the word and its color differ, the subject will typically stumble through responses, be significantly slower, and make many more mistakes—once we learn how to read, we simply cannot inhibit the natural response to read text. The mechanistic explanation is that the reading skill is practiced so much more than the naming skill, thus the reading skill has to be suppressed to name the color. During the Stroop Task, reading, a normally beneficial process, is a detriment to our performance.

A more comprehensive overview of low-level visual processing as well as additional resources can be found in the chapter "Behavior: Basic Psychology of the User" (Ritter et al., 2014, Ch. 4).

3.1.2 Color blindness

Color blindness is a particularly salient concern for designers due to its prevalence among the population. For the Western population, about 7% of men and 0.5% of women have some form of red-green color blindness. This causes affected individuals to have difficulty differentiating red from green. Individuals may also have blue-yellow color blindness, or even total color blindness but these are significantly more rare than red-green color blindness (Ritter et al., 2014).

There are many different forms of color blindness based on the specific deficiency in the visual system, but the general design recommendations that alleviate their effects are the same. Good design will avoid using only color as a signal for an operator. Instead, the design should incorporate multiple signals into a cohesive message for the operator. For example, an important alarm could flash bolded text information, have red coloring, and use textual indicators like exclamation marks to ensure that the message is clear.

Thus, better designs will dual-code results. That is, meaning will not just be encoded by color, but color and font, or line thickness and name, or line type and texture. Dual-coding stimuli makes them faster to be recognized and discriminated (Garner, 1974). It may be useful to check designs against this deficit. There are tools online to show how color-blind individuals perceive images and interfaces[4]. They typically take a URL or image file and show how color-blind individuals would see it. Given the prominence of color-blindness among the general population (8% of men and 1% of women; National Eye Institute), dual-coding signals and ensuring color-blind compliance would be well-advised for any system that requires human operators.

3.1.3 Visual search

The visual system can be broadly broken up into two subsystems based on their role. The eye handles stimulus detection and the brain (in specialized regions) handles stimulus-interpretation. Stimulus detection occurs within the eye, but the process itself is driven by a combination of goal-directed attention from the mind (top-down) and automatic processing of salient features (bottom-up). Top-down and bottom-up directives guide the visual processing and integration of the environment that occur during visual search.

Visual search of the information displayed on an interface is a core activity for operators, regardless of the task. As their attention is oriented to the task at hand, the operator will need to comprehend the information presented on any given interface. Visual processing is an intermittent process in which our eyes are constantly alternating between saccades (rapid eye movements to some feature) and fixations (resting moments of information intake). What we perceive as a continuous experience is actually an intermittent series of fixations that are

[4] https://www.toptal.com/designers/colorfilter/

unconsciously aggregated into a coherent, though not necessarily accurate, mental model of our surroundings (Irwin, Brown, & Sun, 1988). During fixations, the feature-detection relies on distinguishing target features from distracter features through pre-attentive visual processing (Healey & Enns, 2012). This summary of vision as being active can be contrasted with folk psychology and early understanding of vision where humans were understood to see and understand the whole display at once, whereas, we now know that the eye must search for information actively on the display and often refresh it (Findlay & Gilchrist, 2003).

During complex tasks that require visual search, both bottom-up feature recognition and top-down goal-oriented activity influence the performance of the operator at finding that information. While top-down directives lead visual search towards a certain set of features, our eyes are unable to fully inhibit the bottom-up feature detection. Given the effects that distracting features can present for operators, designers should understand what types of visual features draw people's attention and the role of higher-level graphical organization.

3.1.4 Pre-attentive visual processing

Once an operator perceives the signals presented by an interface, the visual processing system immediately begins working to form a coherent mental model of the scene. Cognitive limitations on information processing prevent humans from scanning, processing, and understanding every individual signal within the visual field. Instead, we have developed a complex pattern-matching system that reduces workload without (usually) negatively impacting comprehension.

There are two main processes that occur during the early stages of visual search. The first is pre-attentive visual processing based on relatively simple features of the objects. Figure 7 shows examples of the types of features that are easily and immediately detected during visual search. The common element across these examples is the contrast between features. When objects vary in orientation, length, or size (compared to other objects their environment), they are identified and distinguished much more quickly than other objects. Easily distinguished visual features are more salient to the operator, particularly when the operator is distracted or overworked.

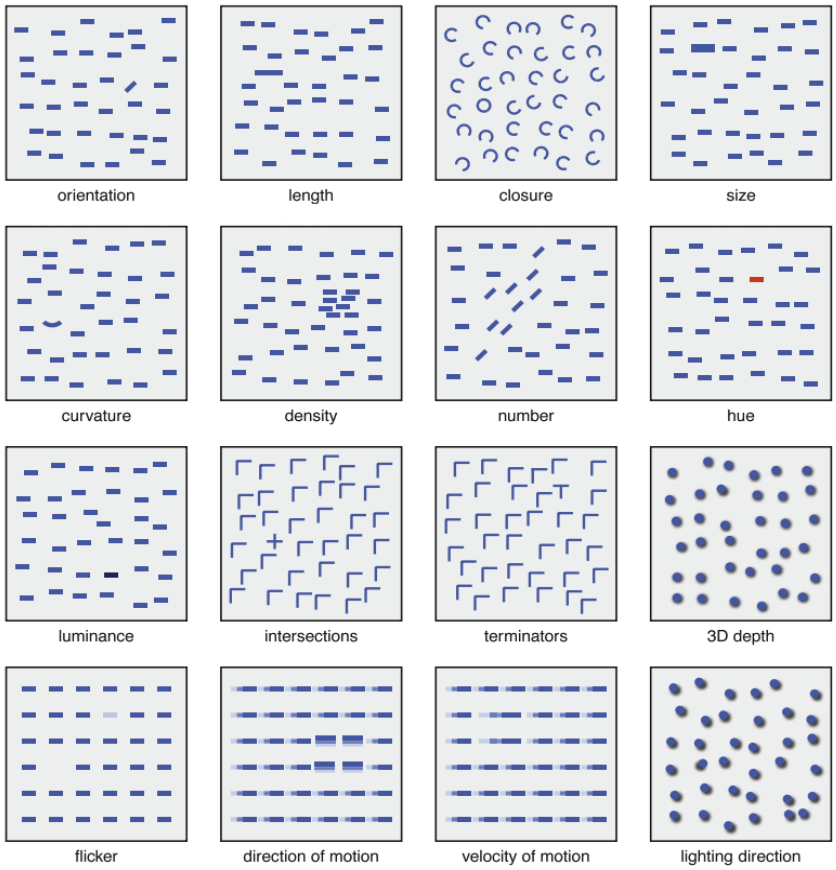

Figure 7. Examples of pre-attentive visual features
(Figure and caption reprinted from Healey & Enns, 2012).

The contrasting features shown in Figure 7 vary in their salience. Just by glancing across the examples, we can notice a difference in how rapidly we acquire the target stimulus among the distracters. The image for hue is easily discerned, while lighting direction and 3D depth are more difficult. Designers must consider the salience of the signals they will present to the operator and provide the most salient cues to the most important differences.

The second major process of early visual processing is the grouping of individual features into shared, higher order visual structures. This is known as Gestalt grouping or Gestalt Theory (Chang, Dooley, & Tuovinen, 2002; Moore & Egeth, 1997). Just as particular features are distinguished individually, sets of features are organized into visual structures to be further processed by the viewer. This allows the viewer of the scene to organize the information-dense world into a coherent set of distinct objects. Just like the processing of pre-attentive visual features, Gestalt grouping is an involuntary processing step that shapes how a person perceives the world around them (Moore & Egeth, 1997).

Gestalt Theory encompasses a family of related psychological principles of perceptual organization used to describe common instances of visual integration. The literature on this subject is varied and as such, the specific principles can often be described in multiple ways depending on the situation or researcher. Though not exhaustive, Figure 8 shows nine of the most common examples of Gestalt principles affecting how we aggregate component pieces of a visual image. These principles can be used by a designer to group information together or separate different subgroups appropriately.

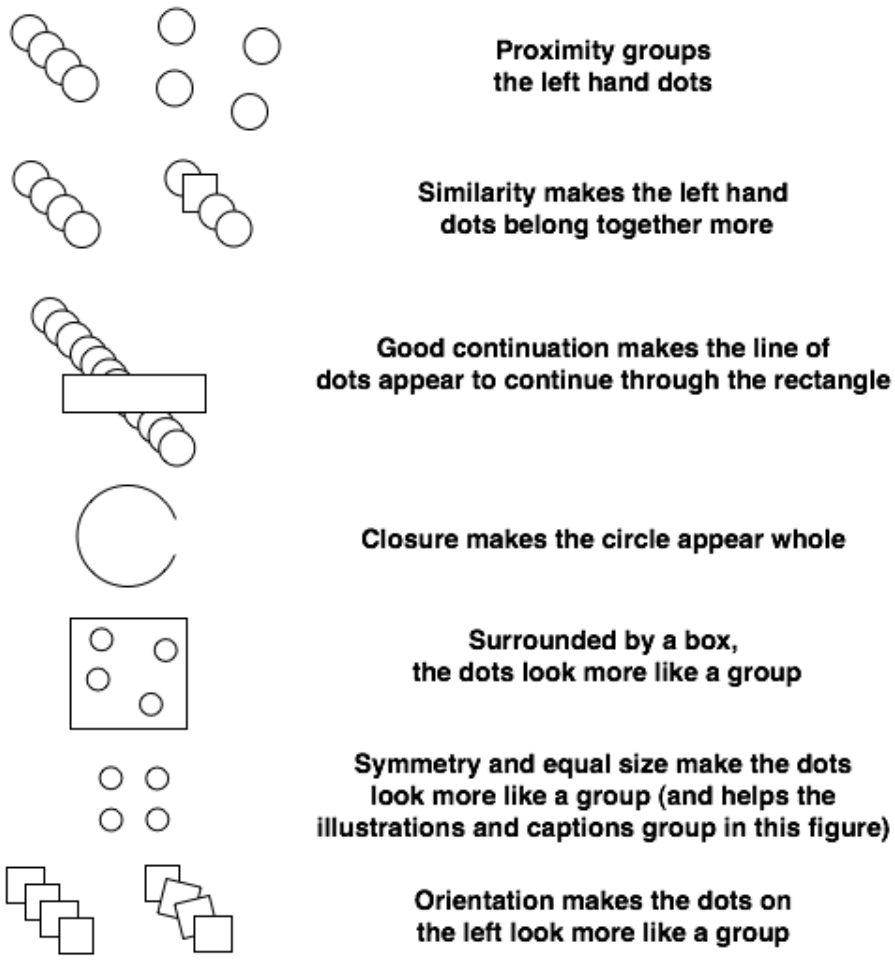

Figure 8. Common examples of Gestalt principles affecting image perception. (Copied from Ritter et al., 2014, Figure 4-15).

Even without other factors affecting visual processing, Gestalt Theory can serve as a useful framework for analyzing and improving the design of an interface. Chang and colleagues (2002) demonstrates how Gestalt Theory can be used to guide the redesign of an electronic learning tool. During their background research, the authors identified a subset of the many Gestalt "laws" from prior research and used these as the basis for their redesign process. The redesign process described by Chang and colleagues provides a useful exemplar of the methodology, however they did not collect the empirical data necessary to provide a detailed analysis of how their redesign affected interface performance.

3.1.5 Summary of visual perception and principles

Nearly everything on the interface is a signal or feature. Designers should assess the importance of each signal as well as the salience associated with it.

To make signals be recognized, change the hue, make it flash, or increase its size, or use pre-attentive visual features shown in Figure 6 to modify the salience of the information. The inverse is also true. For irrelevant features at a given point, ensure their salience is appropriate by modifying their visual representation.

If an operator does not perceive something, they will not know that they missed it. Creating our mental model requires unconscious assumptions about the world. Do not assume that the

operator will realize that they need to attend to a minor signal or remember to look at something; help them.

It may be appropriate to test the interface for color-blindness compatibility. Where colors cannot be changed, one could test the users to support reconsidering changing colors, or to find other ways to support color-blind users.

Gestalt principles give engineers the ability to predict how operators will perceive their interface and its functionality. Designing the system layout around these principles can ensure that the engineer's intentions are clearly conveyed to the operator.

To summarize how to use results from visual perception in design, we present a few design principles related to vision.

Principle 3.1: Designing to accommodate color blindness will solve multiple problems at once.

Color blindness is prevalent among the general population at between 2-7%. Presenting information with multiple signals and modes can help ensure the message is clearly receiving regardless of their color perception and faster overall.

Principle 3.2: Colors must be sparingly used, consistent, and reserved for critical information.

Color can be recognized and interpreted much more quickly than a complex signal, but overuse reduces the effectiveness. If possible, follow these rules: use no more than 4 different colors, adopt a dull screen as background, and reserve specific colors for specific signals.

Thus, ensure that color provides a valuable signal to the operator through purposeful use of specific colors to emphasize critical information on an otherwise dull interface. Often, color can be a distracter just as easily as a signal if the interface overused or misused. Three specific examples are shown in Figures 9, 10, and 11.

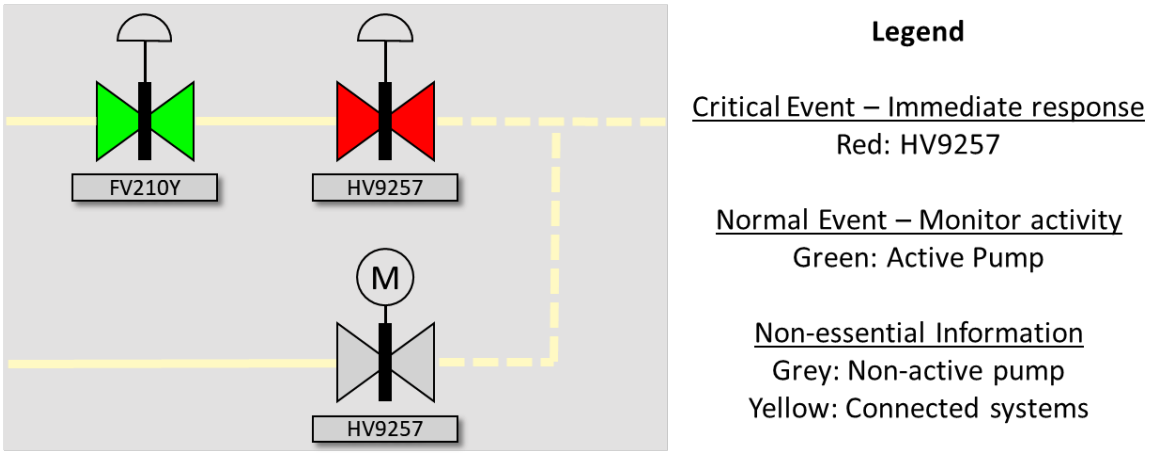

Figure 9. Labeled example of interface with dull color overall, allowing the green "active pump" signal to stand out. Figure redrawn by authors and modified from Ulrich and Boring (2013).

Designers must consider how each color used in the system will be interpreted by operators. Figure 8 shows a relatively dull interface that can be quickly scanned to identify which system processes are active without any distracting signals. Connecting lines between components

(yellow) are easily distinguished, but the reduced saturation demotes their importance during typical use.

Figure 9 shows how to use of color within an interface should be considered as a scarce resource. On a completely plain background, one color can be extremely visible, but each new color and new use reduces the salience of that signal. Figure 10 shows an example of reducing the color usage within an interface to highlight critical information (Ulrich & Boring, 2013).

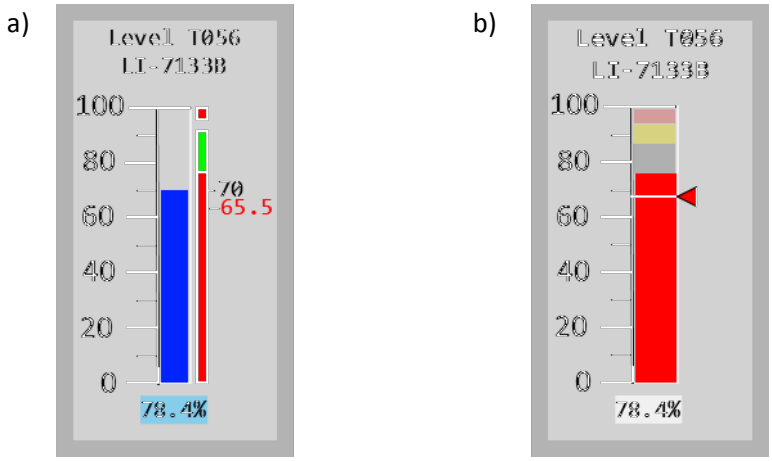

Figure 10. Examples of muted interface with dulled colors, dedicated alarm colors, and merged information for easier perception. Images (a) and (b) show an initial and revised pressure gauge. Figure redrawn by authors and modified from Ulrich and Boring's guidelines ; 2013)

Color is often a major factor used with an interface to code signals with meaning. Color use will usually use pairs or sets of colors to provide a categorical piece of information for the operator. Green, yellow, and red, can indicate the system status on a range from healthy to critical failure. Blue can represent active pumps for a liquid while grey shows inactive. Color is a valuable signaling method for typical operators, but designers should ensure that their design has multiple signals indicating critical information.

For example, the gauges shown in Figure 10a may be unable to provide color blind operators with enough information to ensure system success. Figure 10b shows a revised interface that would be better suited for all users. Though the second gauge sacrifices some contrast between the safe and dangerous system states, the thick black line and arrow indicating the critical level eliminates color blindness as a risk for operator failure.

Principle 3.3: Make text with readable fonts, no more than 3 font types, and of proper sizes, with simple, short text strings

Reading from screens tends to be slower and more difficult than print-based reading. This may be due to the difference between projective and reflective light or due to pixel density. Many operators will not be trained to differentiate font types, so use different fonts sparingly. Improve readability and comprehension by using readable, simple fonts. Ensure font size is appropriate for the expected viewing distance. Concise text, accompanied by a symbol or icon, will be faster than a description and more easily interpreted than an icon alone.

Designers should thus avoid using unnecessarily "fancy" fonts and settle on simple, effective presentation of the key information. In general, long strings of text should be avoided. They can

be replaced with symbols, bullet points, or at the very least, augmented with emphasized words to make scanning easier. Figure 11 shows an example of improvement.

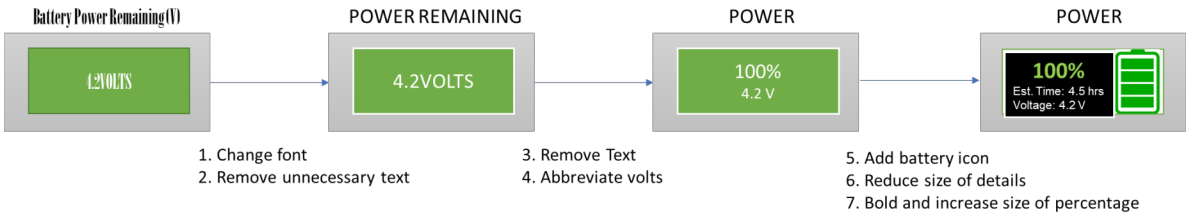

Figure 11. Incremental improvement of power level indicator. Final product can be quickly referenced for general status and examined more closely for detailed information like voltage and time remaining.

In general, reading from digital screens is slower and leads to less comprehension compared to print-based reading. Researchers have studied the effects of screen-based reading quite extensively. They consistently find that reading from screens is slower by 10-30%, leads to increased errors, and fatigues the user more quickly than print reading (Ritter et al., 2014).

Principle 3.4: Ensure signals indicating missing information are clear and obvious.

Operators rely on gathering and interpreting information to make key decisions. Uncertain or missing information can affect performance through incorrect assumptions by operators.

Missing information from a sensor or system can be a signal to the operator about the situation, but this is only possible if the operator is aware that the information is missing. When operators do not realize that some information is missing, they may rely on their base assumption of normal operating conditions, which can lead to potential disaster.

For example, a pilot operating a plane in cloud cover with malfunctioning terrain sensors can respond differently if aware of the missing information. If aware of the issue, they could climb to a safe altitude regardless of any "true" obstacle. If unaware, they may crash after assuming they were on a safe trajectory.

In the WDS system, signals indicating success for a repeating procedure could be represented as a simple binary response: success or failure (1a and 3 from Figure 12). If the update schedule is known to vary by 30 minutes, this could lead to many false alarms if a missing self-test at the exact due time qualifies as a critical failure. The design in Figure 12 allows operators to quickly see when the last test occurred and provides an intermediate signal for a missing self-test. This gives operators a signal to be in a "ready" state to respond to a critical failure.

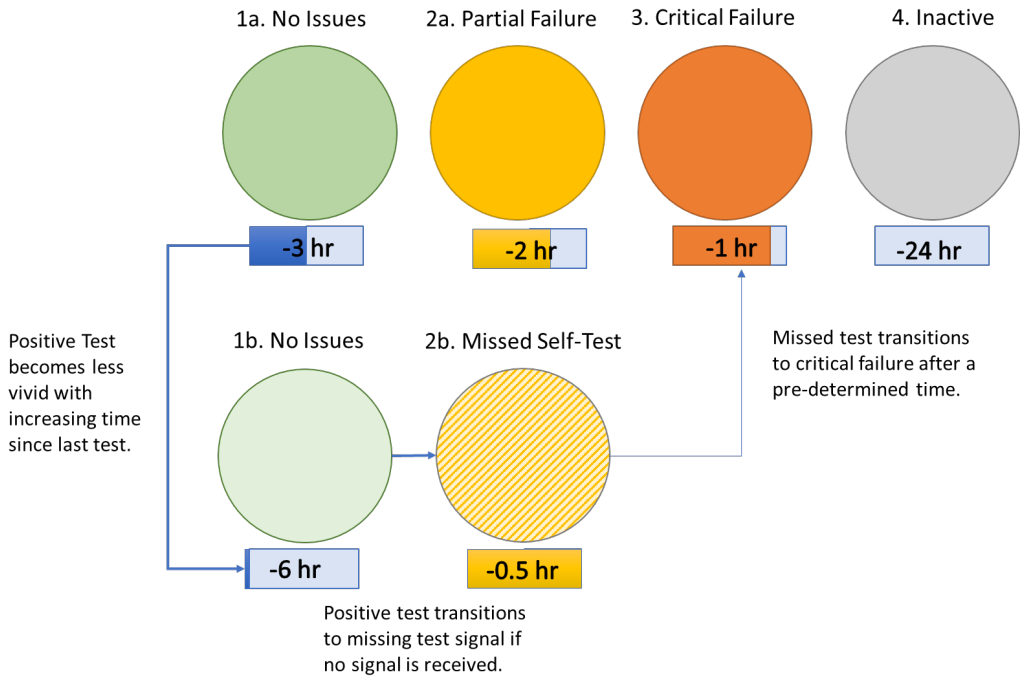

Figure 12. The role of color to represent missing and aging information.

Principle 3.5: Arrangement of screen components should be useful, consistent, and close.

Whether designing the full system interface with multiple objects or creating the objects themselves, limit the amount of distance between signals that are commonly used together. Basically, have a theory of how the interface will be used, and use the task analysis and operator knowledge and characteristics to design the interface so that knowledge information on the same task are near each other.

As an operator scans the system interface during typical monitoring tasks, they will be generally searching for alarms, alerts, or any sign indicating a potentially risky situation. The task analysis should provide a summary of the tasks, their importance, and their frequency. Checking systems with distant components (as measured as travel through the interface) requires more time and effort to perform well. Additionally, upon identifying an alarm, operators often will search for signals that confirm the veracity of the alarm. Grouping related components together makes this easier, reduces strain, and increases their ability to search for information.

Grouping and arrangement should also attempt to follow consistent patterns both visually and semantically across multiple displays. The design guidelines in Appendix 3 provide guidance about the semantics and importance.

3.2 Attention

While visual perception can be described as the integration of information through the field of vision, attention is the "spotlight" that makes a set of stimuli more active or relevant than the rest through state or use. As operators are presented with a constant array of information, an executive control system in the mind is directing attention towards features or items in that set of information. A crucial feature of attention is the enhanced acuity for the target of interest at the expense of awareness of periphery stimuli (Ritter et al., 2014). The shift in focus can occur due

to the salience of certain features, perceived relevance to a particular goal, or an active process of cognitive control.

This section will first discuss the basics of the underlying mechanisms of attention and the effects of task switching. Next, we will describe the causes and implications of limited attentional resources and the attrition of attention.

Attention plays a crucial role in visual perception by providing a mechanism for isolating specific features of interest. Visual perception entails making sense of a world with too much information present; attention is the tool for "working around" this natural limitation. Attention provides guidance (though not total control) for the sequence of eye saccades and fixations during goal-directed search for visual features. The interaction between these two mechanisms is moderated by cognitive control (e.g., goal-directed behavior) and aspects of features in the visual field (e.g., salience). The interaction between these two forces can affect performance by altering the usage of "cognitive resources" during a particular task. For example, inhibiting a response to look at a flashing light requires active control of visual search, and thus attention. The skill at which a user can inhibit these responses is governed, at least in part, by their working memory capacity (Unsworth, Schrock, & Engle, 2004). The inverse is true as well: an extremely salient signal will require fewer cognitive resources to detect.

3.2.1 Attentional vigilance

The role attention plays in cognitive tasks cannot be overstated. While we have primarily been describing the role of attention on visual processes, attention plays a central role in both internal (e.g. problem-solving, goal-sustenance) and external cognitive mechanisms (visual search). The act of maintaining attention on a task is called attentional vigilance, or just vigilance. Tasks that require vigilance are characterized by the need to maintain attention over an extended period while attempting to detect target stimuli without responding to neutral or distracting stimuli. Performance loss is often ascribed to the vigilance decrement, or the performance decline that occurs over a period of active monitoring. This type of task is extremely common of operators within an op center.

Sustained attention on a task is impaired by several factors. First, the salience of the goal signals directly affects the decay rate of operator performance due to the vigilance decrement (Helton & Warm, 2008). Increased working memory load leads to worse performance on vigilance tasks. If an operator needs to remember other tasks or keep unnecessary information in working memory, they will have a higher cognitive load (Helton & Russell, 2011). Depending on the type of information being remembered, the impact on performance may be reduced. For example, listening to a supervisor speak (verbal) and watching a graphical display (visual) will be easier than trying to listen and read (both verbal) simultaneously (Epling, Russell, & Helton, 2016).

3.2.2 Resuming attention: Interruptions and task-switching

Interruptions provide a major risk in disrupting the ability of operators to maintain their attention on a given task. Unanticipated breaks during the completion of a task have been shown to increase subjective workload and error rates, even for experienced professionals (*e.g.,* Campoe & Giuliano, 2017; DeMarco & Lister, 1999). Designers should be aware of how interruptions, even when planned, can impair performance of operators.

The overall framework for understanding task interruption can be divided into several phases. First, the worker will be completing some primary task. At some point prior to completing the

primary task, the worker is exposed to a distraction signaling the need to complete a secondary task. The time between receiving the signal and initiating the secondary task is called the interruption lag. Next, the worker begins the secondary task. The time to complete the secondary task is called the interruption length. Upon concluding the secondary task, a period called the resumption lag occurs until the worker is able to resume the primary task (Trafton et al., 2013). This process can occur multiple times throughout the completion of a primary task.

The distractions force the operator to lose their attention on one task, begin attending to a different task, then transition back into attending to the original task. Each time the operator transfers their focus (in both directions), there will be a necessary "activation period" where the operator is working through the stages of situational awareness: perceiving the task features, forming a mental model of the situation, and finally extending their mental model into likely future scenarios to guide action. This process takes time and leads to performance impairment. It is also a source of errors. Well-designed systems should attempt to alleviate the risks associated with interruptions to primary tasks.

The designer will primarily have control over the design of the associated tasks. While a designer may be able to influence the training of the operator, it is more practical to design the system and task around a range of skill levels if possible. The first method for reducing the effects of interruptions on performance is simply removing them from the possible task structure. Even among experienced professionals working in high-stakes situations, the number of interruptions is directly correlated with an increased error rate, cognitive workload, and stress level (Campoe & Giuliano, 2017).

If interruptions cannot be limited, there are several ways to alleviate the performance impairment. First, designers can provide a preliminary warning signal that indicates an interruption is imminent (within the next 10 seconds). This allows operators to begin the preparing to switch tasks without the need to fully place their focus on the new task. Trafton and colleagues (2003) informally describes the process that occurs after the warning signal as the operator answering two questions and storing the response in memory: "Now what was I doing?" and "Now what am I about to do?". The answer to the first question helps the operator identify the point at which to resume the primary task, thus reducing the resumption lag. The answer to the second question prompts the user to gradually begin attending to the interruption task, thus reducing the interruption lag. The same study demonstrated that providing a warning signal with 10 seconds notice for a distraction reduced the resumption lag by nearly 50% (8 seconds without warning vs. 4 seconds with a warning) for an unpracticed task. While this effect diminished with repeated practice, this design guideline is particularly useful for infrequent tasks that may be minimally practiced.

Besides offering a warning, designers can design interruptions that minimize the performance impairment. First, interruption length is a large predictor of the resumption lag. Working memory plays a significant role in managing attention. Long interruptions impair the ability to rehearse the previous task state, which may lead to an operator forgetting their place in the task. Designers can account for this by reducing the length of interruptions and preventing interruptions during high-stakes tasks (Campoe & Giuliano, 2017). Interruptions that force the operator to change contexts also impair performance. Context-change is a broad descriptor that may include changing locations, unexpected transitions from visual processing to verbal processing (e.g. talking to a coworker) or generally unexpected shifts in cognitive requirements (Marsh, Cook, & Hicks, 2006). So, when possible, allow the operator to finish their current

primary task step. This reduces the resumption lag for computer-based work, though this benefit disappears for manual work (Campoe & Giuliano, 2017).

3.2.3 Signal thresholds and habituation

Visual input has natural limitations on the strength of the stimuli that can be detected. The threshold that separates undetectable and detectable stimuli is called a detection threshold. For the human eye, this is approximately 100 quanta. This somewhat abstract denotation can be better understood through an example: we can detect a candle flame from 50 km on a clear dark night (Galanter, 1962). The amount of change necessary to detect differences in a stimulus, such as differences in color or brightness, is called a just noticeable difference (JND). Interfaces that attempt to show differences that are not one JND apart (imagine a graduated color chart) physiologically cannot be recognized.

While human vision can be very sensitive during initial presentation of a stimuli, there is also a natural process of habituation that occurs during persistent detection of certain stimuli. As an operator becomes accustomed to a predictable, persistent visual stimulus, they lose the ability to perceive it without conscious effort, it becomes background to them. For example, people living next to train tracks stop hearing the trains. Though it is more common with simple stimuli, habituation can also occur with complex stimuli that require action (e.g., clicking a "confirm action" box for every action; Ritter et al., 2014).

System designers already will be taking some steps towards accounting for these low-level issues during the design process. For example, system designers will often use particular visual characteristics such as flickering or flashing lights, changes in color, or motion to indicate that an operator's attention is needed. However, designers should use caution when deciding when to use alerting signals. When a system is working as intended, the designer should be aiming for signals that facilitate habituation, that is, the changes appear normal and do not call attention to themselves. However, once the system detects an alert of some kind, the design principles become inverted. Rather than facilitating habituation, designers should actively prevent habituation.

3.2.4 Speed-accuracy tradeoff (How to design for acceptable errors)

There is a constant in human behavior represented by Figure 13. This graph shows that behavior can be slow and careful with low errors, or rather fast and with higher errors. Operators will vary in what their curve looks like. Similar operators may be at different points on the same curve as well. To avoid the extremes, psychology studies often say 'to work as quickly and accurately as possible" to attempt to put subjects at some ideal center point.

We note this tradeoff to designers so that when they are observing users, they realize that operators may be working at different points in the curve. For example, when typing drafts, we type fast and use spell correction to clean up. When entering passwords, we type slow because errors particularly take time to correct.

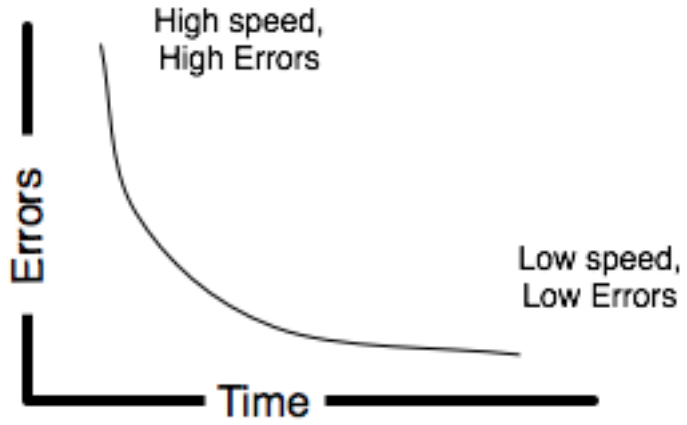

Figure 13. The speed accuracy trade-off curve. Reprinted from Ritter et al., 2014

3.2.5 Summary of attention

Attention can be seen as the tasks and information that the operator is attending to or working with. There are consistencies and effects that arise from this process. To the extent that designers can understand the operator and their tasks, they have a role to facilitate the allocation of attention and to support its use.

To summarize how designers can support operators' attention, we present a few design principles related to attention.

Principle 3.6: Present information needed for comprehension directly

Attention and working memory are limited; information shown to the operator should be processed and integrated as much as possible (but not more) to reduce operator workload and support the system goals.

Avoid giving operator extra work, particularly for tasks that better suited for technology. Methods for implementing this can range in complexity, but beneficial design choices will be structured around eliminating extraneous work for the operator. Simple examples might include reducing unnecessary mental math or just moving related information closer together (eye movements take time, as do mouse movements, milliseconds matter; (Gray & Boehm-Davis, 2000) Complex examples include totally redesigning a complicated display around a relatable design metaphor with a unified representation of the information (Figures 14 & 15).

For example, consider a simple altimeter design. Pilots are often skilled operators with a lot of experience in their primary tasks. However, the human limits on attention and memory are always a factor. Designing to improve comprehension will reduce mental strain for experienced and inexperienced pilots alike.

A pilot need not calculate the difference between assigned altitude and present altitude. Technology has advanced so that this can be calculated and displayed better than the initial dials. Simplify the task and use each systems' strengths. The computer can handle simple mathematical calculations and could show the values using two lines separated by the deviation. The pilot can then visually identify any issues with altitude much more quickly as a visual process.

Compare the two altimeters in Figure 14. On the left, the pilot must do an abstract calculation to compute the difference and direction between present and assigned altitude. On the right, the difference is shown visually, a much faster and less error-prone task.

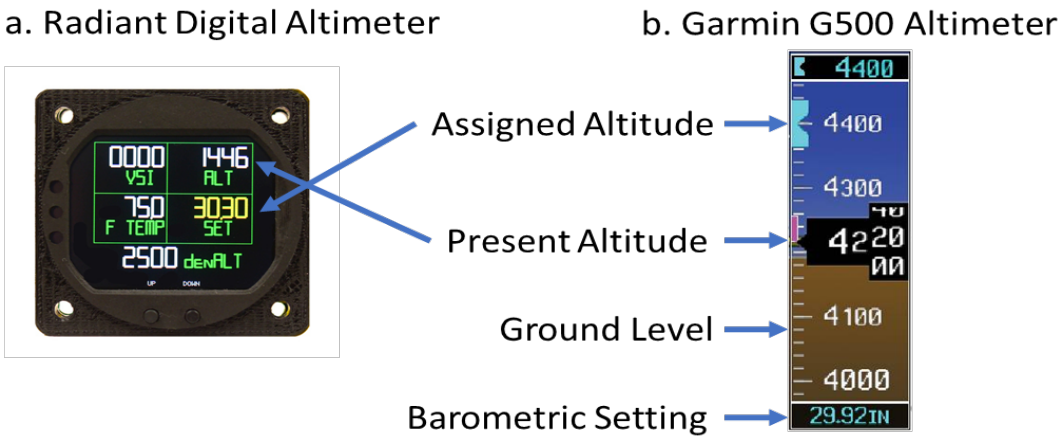

Figure 14. The interfaces for two different altimeters. The Radiant Digital Altimeter[5] (a) requires the pilot to mentally compare their altitude to the set value while accounting variables affecting the instrument accuracy. The Garmin G500[6] (b) simplifies this by including a spatial comparison between accurate barometric altitudes and clear representation of current altitude and ground level.

As another example that is more complex, consider Figure 15. It provides a redesign of an airplane's control panel around a more direct plane metaphor. Flying with traditional airplane displays requires the pilot to mentally calculate their current flight relative to the limits based on the flight envelope (i.e., stable flight based on related parameters like airspeed, altitude, and orientation). This mental calculation is difficult and cognitively taxing, particularly during times of high workload from adverse conditions like fog or turbulence. When vision is impaired, pilots rely solely on instrument flight (IF) with no visual reference frame.

This risky situation led to Temme, Still, and Acromite (2003) to propose an interface titled "Oz" that portrays the key information as an integrated display built around a digital plane (Figure 15a). This display presents exactly what the pilot needs to know for the task: current aircraft performance compared to aircraft limits and optimal values. A comparison between old and new displays are shown in Figure 15 (b and c).

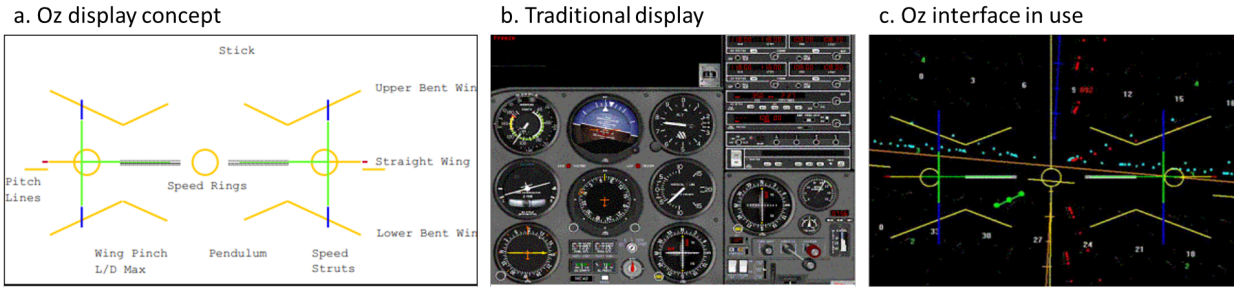

Figure 15. The design and implementation of the OZ compared to a traditional cockpit. (Images from Temme et al., 2003).

Although the OZ display initially appears complex to new users, it was designed to support common tasks based on a mental model of the plane. This was confirmed via testing when the OZ interface performed significantly better than conventional displays. With the OZ display, subjects with no flight experience immediately showed greater flight precision (for orientation

[5] a: http://www.beliteaircraftstore.com/radiant-digital-altimeter-1/
[6] b: https://www.manualslib.com/manual/1230544/Garmin-G500.html?page=53#manual

and altitude) and reduced performance loss from turbulence than when using the typical display. After about 80 hours of flight time with both displays, subjects attempted to perform a reading task while operating the plane. This was essentially impossible with the conventional display, but subjects saw almost no loss in performance when using OZ. Similar designs could be created for control rooms, perhaps as a summary supporting task performance while retaining the raw data visible behind the summary display.

Principle 3.7: Support operators to deal with interruptions.

To summarize, to support operators to deal with interruptions:

1) High-stakes work should be distraction-free.
2) Warn operators that an interruption is imminent when possible, that is, allow operators to prepare for task-switching.
3) Promote completion of primary task steps before beginning secondary tasks. Simplify the process for resuming a postponed task. This can be done by suspending the secondary task, autocompleting the primary task, or provide note-taking about the status of the primary task.
4) If interruptions are necessary, reduce the distance and difference between the primary and secondary tasks as measured semantically or syntactically.

Principle 3.8: Use stimuli habituation appropriately.

Even salient signals will become habituated with repeated presentation.

Constant presentation of a signal leads to habituation, and thus reduced detection and attention by operators. Designers should create a hierarchy of signal salience to ensure the right signals get through.

3.3 Working memory and cognition

Following the perception of information from the environment, the operator needs to use that information to make decisions and complete their work. Task-related information must be analyzed, manipulated, and transformed into useful information that can guide the actions taken by the operator. The operator must integrate their knowledge of the state of the world with their mental model of the task. For example, an operator sees that the temperature of some module is above the safe threshold and the battery is running low. The operator stores these facts in their short-term memory and then consults their long-term memory for how to respond to the issue. The response is then also added to short-term memory alongside the facts about the world state. The operator performs the response on the system, ensures the problem is fixed, and then discards the old information before moving onto their next task. Variations of this process occurs many times throughout an operator's shift. These human memories do not work as well (at least under conventional views) as computer memory, so designers familiar with computers should be aware of the differences. Designers should particularly be aware of the differences because their own mental models of their own memories are likely to be particularly incorrect—if your memory fails you are unlikely to be able to notice this!). This section will describe how working memory and long-term memory affect operator performance.

3.3.1 Working memory

Often, the work performed in op centers requires operators to integrate snippets of information from various sources to come to a decision or understand the situation. This process of storing

and manipulating that information occurs within the working memory of the operator. Working memory stores and manipulates information for near-term use (Ricker, AuBuchon, & Cowan, 2010). Some tasks require multiple pieces of information to be analyzed and processed near-simultaneously; working memory enables people to handle this by offering a "scratchpad" for relevant information. Though particularly relevant during the performance of complex tasks, working memory is a foundational mediator for how each person interacts with the world. Working memory acts as a store for both internal events (i.e., recalling long-term memories) and external events (i.e., perceiving visual signals). In many ways, working memory is often analogized to be comparable to the RAM of a computer system, whereas long-term memory is like the ROM. The RAM, or working memory, allows rapid data access, efficient manipulation, and quick turnover between processes. The ROM, or long-term memory, provides a slower, semi-permanent location for information storage and retrieval.

The RAM-ROM analogy also applies to the limitations of working memory. While long-term memory does not appear to have a clear storage limit in humans, working memory is constrained by a capacity of only a few items—the most common general storage limit is about seven items plus or minus two items (Miller, 1956). The seven-item limit is overly simplistic but provides a useful anchor for working memory capacity. Working memory capacity also varies across the population with greater working memory capacity being associated better performance at cognitive tasks (Just & Carpenter, 1992), and how abstract and how well known the concepts are (less abstract and more practiced tasks are easier to remember and use; Ritter et al., 2014, Ch. 5 for more information).

The approximate limit for working memory capacity becomes even more complex due to processes like chunking. Chunking refers to a mental process for grouping sets of individual information pieces into easily recognizable sets. For example, it will be easier to remember a sequence of items like "N S A F B I" (chunked as: NSA, FBI) than "Q G Z T Y V" (not chunkable by most; Chalmers, 2003; Ellis, 1996). Chunking mechanisms can be leveraged by system designers to increase the practical working memory capacity of the users.

Modern theories of memory suggest that working memory is built from specialized subsystems that differ based on their input: the "visuospatial sketchpad" for visual spatial information and the "phonological loop" for verbal information (Baddeley, 2000). This distinction between verbal and visual working memory stores is important because these two systems can perform semi-independently without much interference (i.e., loss of performance) between them. When implemented successfully, this can allow someone to drive a car while listening to an audiobook with almost no loss of performance for the primary task (Granados, Hopper, & He, 2018). However, implementing this concept is not necessarily foolproof. When the secondary task requires too much mental effort (i.e., maintaining a conversation vs. passive listening), driving performance tends to be degraded to a noticeable degree (Strayer, Drews, & Johnston, 2003). While multi-tasking is best avoided, making attempts to isolate the tasks to distinct working memory stores can provide some measure of risk-reduction.

For the designer, there are a few takeaway implications for design. (a) Working memory has limitations on capacity and performance. Don't use it up asking the user to remember items the system can remember for them. (b) Chunking of items can increase the functional working memory capacity. Support chunking when you can by putting items in a canonical order, spacing items to support chunking (e.g., FBI vs F__:B-I) and understanding the patterns operators know and choose. (c) Working memory has a time-based decay. Maintenance requires

rehearsal at some cost to the operator's cognitive resources. Don't expect users to remember information across minutes.

3.3.2 Cognitive load

Cognitive work is inherently taxing on our mental resources. We have previously discussed the impairment of cognition as it relates to attention, but higher-order processes are also affected. Throughout the performance of cognitive work within an op center, operators are presented with information that must be monitored and assessed and may need to be compared across time. These types of work are inherently difficult, particularly when during long periods of performing the tasks. Cognitive load theory (CLT) describes how the various factors like working memory load, personal stress, and task difficulty can provide an overall decrement on performance of cognitive work (Sweller, 1988). Cognitive load theory provides a way to compare task difficulty (relative to the expertise of the user) across different task environments. Reducing cognitive load provides a broadly effective way to improve performance by freeing up working memory capacity for more important tasks like integrating information and learning. CLT is a useful concept, but currently it lacks units and an objective way to measure it. We find it useful non-the-less.

A review of cognitive load's role in human-computer interaction design is provided by Hollender, Hofmann, Deneke, and Schmitz (2010). This review integrates CLT research into a useful framework for systems engineers. They posit three main types of cognitive load: intrinsic, extrinsic, and germane. Intrinsic cognitive load refers to the inherent complexity of the information being processed by the user. Comparing intrinsic load can only really be done by comparing two tasks rather than by providing a standalone value. For example, driving on an empty highway would likely provide less inherent complexity compared to driving on a busy city street.

Extrinsic cognitive load refers to environmental and context-dependent factors that provide unnecessary contributions to task difficulty. Integrating spatially distant information from displays that are on opposite ends of the room will be inherently more difficult than if the displays were side-by-side due to the required storage of the information in working memory between task steps.

Finally, germane cognitive load refers to the beneficial cognitive work that improves task performance. Learning and practice of the skills and schema required to perform a task also require cognitive resources, in contrast to unhelpful portions of the overall cognitive load. All three types of load contribute to the overall working memory needs of any given task, and the ideal task will reduce the intrinsic and extrinsic load to provide more resources for the beneficial mechanisms that occur from germane cognitive load.

Reducing the cognitive load of extraneous tasks can provide a consistently useful method for improving the performance of operators. A simple method for reducing cognitive load is by enforcing consistency across the layout, color scheme, and overall information presentation style for components of an individual system and across multiple systems (Chalmers, 2003). Even experienced users that may switch between a Windows OS and Mac OS will know the feeling of attempting to use a Mac-only shortcut on a Windows machine (or vice versa).

Many of the recommendations for reducing cognitive load can be succinctly described as reduce the space and distance between co-dependent information. In some cases, this can be a relatively simple process with multiple solutions. Disparate information sources could be split across multiple displays to maximize information presentation, or alternatively, a single display

could be trimmed of unnecessary information to bring the most important features onto a single, more efficient display (Brown, Greenspan, & Biddle, 2013). Other cases provide less clarity in determining the best practices for a given context. Providing redundancy in feature presentation can help reinforce certain information, but the additional features inherently increase the intrinsic cognitive load during interaction with the system (Grunwald & Corsbie-Massay, 2006). Engineers and other stakeholders must use the risk-driven approach to make informed decisions; competing design recommendations are rarely weighted on easily comparable scales. Krug's (2005) approach provides further suggestions to reduce cognitive load.

Further ways to support operators by reducing cognitive load in them is by increasing cognitive load in the system. This includes (a) Reminding operators to do tasks that need to be done. (b) Reduces cognitive load by simplify tasks that that can be simplified in actions number, length, and complexity. (c) Automate tasks that can be automated. Like your turn signal automatically shutting off when you return your tires to straight.

3.3.3 Summary of working memory and cognition

Working memory is used extensively by cognition. More is usually better, and less stress on working memory by decreasing the amount required and the time information has to be held also reduces errors. To summarize how designers can make best use of operators' working memory, we present a design principle related to working memory.

Principle 3.9: Reduce the cognitive resources used during multi-step tasks

Operators' cognitive resources like working memory and attention are limited, and these limitations are made worse by fatigue, stress, and task difficulty. Simplifying the work will reduce workload and make errors less likely to occur.

Simplifying tasks can be done in many ways depending on the specific scenario. The common factor for all successful implementations of this guideline is a reduction in the amount of working memory, attention, or any other cognitive resource needed to perform the task.

If an operator is alerted for a task that needs done in thirty minutes, the system should provide an additional reminder at the appropriate time rather than relying on the operator's memory.

If a common task requires several steps to complete, provide an interactive task checklist that indicates the current state of the procedure. (Checklists are very helpful to support complex tasks). A simpler solution could be incorporating a window showing all inputs and outputs for the system with associated timestamps.

3.4 Summary

The mechanisms that operators use to perform their work influences how the work gets done, what errors are likely to occur, and how to design to support it in the same way that how the components in electrical circuits work influences how they produce their outputs, what errors are likely to occur, and how to design with them. The most salient mechanisms of operators to improve the design of op centers are perception, attention, and working memory. These are used to generate operator behavior. They interact, and good design will be based on a theory of how they are used by operators to perform their tasks based on the information presented to them in the interface.

We include design principles to help with design. When these principles contradict themselves, which design principles and guidelines will inevitably do, the design should resort to the tasks, their importance and frequency, to resolve the design tradeoffs.

There are also other mechanisms of operators, shown in Figure 5, that will influence op centers. These mechanisms include motor output and other forms of perception. An overview of these mechanisms is available in Ritter et al. (2014) including further readings.

4. Conclusion

This report summarized a process for designing and implementing op centers in Section 1. As the work is performed, risks are assessed using a spiral development model that checks with stakeholders at each major phase, and adjusts the process based on the risks that can be perceived at that stage. The intermediate and final system can be assessed using simple usability tests as well as cognitive walkthroughs.

The process uses shared representations of the operators, their tasks, and the context of the work. An example of these is provided in Appendix 1. These shared representations are used to design and create an op center. Appendix 1 with its subsections provides an example set of documents for knowing your users and their tasks. Larger systems will need correspondingly larger and more complex documents. Smaller systems will need less. Systems only used by their developers might not need anything, but systems that are designed without these documents are designed informally and solely their designer's use, not for the operators. As architects would discuss blueprints particularly before building a project, op center designers should expect to prepare and discuss these documents during design with other stakeholders, such as managers, future operators, and funders. These discussions can reduce misunderstandings, lead to supporting all the tasks for all stakeholders, defend designs, and help keep the relevant goals, mission, and tasks in mind when designing a system. Using these documents reduces risks (Pew & Mavor, 2007).

Sections 2 and 3 provide design principles that managers, designers, and implementers can be informed by. These stakeholders can also be informed by greater knowledge of the operators as a type of system component. Section 3 provides a short overview of the types of knowledge of operators that can help inform system design and implementation. Further sources for learning more are noted in each section.

This report should also be seen as an initial review. There is more to know about how to support operators than is covered here. Appendix 2 provides pointers to further information on how to support operators in control rooms and to support the designers who create them.

4.1 The need for user-centered design

One of the difficulties with this approach will be investing the perceived additional time and effort to avoid the risks that this approach helps mitigate, ameliorate, or avoid. Typically, this approach takes additional effort, and organizations do not always see the risks until they arrive. There is evidence, however, that a mindful approach can overall reduce costs (Booher & Minninger, 2005).

A problem that remains then, is to provide evidence that there are risks and that this approach helps reduce risks and their impact. Pew and Mavor (2007) call for examples to help motivate the different team members to appreciate how usability can influence system performance. Table 7 notes a few examples. Support from management for this more engineering-based approach as well as further local examples could be useful to motivate implementer and technology designers to take operator tasks and their knowledge, skills, and abilities more seriously.

Keeping a list of known risks and accidents could be helpful in several ways. The particular risks to op centers' success may be difficult to quantify and will often arise from unexpected events. It may be worthwhile for an organization to keep track of misses and near misses to accidents, as NASA does for air traffic control in the NASA Aviation Safety Reporting System.

Table 7. Examples of usability problems leading to accidents (or extreme training or testing avoiding them). Further examples are available in Casey (1998).

The USS Vincennes Incident
The US Airways Flight 1549 that landed in the Hudson River
A Tomahawk launch system that was cancelled for not meeting response time when the problem was known (Chipman & Kieras, 2004).
Task analyses of various Army projects that lead to saving $100M's across multiple projects (Booher & Minninger, 2005).

4.2 The need for better shared representations

Another problem is the usability of the shared representations themselves. The managers, designers, and implementers can come from different intellectual backgrounds, and have different assumptions themselves. There is a need to translating some representations to "engineer speak", and perhaps in the other direction. There is a young literature on how to prepare knowledge about design aspects to share with other team members. This is a problem noted by Pew and Mavor (2007), where it is called shared representations, and work remains to make the shared representations are as useable as they can be.

4.3 Open problems

We can now revisit Table 3, presented here as Table 8. The responses are included in the table for convenience of reading and presentation.

As the material in Table 8 notes, there remain open problems with applying this approach. How detailed the documents for particular op centers and even different technologies will vary and will have to be adjusted. The risks that arise in the use of particular op centers will vary with the domain that the op center is supporting. This approach does not guarantee a perfect or even a better system, but it overall reduces risk and the probability of system failures.

Table 8. Questions answered by this document.

Process Performance

1) What user interface features reduce user stress and improve and maintain level of performance?

Reducing cognitive load will reduce user stress and improve and maintain performance. This load depends on multiple aspects of an interface. Making each of the substeps in the user's tasks will support this. Doing so is done by matching the user's capabilities with the interface.

2) Which task UI factors design mitigate performance degradation (speed, accuracy) during the execution of detailed procedures for trouble shooting?

The factors noted in answer 1, as well as avoiding interruptions or supporting their graceful entry and exit.

High Throughput Reaction Times

3) What levels of fast and complex interfaces impair or enhance user reaction time and accuracy?

The factors are detailed in the review. Briefly, making perception of the task fast and easy, reducing the cognitive load by type and number of substeps, and making the output easy.

4) What are the reaction time and accuracy for a user to react to an alert and respond to the alert with the correct actions using the task UI? What are the upper limits of number and speed of alerts before performance degrades?

We have ways to estimate the time to handle an alert. The Keystroke Level Model (Card et al., 1980, 1983) can be used to estimate response times. The upper limit must be based on an interface specified in enough detail to make predictions. The field does not have to our knowledge tools to fully compute the upper limit because the limit would depend on many things that we don't yet have fully computational or algorithmic equations for.

5) What are the reaction time and accuracy for a user to distinguish between levels of criticality using the task UI?

This time would depend on the perceptual display, the relatively frequency of signal and noise, the payoffs between signal and noise. We do not know of an equation to compute this, but an equation could be created for fixed measures and validated empirically with operators.

6) What are the effects on reaction time and accuracy for a user using the system over time?

In general, with practice, reaction time goes down (Ritter et al., 2014, Ch. 5), but fatigue goes up. There are formulas to compute the general effect of fatigue (FAST ;Hursh et al., 2004). They are validated but require some examination and understanding before use in a given situation.

Interface Generalizable and Individualized Effectiveness

7) Which interface design elements vary and do not vary in effectiveness across various demographics?

Design elements will vary based on previous experience with the design elements. The design elements would have to be specified to fully answer this question.

8) Which of the above questions are affected by age and prior education?

All of these questions are affected by age and prior education. Just how, varies on the question and the type of education. Typically, people become slower with age with raw response time, but this is typically not seen due to additional practice that contributes to lower response times as well as more knowledge which leads to better strategies and less search and problem solving. Prior education that gives practice on the task or related tasks decreases time. Education that teaches useful theory will lead to better strategies that will in time but perhaps not immediately reduce response time. Further reviews are available.

References

Anderson, J. R. (2007). *How Can the Human Mind Occur in the Physical Universe?* Oxford: Oxford University Press. https://doi.org/10.1093/acprof:oso/9780195324259.001.0001

Baddeley, A. D. (2000). The episodic buffer: A new component of working memory? *Trends in Cognitive Sciences*, *4*(11), 417–423. https://doi.org/10.1016/S1364-6613(00)01538-2

Banbury, S., Selcon, S., Endsley, M. R., Gorton, T., & Tatlock, K. (1998). Being certain about uncertainty: How the representation of system reliability affects pilot decision making. In *Proceedings of the Human Factors and Ergonomics Society Annual Meeting* (pp. 36–39).

Baxter, G. D., Churchill, E. F., & Ritter, F. E. (2014). Addressing the fundamental attribution error of design using the ABCS. *AIS SIGCHI Newsletter*, *13*(1), 76–77.

Besnard, D., Greathead, D., & Baxter, G. D. (2004). When mental models go wrong. Co-occurences in dynamic, critical systems. *International Journal of Human-Computer Studies*, *60*(60), 117–128.

Blackmon, M. H., Polson, P. G., Kitajima, M., & Lewis, C. (2002). Cognitive Walkthrough for the Web. In *Proceedings of CHI 2002* (pp. 463–470). ACM Press.

Boehm, B., & Hansen, W. (2001). The Spiral Model as a Tool for Evolutionary Acquisition. *CrossTalk*, *14*(5), 4–11. Retrieved from http://nkhalid.seecs.nust.edu.pk/SE/software p. models readings/presentation 1.pdf

Boff, K. R., & Lincoln, J. E. (1988). *Engineering data compendium: Human perception and performance*. Wright-Patterson Air Force Base.

Bolstad, C. A., Cuevas, H., Wang-Costello, J., Endsley, M. R., & Angell, L. S. (2010). Measurement of situation awareness for automobile technologies of the future. *Performance Metrics for Assessing Driver Distraction: The Quest for Improved Road Safety*, *4970*(May 2016), 195–213. https://doi.org/10.4271/R-402

Booher, H. R., & Minninger, J. (2005). Human Systems Integration in Army Systems Acquisition. In H. R. Booher (Ed.), *Handbook of Human Systems Integration* (pp. 663–698). Hoboken, NJ, USA: John Wiley & Sons, Inc. https://doi.org/10.1002/0471721174.ch18

Brown, J. M., Greenspan, S. L., & Biddle, R. L. (2013). Complex activities in an operations center: A case study and model for engineering interaction. In *Proceedings of the 5th ACM SIGCHI symposium on Engineering interactive computing systems - EICS '13* (Vol. 2, p. 265). New York, New York, USA: ACM Press. https://doi.org/10.1145/2494603.2480310

Cairns, P., & Cox, A. (2008). *Research methods for human-computer interaction*. (P. Cairns & A. L. Cox, Eds.). Cambridge University Press.

Campoe, K. R., & Giuliano, K. K. (2017). Impact of Frequent Interruption on Nurses' Patient-Controlled Analgesia Programming Performance. *Human Factors*, *59*(8), 1204–1213. https://doi.org/10.1177/0018720817732605

Card, S. K., Moran, T. P., & Newell, A. (1980). The keystroke-level model for user performance time with interactive systems. *Communications of the ACM*, *23*(7), 396–410. https://doi.org/http://doi.acm.org/10.1145/358886.358895

Card, S. K., Moran, T. P., & Newell, A. (1983). *The Psychology of Human-Computer Interaction*. Hillsdale, N.J.: Lawrence Erlbaum.

Casey, S. M. (1998). Set phasers to stun: And other true tales of design, technology, and human error. Santa Barbara, CA, USA: Aegean.

Chalmers, P. A. (2003). The role of cognitive theory in human-computer interface. *Computers in Human Behavior*, *19*(5), 593–607. https://doi.org/10.1016/S0747-5632(02)00086-9

Chang, D., Dooley, L., & Tuovinen, J. E. (2002). Gestalt Theory in Visual Screen Design - A New Look at an Old Subject. In *Selected Papers from the 7th World Conference on Computers in Education (WCCE'01)* (pp. 5–12). Copenhagen. Retrieved from http://crpit.com/confpapers/CRPITV8Chang.pdf

Chilton, E. (1996). What was the subject of Titchner's doctoral thesis. *SigCHI Bulletin*, *28*(2), 96.

Chipman, S. F., & Kieras, D. E. (2004). Operator centered design of ship systems. In *Engineering the total ship symposium*. NIST, Gaithersburg, MD: American Society of Naval Engineers.

DeMarco, T., & Lister, T. (1999). *Peopleware: Productive projects and teams*. New York, New York, USA: Dorset House Publishing.

Dow, S. (2011). How prototyping practices affect design results. *Interactions*, *18*(5), 54–59.

Ellis, N. C. (1996). Sequencing in SLA: Phonological memory, chunking, and points of order. *Studies in Second Language Acquisition*, *18*(1), 91–126. https://doi.org/10.1017/S0272263100014698

Endsley, M. R. (1995). Toward a Theory of Situation Awareness in Dynamic Systems. *Human Factors: The Journal of the Human Factors and Ergonomics Society*, *37*(1), 32–64. https://doi.org/10.1518/001872095779049543

Endsley, M. R. (2000). Theoretical Underpinnings of Situation Awareness: A Critical Review. *Situation Awareness Analysis and Measurement*, 3–32. https://doi.org/10.1016/j.jom.2007.01.015

Endsley, M. R., Bolstad, C. A., Jones, D. G., & Riley, J. M. (2003). Situation Awareness Oriented Design: From User's Cognitive Requirements to Creating Effective Supporting Technologies. *Proceedings of the Human Factors and Ergonomics Society Annual Meeting*, *47*(3), 268–272. https://doi.org/10.1177/154193120304700304

Endsley, M. R., Bolte, B., & Jones, D. G. (2003). *Designing for situation awareness: An approach to user-centered design*. (M. R. Endsley, Ed.) (1st ed.). New York, New York, USA: CRC press.

Epling, S. L., Russell, P. N., & Helton, W. S. (2016). A new semantic vigilance task: vigilance decrement, workload, and sensitivity to dual-task costs. *Experimental Brain Research*, *234*(1), 133–139. https://doi.org/10.1007/s00221-015-4444-0

Findlay, J. M., & Gilchrist, I. D. (2003). *Active Vision: The psychology of looking and seeing*. Oxford, UK: Oxford University Press.

Galanter, E. (1962). Contemporary psychophysics. In R. Brown, E. Galanter, E. H. Hess, & G. Mandler (Eds.), *New directions in psychology* (pp. 87–156). New York: Holt, Rinehart, Winston.

Garner, W. R. (1974). *The processing of information and structure*. Potomac, ML: Erlbaum.

Granados, J., Hopper, M., & He, J. (2018). A Usability and Safety Study of Bone-Conduction Headphones During Driving while Listening to Audiobooks. *Proceedings of the Human Factors and Ergonomics Society Annual Meeting*, 1373–1377. https://doi.org/10.1177/1541931218621313

Gray, W. D., & Boehm-Davis, D. A. (2000). Milliseconds matter: An introduction to microstrategies and to their use in describing and predicting interactive behavior. *Journal of Experimental Psychology: Applied*, *6*(4), 322–335.

Grunwald, T., & Corsbie-Massay, C. (2006). Guidelines for cognitively efficient multimedia learning tools: educational strategies, cognitive load, and interface design. *Academic Medicine : Journal of the Association of American Medical Colleges*, *81*(3), 213–223. Retrieved from http://journals.lww.com/academicmedicine/Fulltext/2006/03000/Guidelines_for_Cognitively_Efficient_Multimedia.3.aspx

Healey, C., & Enns, J. (2012). Attention and visual memory in visualization and computer graphics. *IEEE Transactions on Visualization and Computer Graphics*, *18*(7), 1170–1188. https://doi.org/10.1109/TVCG.2011.127

Helton, W. S., & Russell, P. N. (2011). Working memory load and the vigilance decrement. *Experimental Brain Research*, *212*(3), 429–437. https://doi.org/10.1007/s00221-011-2749-1

Helton, W. S., & Warm, J. S. (2008). Signal salience and the mindlessness theory of vigilance. *Acta Psychologica*, *129*(1), 18–25. https://doi.org/10.1016/j.actpsy.2008.04.002

Hollender, N., Hofmann, C., Deneke, M., & Schmitz, B. (2010). Integrating cognitive load theory and concepts of human-computer interaction. *Computers in Human Behavior*, *26*(6), 1278–1288. https://doi.org/10.1016/j.chb.2010.05.031

Hursh, S. R., Redmond, D. P., Johnson, M. L., Thorne, D. R., Belenky, G., & Balkin, T. J. (2004). Fatigue models for applied research in warfighting. *Aviation, Space, and Environmental Medicine*, *73*(3), A44–A53.

Irwin, D. E., Brown, J. S., & Sun, J. (1988). Visual masking and visual integration across saccadic eye movements. *Journal of Experimental Psychology. General*, *117*(3), 276–287. https://doi.org/10.1037/0096-3445.117.3.276

Jones, D. G., & Endsley, M. R. (1996). Sources of situation awareness errors in aviation. *Aviation, Space, and Environmental Medicine*.

Just, M. A., & Carpenter, P. A. (1992). A capacity theory of comprehension: Individual differences in working memory. *Psychological Review*, *99*(1), 122–149. https://doi.org/10.1037/0033-295X.99.1.122

Kieras, D. E., & Bovair, S. (1984). The role of a mental model in learning how to operate a device. *Cognitive Science*, *8*, 255–273.

Kosslyn, S. M. (2007). Clear and to the point: 8 psychological principles for compelling PowerPoint presentations. New York, New York, USA: Oxford University Press.

Krug, S. (2005). *Don't make me think: A common sense approach to web usability* (2nd editio). Berkeley, CA: New Riders Press.

Lewis, C., & Rieman, J. (1994). Task-Centered User Interface Design: A Practical Introduction.

Marsh, R. L., Cook, G. I., & Hicks, J. L. (2006). Task interference from event-based intentions can be material specific. *Memory & Cognition*, *34*(8), 1636–1643. https://doi.org/10.3758/BF03195926

Moore, C. M., & Egeth, H. (1997). Perception Without Attention : Evidence of Grouping Under Conditions of Inattention to shape constancy . Similar findings have been reported for. *Journal of Experimental Psychology. Human Perception and Performance*, *23*(2), 339–352.

Moray, N. (1996). A taxonomy and theory of mental models. In *Proceedings of the Human Factors and Ergonomics Society 40th Annual Meeting* (pp. 164–168).

Pew, R. W. (2008). Some New Perspectives for Introducing Human-Systems Integration into the System Development Process. *Journal of Cognitive Engineering and Decision Making*, *2*(3), 165–180. https://doi.org/10.1518/155534308X377063

Pew, R. W., & Mavor, A. S. (2007). Human-System integration in the system development process: A new look. National Academies Press.

Polson, P. G., Lewis, C., Rieman, J., & Wharton, C. (1992). Cognitive walkthroughs: a method for theory-based evaluation of user interfaces. *International Journal of Man-Machine Studies*, *36*(5), 741–773. https://doi.org/10.1016/0020-7373(92)90039-N

Ricker, T. J., AuBuchon, A. M., & Cowan, N. (2010). Working memory. *Wiley Interdisciplinary Reviews: Cognitive Science*, *1*(4), 573–585. https://doi.org/10.1002/wcs.50

Ritter, F. E., Baxter, G. D., & Churchill, E. F. (2014). *Foundations for Designing User-Centered Systems*. London: Springer London. https://doi.org/10.1007/978-1-4471-5134-0

Strayer, D. L., Drews, F. A., & Johnston, W. A. (2003). Cell Phone-Induced Failures of Visual Attention During Simulated Driving. *Journal of Experimental Psychology: Applied*, *9*(1), 23–32. https://doi.org/10.1037/1076-898X.9.1.23

Stroop, J. R. (1935). Studies of interference in serial verbal reactions. *Journal of Experimental Psychology*, *18*(6), 643–662. https://doi.org/10.1037/h0054651

Sweller, J. (1988). Cognitive Load During Problem Solving: Effects on Learning. *Cognitive Science*, *12*(2), 257–285. https://doi.org/10.1207/s15516709cog1202_4

Temme, L. A., Still, D. L., & Acromite, M. (2003). OZ: A human-centered computing cockpit display. In *45th Annual Conference of the International Military Testing Association* (pp. 70–90).

Trafton, J. G., Altmann, E. M., Brock, D. P., & Mintz, F. E. (2003). Preparing to resume an interrupted task: Effects of prospective goal encoding and retrospective rehearsal. *International Journal of Human Computer Studies*, *58*(5), 583–603. https://doi.org/10.1016/S1071-5819(03)00023-5

Tufte, E. R. (2001). *The visual display of quantitative information* (2nd ed.). Cheshire, CT: Graphics Press.

Tufte, E. R. (2006). *Beautiful evidence* (2nd ed.). Cheshire, CT: Graphics Press.

Ulrich, T. A., & Boring, R. L. (2013). Example User Centered Design Process for a Digital Control System in a Nuclear Power Plant. *Proceedings of the Human Factors and Ergonomics Society Annual Meeting*, *57*(1), 1727–1731. https://doi.org/10.1177/1541931213571385

Unsworth, N., Schrock, J. C., & Engle, R. W. (2004). Working memory capacity and the antisaccade task: individual differences in voluntary saccade control. *Journal of Experimental Psychology. Learning, Memory, and Cognition*, *30*(6), 1302–1321.

Appendices

Appendix 1: Detailed Problem Space: A Water Detection System (WDS)

SatCorp is an imaginary corporation that builds user interfaces for a unique class of command and control systems. These systems, while all unique, have many features that are consistent throughout their designs. The goal of this fictitious use case is to enable readers to consider an example case that is typical of such op centers. The initial draft of this system description was created with Mark Foster of L3Harris Technologies. The system description includes an overview, system architecture, key features, example day in the life (or scenarios), typical issues, user types, and task analysis. Each of these could and should be expanded in more detail for real op centers. A set of these descriptions provides a solid basis for designing with the operator in mind.

A1.1 Overview

The fictitious use case involves building a user interface to command and control a remote Water Detection System (WDS). This WDS will be deployed to Mars in an attempt to detect pockets of water underneath the surface or traces of water in the soil on the surface.

The WDS will take 5 years to develop and test before its ready to deploy. Once ready, it will be sent to Mars as part of larger manned mission. Due to space constraints on the manned vessel, the WDS will be disassembled before launch. It will be the responsibility of the team on this space mission to assemble the WDS, perform some initial checkout of the system, and ultimately deploy the WDS on the surface of Mars. During the assembly and checkout of the system, the team will command and control the system via a laptop with a local LAN connection to the WDS. The system checkout of the system is intended to exercise the different parts of the system to make sure they are still operational. Spare parts have been shipped with the system in case anything has been damaged in transport.

Once deployed on the surface of Mars, the WDS is expected to a have a 10-year mission where it is solely commanded and controlled by NASA's operations center. The operators in NASA's ops center are on duty 24/7. The WDS is only one of dozens of systems they monitor. Decision making with regards to how the WDS is utilized comes from the scientists in the Program Office who funded the development of the WDS. It is the Program Office's charter to find water sources in other locations throughout our solar system.

This example (and associated material) ignores the communication delays with Mars because most op centers do not deal with such long time-delays in communication media (although they will see delays in reports from other systems).

A1.2 System architecture

The WDS is comprised of several elements. These elements are listed with a brief description of each.

Main Control Element (MCE)

The MCE acts as the brain in the field. It is the responsibility of the MCE to facilitate commands from Earth and collect data and status to send back to Earth. More specifically, when commands are sent to the WDS, the MCE oversees executing those commands. Commands that are scheduled for future date, will reside in the MCE until it is time to execute such commands. Commands for immediate execution will be executed upon receipt. Depending on the command type, the MCE is tasked with powering on the necessary elements and forwarding subcommands to those elements. All the while, the MCE is also constantly polling the other elements for status. In addition, the MCE provides storage for water analysis data from the Rock and Sand Exploration Element and Deep-Water Detection Element. When the WDS sends data home, it is the responsibility of the MCE to bundle element status and water analysis data, perform compression and encryption, and then forward that data when appropriate to the Communications Element.

Communications Element (CE)

The CE contains the antenna for communicating with Earth. This antenna is single-duplex and therefore can only receive or transmit at a given time. Due to this limitation, the antenna is by default in receive mode to receive commands from Earth. The team on Earth must command it into Transmit mode to transmit data home. Typically, the team will schedule several Transmit commands per day on the MCE that cover roughly a week.

Autonomous Navigation Element (ANE)

The ANE controls the components of the WDS that are responsible for moving the WDS from one location to another. The ANE includes cameras for a taking pictures of the terrain around it and has special image detection algorithms for identifying obstacles it must navigate around. The ANE can be commanded to move from point A to point B, and on its own determine the best route to get there that may not be a straight line if obstacles are in the way. In addition, this element controls the drive motor, wheels, and steering functionalities. It also controls the emergency assist wheels and arms that enable it to get out of precarious physical situations.

Rock and Sand Exploration Element (RSEE)

The RSEE controls the shovel like apparatuses the WDS is equipped with. It also controls the cameras and sensors that are used to evaluate a segment of sand or rocks. Data recorded from this element is forwarded to the MCE for storage until it is sent back to Earth for analysis.

Deep Water Detection Element (DWDE)

The DWDE controls the drill and soil probe the WDS uses to search for water underneath the surface. When commanded to do so, the DWDE will drive the probe into the ground to gather water analysis data. In cases where the ground is too solid, the DWDE will remove the probe, and use the drill to loosen the ground underneath the surface. After drilling, the probe is reinserted into the ground to continue gathering water analysis data. Like the RSEE, data recorded by this element is forwarded to the MCE for storage until it is sent back to Earth for analysis.

Power Generation Element (PGE)

The PGE consists of solar panels and the system batteries. The PGE has a set of solar panels that are distributed around the WDS. These panels are used to generate power and charge the system batteries. The solar panels can rotate and tilt as needed to maximize sun exposure. The PGE is responsible for calculating the ideal rotation and attitude.

A1.3 Key features of the WDS

The following sections outline the key features the user interface must accommodate.

Status

All six system architecture components listed above contain numerous status items that must be reported on a regular basis. Status can range from environmental measures such as pressure, temperature, and humidity, to element-specific status such as current speed (mph or kph) for the ANE.

One of the roles of the MCE is to periodically poll all the components for their latest status values. The MCE then stores all these values until the next opportunity to transmit data to Earth. It is important to note, the MCE polls only the A or B side of a given component, depending on which side is currently booted. If the MCE attempts to poll a given component that is unresponsive, the MCE can power cycle that component or even switch sides of that component. This usually only occurs after some threshold of unresponsive polls. This threshold is configurable.

Event Logs

Like reporting status, each component is recording an event log of the activities that component is executing. Periodically, when a given component's log reaches a given threshold, that component will start a new log, and transfer the old log to the MCE. The MCE will send all the logs home at the next opportunity to transmit data to Earth. The command and control GUI back on Earth will consolidate these logs into a single system log, but typically needs to filter out element specific details.

Configuration

Each element also maintains a set of configuration fields. For example, the MCE may be configured to power cycle a given component after a certain number of unresponsive polls by the MCE. This value is configurable because different scenarios may want to power cycle if 3 polls are unresponsive, while others may want to wait until 10 polls without responding. In addition, whether to power cycle a component or power cycle a component and switch sides is another configurable feature of the MCE. Another common configuration field is which side of a component to use. The MCE holds a field for each component, such that when commands are received from Earth, the MCE know which (A or B) side of each component to power on to execute the commands.

Commands

While commands are always sent to the MCE, each component supports a set of its own commands. For example, a Transmit command that is scheduled for one week from the current day would reside on the MCE's schedule for a week. Then shortly before the Transmit command, the MCE would power on the CE, pass it the Transmit command, and a bundle of data

to transmit. At the scheduled time, the CE will then execute transmitting the data bundle back to Earth.

Redundancy

The WDS system will be deployed to Mars for a 10-year mission. During those 10 years, there will not be any maintenance missions, so every part of the WDS must have built in redundancy to assure the system can last 10 years. Except for the PGE's solar panels, every component has both an A and B side. For example, the MCE has two processor boards. One known as the A side and one known as the B side. The system only uses one at a time but can be configured to use either side. Furthermore, each side of a component has its own status. For example, the RSEE uses advanced moisture sensors to detect traces of water in the soil. In this case, the A side has a set of moisture sensors, and the B side has a completely different set of moisture sensors. Similarly, the network that connects all these components is also completely redundant. There is an A and B network.

A1.4 Day in the life

A day in the life of the WDS is often unique. Table A1.1 is a timeline for an example 24-hour period (24 to a day, scaled from the Martian cycle). For the purpose of this use case, the Mars daylight hours will mirror those of Eastern Standard Time.

Table A1.1. Example Day for WDS.

Time	Activity
00:00 – 06:00	System idle time to avoid draining batteries below emergency shutdown threshold.
06:00 – 06:15	Receive the following Immediate commands from Earth: * Relocate to the Tarakan Crater. * Survey the surface of the Crater. * Find Location of the Tarakan Crater Low Point. * Relocate to Tarakan Crater Low Point. * Probe the Tarakan Crater Low Point. Because these commands are "Immediate" commands the MCE will begin executing them in the order they were received. The MCE will maintain a queue of these commands until they are all complete.
06:15-07:25	The MCE begins the first Relocate command. It starts by powering on the ANE. The ANE takes about 4 minutes to boot. Once booted, the MCE passes the command to the ANE. The ANE begins calculating its navigation plan to the Tarakan Crater.
06:25-07:25	The ANE drives the WDS towards the Tarakan Crater.
07:25-07:30	The ANE is continually imaging the terrain and detects an obstruction in its path to the crater. The ANE stops driving and recalculates a new navigation plan.
07:30-07:35	The ANE continues driving towards the Tarakan Crater.
07:35-07:40	While driving, the MCE powers on the CE, as there is a scheduled Transmit command for 08:20 today. The CE takes about 3 minutes to boot, but the MCE has several GB of data that will take about 30 minutes to bundle, compress, encrypt, and copy over to the CE. All of this will occur in the

background while the system is doing other activities.

07:40-07:50 The ANE finishes driving to the crater and locates itself in the center most point of the crater.

07:50-07:55 The MCE receives events from the ANE that the Relocate command is complete. The MCE then begins the Survey command. It starts by powering on the RSEE. Even though the Relocate command is complete, the MCE does not power off the ANE, as the MCE knows it will need the ANE powered up to conduct the Survey command. Once the RSEE has booted, the MCE sends both the RSEE and ANE the Survey command.

07:55-08:20 The two elements then begin executing their commands in tandem. A survey is conducted by the ANE slowly navigating the WDS over a given area, while the RSEE continually scoops sand and rocks to gather water analysis data. Both elements are logging events while executing their commands. The MCE will monitor both their event logs to make sure they are staying synchronized. Due to the size of this crater, this survey will take up most of the day.

08:20-08:30 The CE executes a Transmit command.
The Survey continues.

08:30-12:30 The Survey continues and completes.

12:30-12:40 The MCE finishes receiving the water analysis data from the RSEE and the corresponding events such that the MCE knows the RSEE has completed the survey. The MCE shuts the RSEE down.

12:40-12:45 The MCE kept the ANE power up, and now passes the Find Location command to the ANE. The ANE uses its terrain data to determine the lowest point of the crater. Via events the MCE is notified the ANE has completed the Find Location command.

12:45-12:55 The MCE passes the next Relocate command to the ANE. The ANE drives the WDS to the low point of the crater.

12:55-13:10 The MCE sees the ANE has completed the second Relocate command and powers down the ANE. The MCE then powers up the DWDE. The DWDE takes about 8 minutes to boot up. Once booted, the MCE passes the Probe command to the DWDE for execution.

13:10-17:40 The DWDE executes the Probe command but encounters a lot of solid rock. This forces the DWDE to alternate between Probe and Drill frequently. After over 4 hours of mostly drilling, the batteries have taken a significant hit, because the solar panels cannot keep up with the power needs of the drill.

17:40-19:35 A (configurable) low battery threshold is reached that causes the MCE to take over and pause the Probe command. The MCE powers down the RSEE and transitions into an idle mode to allow the system to charge.

19:35-22:00 The sun has set, and system can no longer charge the batteries again until the next day.

22:00-22:50 The MCE powers on the CE, as there is another scheduled Transmit command for 22:40 today. While the batteries are still not charged enough for a drilling activity, the battery threshold for a Transmit is much lower. Batteries are sufficient for a Transmit command and therefore the system

successfully transmits at 22:40.

22:50-23:59 System resumes idle mode. This will continue until the next day's sunrise.

A1.5 Example issues

The WDS is designed to autonomously handle issues that arise, but human interaction is required on a regular basis. Many of these tasks are simple maintenance and acknowledgement of warnings. For example, when batteries are low, the operator is required to acknowledge the low battery threshold. No action is required other than clearing the notification. Occasionally, however, the WDS will face an urgent problem that requires human input. These scenarios are rare, so the operator typically has limited training in how to address the issues. Here are some examples:

Problem: The WDS is navigating in the crater and gets stuck.

Operator from Earth must manually drive the WDS and control the ANE. The typical operator is not trained in this task, so the supervising manager must take control. The operators need to escalate the issue quickly because the WDS witnessed unexpected terrain. The mappings of Mars must be updated appropriately.

Problem: Dust storm prevents batteries from charging.

The MCE cannot task all the scheduled commands for the day. The CE alerts the NASA operators of the low battery status. The operator must re-task the day's commands because the ANE would use all the remaining power. This task is simple and can be completed by a novice employee but will require review by a supervisor.

Problem: Wall of Screens has many other systems represented at the same time.

If the WDS has a problem, it might take a few days for the engineers to remote in to fix the issue. Therefore, the overview screen will remain in a degraded (fault-shown) state. The problem arises when something else goes wrong on the system. For example, while at low power, a piece of equipment might become over temperature and be in danger of catching fire. The operators need to be alerted to this new degraded status and respond quickly.

A1.6 Stakeholder analysis

When designing a system, it is worthwhile keeping the stakeholders, the audience for the system, in mind (Boehm & Hansen, 2001; Pew & Mavor, 2007). Stakeholders for the Water Detection System (WDS), and other complex systems, will follow a similar structure as the one shown in Table A1.2. Direct users (i.e., operators), funders, and other stakeholders will each have their own requirements for the project. The stakeholders identified for the WDS are described in the rest of this section.

Table A1.2. List of stakeholders and a brief overview of their role in the project.

Stakeholder	Role
NASA 24/7 Operators	Lower-skilled workers that handle routine tasks on various systems within the op center.
Operation/Command Center Supervisors	Experienced managers that handle complex tasks and monitor op center performance.
System Developers and Engineers	Experienced engineers that build and maintain the system.
NASA Program Office Scientists	Highly experienced project managers that direct the WDS actions and use the data that is collected.
Project Funders and other High-Level Stakeholders	Outside managers responsible for ensuring project success and making high-level project decisions.
On-Site Astronaut Install Team	Extremely skilled operators that will deploy and troubleshoot the system (if necessary).

NASA 24/7 Operators

Primary operators (or users) of the system are those that that perform routine activity monitoring, respond to low-level alarms and events, and identify issues that require outside performance. They want a task that is within their knowledge, skills, and abilities and provides them with job satisfaction.

The primary objectives of the 24/7 Operators is to monitor the WDS for anomalies or issues and maintain communication with the WDS. It is the role of the operators to plan sets of transmit commands for the WDS system (which requires coordination with third party communication systems) and send those commands to the WDS. Additionally, they must monitor the WDS interface to verify the WDS has transmitted data to Earth when it is scheduled to. Upon receipt of this data, the operators perform a cursory review of the data to determine if there are any system issues that need to be addressed. In most cases, upon discovering a system issue the operators will contact the Program Office or Engineering Development Team to troubleshoot the issue. Lastly, the operators are expected to respond to requests for information regarding the WDS. At any time, if the Program Office or Engineering Development Team needs some data points from the system, the operators should be able to retrieve that data for them.

The risk of overall project failure due to operator abilities and needs is relatively more difficult to specify due to the delay between operator feedback and interaction with a system. The most common sources of major failure will likely be due to unforeseen issues that are preventable by experienced (or lucky) operators that can react to the system beyond the pre-determined alarm and event conditions. For example, a system overheat event can lead to a positive feedback loop of further heating of other components that destroys key components. This could plausibly have been detected by a perceptive operator, but system alert priorities might not directly reveal this as a critical issue until it was too late.

A source of "minor" project failure could be through overall issues with design that lead to high error rates that increase project cost and reduce the perceived reliability of the system. While an operator taking the wrong action (e.g., a command scheduling issue is first reported to the system's development team before calling Program Office Scientists) is a relatively minor issue at first, high error rates from operators increase costs of the project and reduce the overall effectiveness of the operation center.

The environment the operators work in is a command center that is staffed 24/7 with approximately 15 workstations and an average staff of 10 operators. The primary environment is a "dim" room with desks in the center (i.e., not along the walls). The front wall, which all the desks face, is a wall of screens. The back wall, which no one faces, is a secondary wall of screens. Both walls of screens consist of multiplexed, disparate displays of 40-100 systems.

There are approximately 12-15 operators with 1-2 task leads during the day and 10 operators with one task lead at night. Operators alternate in twelve-hour shifts, with a day shift from 7am to 7pm, and a night shift from 7 pm to 7 am.

The night shift operators are typically former enlisted personnel, hence generally not college educated, and mostly in their early twenties. The day shift workers typically have a more advanced skill set than the night shift operators. The average age is greater compared to the night shift. The day shift operators tend to have more system knowledge and can handle slightly more advanced troubleshooting or analysis than the night shift.

Operation/Command Center Supervisors

Supervisors within the command center that ensure operator performance and respond to high-level alarms and events upon notification by the primary operators. Like operators, the supervisors want job satisfaction and a task that is within their abilities.

The supervisor's use of the system will share mostly the same set of risks as operators; risks to project failure will likely be the result of unforeseen issues that could be successfully caught with experienced or skilled workers. Supervisors act as the interface between the high-level management from NASA research scientists and the ground-level operators that directly interact with the op center systems.

System Developers and Engineers

The Engineering Development team is a cross-discipline team that has developed the WDS over a 4-5-year period. While during the development phase, the WDS program consisted of hundreds of engineers, now nearing deployment the program has reduced to essential personnel. Most of the remaining personnel are Software Engineers, Systems Engineers, and Integration/Test Engineers. This team's primary responsibility is to work off bug tickets regarding the WDS software. This team is continually integrating and testing the latest software. Once a software release is ready, it will be loaded to the WDS, whether the WDS is still being used for training at NASA or if it has been deployed on Mars. In addition, any issues or anomalies with the system are investigated by the Engineering Development team in their development lab.

The developers want mission success (as measured by other stakeholders), an easily programmed system, clear instructions, and to generally avoid "hard mental operations" leading to difficult to program constructs when possible.

Developers will need to able to create the system within the constraints of the other stakeholders while also meeting their funding and time constraints. Besides these "common" risks that engineers should be familiar with, the other major risk of project failure facing developers is ensuring that all the needs of the system and users are met. The example of a major failure described under "operators" would partially be the fault of the developers (for not identifying the tasks and needs), the Program Office Scientists (for not providing an adequate list of tasks and needs), and possibly the Op Center Supervisors depending on the circumstances.

However, the developers should make strides to gather this information or risk having their reputation be negatively affected (whether or not the failure is directly related to their decisions).

The Engineering Development team works primarily in a large lab with the same equipment that will be or has been deployed on Mars. This enables the team to test the software releases and procedures before releasing updates. The team is available to address any issues that arise after deployment. The Program Office Scientists relay the issues that are presented by the NASA operators. Occasionally the Engineering team can interface directly with NASA to get their feedback on the WDS software, but this is usually limited. Therefore, the team must prioritize tasks based on Program Scientists feedback. The Engineering team tests software updates with their mock hardware.

NASA Program Office Scientists

The Program Office Scientists are highly educated individuals whose charter is to find water on Mars. This team is formally the customer for the Engineering Development team, and while colleagues of the operators, receive customer-like status when in the operations center. This team owns the decision making on everything from design details to live mission judgment calls. They are the consumers of the water analysis data received from the WDS. They will use this data to generate reports for upper management at NASA and politicians. Their work heavily influences the direction of our country's Space Program. This team decides where the WDS should navigate on Mars, and when the WDS should attempt to gather more water analysis data.

They need to be able to complete all necessary technical tasks (which are assumed to be known to the developers and engineers for the system). They also need to be able to interpret the data from the WDS, input and alter commands, and interact with the WDS via the same GUI as the operators that work within the operation center.

Program Office Scientists should be able to provide an adequate set of requirements for the system or risk finding out that their needs are unable to be met once the WDS arrives on Mars.

The Program Office Scientists interface with the WDS via the same command and control GUI as the 24/7 operators. They frequent the operations center during business hours and especially around the time when transmit commands are scheduled with the WDS. While their primary expertise is in the science behind the water analysis data, they are fairly well versed with the WDS, as most of them have been a part of this program during the development of the WDS. Furthermore, most of them have experience working on similar systems deployed to other parts of the galaxy.

Project funders and other high-level stakeholders

The various individuals and organizations that oversee the project and provide funding for the work. They will be responsive to the assessments from the Program Office Scientists, explanations from the Developers, and requirements from the Supervisors within the operation center. However, they also have their needs and desires for the project. They may require design features based on a naïve understanding of the project's technical and scientific needs. For example, they may refer too great a consistency across projects (e.g., a common event log button across all systems), the use of incompatible software or hardware, or to prioritize a task (and interface elements) that does not correspond to other stakeholder needs. They also may provide necessary restrictions on work due to classification or other regulations that limit otherwise valuable sources of collaboration and feedback. They often want to have mission success with reduced resource costs.

As the funders of the program, high-level personnel will have their own expectations for project success. These expectations may differ from the assessments made by the Program Office Scientists, System Developers, and other stakeholders. Many of the risks to system failure will come from lack of communication or miscommunication between the stakeholders.

NASA Astronaut Install Team

The astronaut install team is the last primary stakeholder for this project. They are responsible for assembling the WDS, conducting pre-deployment tests on the system, and launching it (thus releasing it from their responsibilities). This primarily provides technological requirements (e.g., the device must be able to be assembled with the resources available to the astronauts). Besides the technological requirements, they will need to be able to interact with the ground team to troubleshoot any issues or pass off the machine for remote troubleshooting via the operation center.

The installation environment for the installers is obviously Mars. Therefore, their time is very limited as their mission is bounded by the resources (i.e., air, water, food, fuel) they have with them. Their energy levels are expected to be perpetually compromised after the extended time in space required to travel to Mars. Due to the annual meteor storm on the sector of Mars where the Program Office desires the WDS to be deployed, the install team will not have communication with Earth during the installation.

Summary and Lessons

Each project will have multiple stakeholders. The list of relevant stakeholders is not simply limited to users that directly interact with the completed system or the implementers of the system. System success requires integration of the needs of the various stakeholders into a cohesive project plan that addresses their needs, capabilities, and abilities. This example system also has a wide range of stakeholders. Like other systems, there can be conflicts and tradeoffs between their goals.

A1.7 Task analysis for 24/7 operators

The hierarchical task analysis developed for the NASA 24/7 Operators provides a clear set of the most important tasks performed by the operators. The interface of a system should be designed to match the needs and capabilities of the stakeholders that are impacted by the interface. We focus on the 24/7 operators to provide a blueprint for the tasks that need to be accomplished using any interface designed for the WDS system.

Table A1.3 gives an overview of the tasks described by the task analysis. Following the table is the detailed view of the tasks showing subtasks and other components. This turns into an operation manual for the study and task list for performing a cognitive walkthrough of the interfaces (Polson et al., 1992).

Table A1.3. Overview of the tasks for the NASA 24/7 Operator for managing the WDS.

Task 1: Periodic comprehensive review of WDS system.
Task 2: Repair or respond to any alarms following a WDS data update.
Task 3: Ensure WDS transmits data to Earth per schedule and troubleshoot any delays.
Task 4: Send commands to WDS.
Task 5: Responding to information requests regarding the WDS.
Task 6: Respond to other events, alarms, and alerts that occur in non-WDS systems.

The six tasks shown in Table A1.3 are an overview of the responsibilities for the operator of the WDS within an Op Center. Each task is decomposed into subtasks to identify the key steps and decisions taken by an operator while completing the task.

Task 1: Periodic comprehensive review of WDS system

Assumptions: WDS Periodic update takes 300 30 seconds

1) Identify if a comprehensive review of the WDS is necessary
   a) Find and check the WDS review schedule
   b) Compare time for the scheduled review and the current time
      i) If review is not necessary, END TASK
      ii) If review is necessary, proceed to 1b
2) Check the WDS update time and ensure that there is at least 3 minutes before next update
   a) If time before next update is insufficient, POSTPONE until after next update END TASK
   b) If time before next update is greater than 180 seconds, proceed to 1c
3) Perform the WDS periodic review
   a) Complete the WDS periodic review checklist
   b) Record the findings of the checklist in the <appropriate location> and END TASK

Task 2: Repair or respond to any alarms following a WDS data update

1) Identify the cause of the alarm
2) Fix high priority alarms first
   a) If alarm origin is Power Generation Element, proceed to 2bi1
      i) Check expected charge and determine if expected charge will bring battery above the acceptable threshold
      ii) If charge will resolve alert, contact NASA Program Office Scientists and report overtasking of battery then return to step 2b
      iii) If charging is low or nonexistent, contact WDS Development Team and report battery charging failure then return to step 2b
      iv) If issue is unknown, contact Op Center Supervisor and report unknown issue with PGE then return to step 2b
   b) If WDS requires manual navigation control, proceed to 2bii1
      i) Contact Op Center Supervisor and report WDS request for manual control
      ii) Return to step 2b
3) Fix low priority alarms and latching alerts
   a) Determine cause of latching alert
      i) If latching alert originated from WDS, proceed to 2ci1a
         (1) Find WDS element that sent the latching alert
         (2) Identify the command schedule file used during the alarm

(3) Report the command schedule file, WDS element, and status data associated with the element to the NASA Program Office Scientists

ii) If latching alert did not originate from the WDS, proceed to 2ci2a

(1) Report the latching alarm origin and any other associated information to the Op Center Supervisor

4) Resolve any event notifications

a) Identify special event priorities, if any, that have been requested by the NASA Program Office Scientists or Op Center Supervisors

b) If new event notifications match special event priorities, proceed to 2dii1

i) If special event priorities include action plan for event occurrence, follow instructions from the action plan

ii) If no action plan is present, report event occurrence and associated data to the program that placed the special event priority

c) If no special priority events are found, dismiss all new events

5) If all events, alerts, and alarms are processed, END TASK
Else, return to step 2a

Task 3: Ensure WDS transmits data to Earth per schedule and troubleshoot any delays

Assumptions: The scheduled update timeframe includes a margin of error. Experiment will use a five-minute update schedule with a 30-second margin of error. The timeframe and margin of error reduces unnecessary alarms for missing updates during normal operation.

1) Find the WDS interface and expected time of next update

a) If the update has not loaded AND the update is not due

i) END TASK, Resume other duties

b) If update has loaded, check ensure the next update time is shown and END TASK

c) If the update is not here AND the update is due, check the margin of error for the update schedule

i) If within the margin of error, perform other duties until margin of error passes

ii) If update has not appeared after margin of error, continue to the next step (b)

2) Follow the troubleshooting protocol for a missing update

a) Check if the file was received

i) Go to operation center event log

ii) Determine if a file update event is found within update time frame within the op center event log

(1) If file not received, check for connectivity issues between satellite and operation center

(a) If connectivity issues, call Comms team, inform of missing file, and END TASK

(b) If no connectivity issues, call WDS Development Team, inform of unknown cause of failed data upload, and END TASK

(2) If file was received, determine cause of failed update via event logs

(a) Check operation center event logs and look for an application error

(b) Check operation center event logs and look for an error processing file

(c) If either is found, call EIT, report the error, and END TASK

Task 4: Ensure that WDS maintains a regular, constant supply of commands throughout use

Assumptions: A single schedule for WDS commands may cover a variable amount of time. The experiment will use command files that cover approximately 15 minutes.

1) Determine if WDS needs new commands within the next 10 minutes

a) Find the WDS commands details module
b) Check the latest WDS command file's end time
c) If the end time is more than 10 minutes away, END TASK
d) If end time is within 10 minutes, move on to 4ab

2) Receive (or acquire) the new command file from NASA Program Office Scientists
   a) In the command details module, find the section showing commands waiting to be uploaded
   b) If no commands are present, Call NASA Program Office Scientists, report lack of commands, and END TASK
   c) If new commands are present, proceed to 4c
3) Set the new command file to be uploaded to the WDS
   a) Verify that new command file is ready for update
   b) Schedule command file update
4) Verify command file is sent and received by WDS
   a) Wait until next update from WDS
   b) Check Comms event log for a successful command file download during last update cycle
   c) If event is found within correct time window, END TASK
   d) If no event is found, call Op Center Supervisor, report findings, and END TASK

Task 5: Responding to information requests regarding the WDS

1) Identify the element or module associated with the information request
2) If event related, go to the location of the event history for the element or module
   a) Isolate the requested event history
   b) Transmit the requested event history to the NASA Program Office Scientists
3) If related to current status for the element or module, find the element or module widget
   a) Isolate the requested information for the NASA Program Office Scientists
   b) Transmit the requested current status information to the NASA Program Office Scientists

Task 6: Respond to other events, alarms, and alerts that occur in non-WDS systems.

1. Recognize an alert from a non-WDS console.
2. If currently working on a WDS task, appropriately determine prioritization
   a) Request supervisor support if unsure of appropriate priority.
3. If currently working on WDS task, note the stopping point in the activity.
4. Resolve the non-WDS events, alarms, and alerts according to their system's protocol.
5. Return to WDS and complete task.
   a) Check for system changes
   b) Identify stopping point for interrupted task.
   c) Complete interrupted WDS task.

Appendix 2: Ways to learn More

Designers of control rooms will need to know more about design and operators than is in this short document. They will need to know more theory about design and human users, and they will need more details about the situations and operators and tasks that they are designing for. This appendix notes a few ways to learn more. These ways include reading, discussion, and formal and informal education. An hour a week of learning is not much in a week, but in a year, it can change how you think.

A2.1 Readings to learn more

Designers wanting to learn more about design and operators can most easily read more. There are numerous books on how operators (as people) think and learn. A good book of this type is Anderson's *Cognitive psychology and its implications* (2004). There are similar books for learning about perception (Sekuler & Blake, 2001). Norman's (2004) book helped start the area of human-computer interaction but does not provide a unified theory of how to support design. It makes the case for paying attention to users and provides food for thought. As design moves in different directions, related books and textbooks can be found. For example, to include how emotions influence use (Norman, 2006).

There are also books describing operators in terms that support design. Our favorite is *Foundations for designing user-centered systems* (Ritter, Baxter, & Churchill, 2014), but textbooks by Wickens (e.g., Wickens & Hollands, 2000) and Lewis and Rieman (1994) are also useful. If detailed knowledge about users is required, one can try to find the information in Boff and Lincoln's (1968) large compendium, but often the designer will be driven to asking experts, running a study, or making an educated guess based on similar circumstances. Finally, Endsley's book *Designing for situation awareness* (Endsley, Bolte, et al., 2003) provides further useful advice. It will be familiar because we use it extensively in this report.

A2.2 Reading groups

A way to solidify knowledge from reading and to learn information not completely codified yet, is to be part of a reading group. Sometimes these groups appear as graduate courses, and they can be organized around a work group or better, across work groups. They take time, but a group can help digest a book, and even the social loafers who do not read the material can learn something. It is also a way to build a shared theory of design in a workplace.

A2.3 Continuing education

Finally, the most solid but expensive way to learn more is to take courses. Some will be available at local universities, and some are available online. Coursera and Lynda offer various courses that are related to these topics.

A2.4 Readings referenced in Appendix 2

Anderson, J. R. (2004). Cognitive psychology and its implications (5th ed.). New York, NY: Worth Publishers.

Boff, K. R., & Lincoln, J. E. (Eds.). (1988). Engineering data compendium (User's guide). Wright-Patterson Air Force Base, OH: Harry G. Armstrong Aerospace Medical Research Laboratory.

Endsley, M. R., Bolte, B., & Jones, D. G. (2003). Designing for situation awareness: An approach to user-centered design. (M. R. Endsley, Ed.) (1st ed.). New York, New York, USA: CRC Press.

Norman, D. A. (2006). Emotional design: Why we love (or hate) everyday things. New York, NY: Basic Books.

Ritter, F. E., Baxter, G. D., & Churchill, E. F. (2014). Foundations for designing user-centered systems: What system designers need to know about people. London, UK: Springer.
Sekuler, R., & Blake, R. (2001). Perception. New York, NY: McGraw-Hill.
Wickens, C. D., & Hollands, J. G. (2000). Engineering psychology and human performance (3rd ed.). Prentice-Hall: Upper Saddle River, NJ.

Appendix 3: Design Guidelines for asynchronous autonomous systems

This appendix provides guidelines for desktop implementations of operation center interfaces. The guidelines draw heavily on Apple's Human Interface Guidelines[7] for desktop applications but are modified to apply to the WDS system, its users and technology, and the users' tasks.

These guidelines are annotated, modified, and abridged to assist designers and engineers during the development of the applications and systems within operation centers. They are numbered and where appropriate sub-numbered. They are annotated according to four criteria: evidence level, testability, value added, and assessment for testing by the authors.

Table A3.1: Criteria definitions for the design guidelines.

1) Evidence Level (Ranging from a case study within op center to some consensus from experts)
   a) Level 5 is highly supported by research directly on the design feature.
   b) Level 4 is highly supported by research but without a direct case study on the design feature.
   c) Level 3 is likely supported based on integrating literature and expert opinions.
   d) Level 2 is plausibly supported by research and supported by multiple expert opinions
   e) Level 1 is broadly accepted as valuable by the field of HCI, but may be ['untestable'], or untested to our knowledge
2) Testability
   a) T- (Difficult to test overall or difficult to test without major work)
   b) T (Middle)
   c) T+ (Easily or close to easily testable)
3) Value added by experiment (e.g., Avoiding attentional tunneling vs perfect shade of blue)
   a) V1 (Low value), V2 (Moderate value), V3 (Most valuable)
4) Should we test this?
   a) Yes, No, Maybe, No need

The criteria are represented after the guidelines in the following format:

Example guideline **[Level 1], [T+], [V1], No**

For this example, the format means that his guideline has some support from UCD and HCI experts (level 1), could be easily tested for a given interface (T+), would not be much value to test (V1) for a given interface, and is not recommended to test by the authors (No)

In the case of complex guidelines, like the first guideline, we apply a general level without breaking it down to every sub-statement, which might not be a guideline but an example, or might not have the same level of support. If only the high-level heading is rated on the criteria, please assume that the guidelines below that heading are a "set" that should be considered as a whole (e.g., heading D: Help and Tooltips under General User Interaction Guidelines). Otherwise, the high-level heading rating should be considered an overall assessment that is somewhat like an average of the ratings for individual guidelines.

Finally, the support and evidence for the guidelines is provided in comments appended to the guidelines. A list of useful acronyms is described below in Table A3.2. These will cover the majority of the evidence support, but some guidelines are also supported by links to full references to the research articles.

[7] https://developer.apple.com/design/human-interface-guidelines/macos/overview/themes/

Table A3.2: Common Acronyms used throughout the guidelines and comments.

Acronym	Meaning	Source
GOMS	Goals, Operators, Methods, and Selection Rules (task analysis variant)	(Card et al., 1983)
CPM-GOMS	Critical Path Method-GOMS (task analysis variant)	(Gray, John, & Atwood, 1992)
FDUCS	*Foundations for Designing User-Centered Systems* (user-centered design textbook)	(Ritter et al., 2014)
CWT	Cognitive Walkthroughs (A usability method and its rationale)	(Lewis & Rieman, 1994; Polson et al., 1992)
ADG	Apple Design Guidelines (Expert opinions)	https://developer.apple.com/design/
FOK	Feeling of Knowing effect	(Ritter & Reder, 1992)
TA	Task Analysis literature	(Ritter et al., 2014)
WMTIH	Writing mistakes that I hate (essay by Frank E. Ritter)	(Ritter, 2016) http://acs.ist.psu.edu/ist597/writing%20tips3.pdf
LR	Literature Review (HCD review for Harris)	(Oury & Ritter, 2018)
ISO/CD	International Standards Organization Committee Draft 9241-151	(Bevan & Spinhof, 2007), ISO 9241-151:2008
PM	*Human-System Integration: A new look*	Pew & Mavor's (2007) National Research Council Report
NN/g	Nielson Norman Group (Expert opinions/blog)	https://www.nngroup.com/

Introduction/Design Themes [Level 5], [T-], [V3], No

CWT, FDUCS, GOMS, others.

It is helpful for users to be able to anticipate design elements in an interface. It is useful, thus, for the elements to appear to be drawn using the same overall design framework with the same color palette, style and use of verbiage, style of tone, and word choice (e.g., word length, concreteness of words, use of articles, verb tense, and representational mapping). The same things should always appear as the same things, so differentiation can be reserved for useful, functional differences.

Thus, conducting a design review after a multi-person team finishes building an interface can be a useful method for improving the coherence of the design. A thorough design review will help pull the interface elements together and meld them into a coherent, intuitive whole that allows users to draw from a unified set of task and context knowledge applicable across all Harris systems. Design reviews can be made even more effective by implementing methods like

heuristic evaluation by HCI experts on system design and cognitive walkthroughs to evaluate the system interactions.

General user interaction guidelines

Loading and Delays [Level 5], [T-], [V2], No/Maybe

Operators want an application that acts on their commands and communicates how long processing will take. If your application presents blank or static content and does not provide feedback, people might think your app is frozen.

1) Provide instant acknowledgement of user interactions. Users expect to receive feedback for their actions throughout the interface. For example, buttons should visually respond to clicks and the pointer should change depending on its location (when appropriate). **[Level 5], [T], [V2], Maybe**

   FDUCS §6.2.3: Feeling of Knowing and Confidence Judgements. Swift feedback helps users develop their knowledge for working with the sytem and avoid confusion.

   LR §2.2.2 Stage 2 – Comprehension. Support comprehension by providing users with awareness of the system state.

2) Help people gauge how long a process will take to complete by providing time estimates, activity spinners indicating action, and preferably an explicit progress indicator and supplementary descriptive text. **[Level 4], [T], [V2], Maybe**

   LR §2.2.2 Stage 2 – Comprehension. Support comprehension by providing users with awareness of the system state.

3) Show content as soon as possible by showing placeholder text, gradually improving image quality, and preloading content when possible. **[Level 3], [T-], [V1], No**

   LR §2.2.2 Stage 2 – Comprehension. Support comprehension by providing users with awareness of the system state.

Supporting Novice and Expert users [Level ≈4], [T+], [V2], Yes

Installation of op center systems may include up to 6 weeks of training to support new users, however replacement workers may not receive that same support. These systems should accommodate experienced and novice users by providing in-system tools that enable learning of new tasks and reviewing procedures for uncommon or obscure tasks.

1) Establish a default configuration that's applicable to most or all operators. **[Level 3] [T], [V3], Yes**

   LR §2: Know your users, tasks. LR §3.1.5: Design to accommodate colorblindness.

2) Avoid unnecessary splash screens and instructions. Typically splash screens are fine for showing progress, but they are often just for show. If tutorials or intro sequences are necessary, provide a way to skip them. **[Level 3], [T-], [V1], No**

   FDUCS §11 & §12 on Task analysis; ADG. Splash screens can waste time, but also can be a source of feedback as the system loads. Splash screens can provide information at the expense of task efficiency.

3) Anticipate the need for help and provide integrated help features. **[Level 4], [T+], [V3], Yes**

   a) Proactively look for times when people might be stuck. For obscure work and uncommon tasks, provide additional help in menus.

   b) Add help tags to system-specific controls

   c) Provide task-oriented documentation through a form of supplementary help documentation (either digitally or as a physical copy of a help document).

LR §2.2.2 Stage 2 Comprehension; LR §3.3 Working Memory and Cognition. Providing integrated help reduces cognitive load by reducing the amount of time spent searching for help and reducing the time and space between the issue and task completion. Including a help button would allow users to find help when needed, and also provide a metric for which screens or tasks needed the most help. Testing could be done by comparing how users respond to in-system help, providing a physical help guide, and another option.

4) Use keystroke accelerators (KSAs) to improve performance of expert users. **[Level 5], [T+], [V3], Yes/maybe**

   a) Provide KSAs in menus to support learning.

   b) Base KSAs on typical Windows KSAs like ctrl-s for Save and ctrl-p for Print.

   c) Provide a full list of KSAs that can be viewed and/or printed out.

   GOMS, ADG, FDUCS. Clearly using KSAs would improve performance, however outstanding questions include the value of KSAs for each task, time required to learn KSAs, and maybe others.

Data entry [Level 3.5], [T+], [V3], Yes

Whether using a keyboard, mouse, or any other input mode, inputting information can be a tedious and sometimes error-prone process. When an app asks for lots of input before doing anything useful, people can get discouraged quickly.

1) When entering data, prompt operators to choose an input rather than enter free text whenever possible. Selecting from a table, pop-up button, or set of radio buttons improves accuracy and reduces error rates, especially when the input needs to be exactly correct. **[Level 4], [T-], [V3], No need**

   LR §3.3. Working Memory and Cognition; FDUCS §10 Errors. Recall memory is slower, harder, and more error-prone than recognition memory. Even expert users are going to make errors at some point, so using recognition memory will reduce the number of errors and constrain errors to be within the known selection list.

2) Simplify navigation of value lists unless there are times when none will apply. Long lists should be sortable and filterable, and all lists should be arranged logically like alphabetical order or grouped by type. **[Level 3], [T], [V2], Yes**

   FDUCS §7.3.4 Scanning Displays and Menus; People tend to scan displays rather than deeply read them and the information should be presented in a scannable way that is sorted according to the operator's mental model.

3) Use introductory labels to describe text entry fields. Support the labels with clear, visible hints placed closely outside the text field. **[Level 3], [T-], [V1], No need.**

   LR §2.2.1 Stage 1 Perception; FDUCS §5.2.4.4 Priming; FDUCS §4.4.6 Pop-Out Effects; FDUCS §7.3 Reading ;NN/g; Labels help users understand what they are looking at and prompt them to begin thinking about the relevant information needed for the task. Also, words are automatically processed for experienced readers so they will pop-out upon being viewed by the user. Also, users read a word faster than naming an icon.

4) Support effective reading and comprehension for text within a text field and long strings of texts like event logs. **[Level 4], [T+], [V2], No need**

   FDUCS §7.3 Reading

   a) Adjust text field line breaks accordingly. By default, any text extending beyond the bounds of a text field is clipped. A text field, however, can be set to wrap text to a new line at the character or word level, or to be truncated (indicated by an ellipsis) at the beginning, middle, or end.

   b) Consider using an expansion tooltip to show the full version of clipped or truncated text. An expansion tooltip behaves like a help tag and appears when the user places the pointer over the field.

5) Let the user adjust text attributes if it makes sense. If your text field contains styled text, it may add value if the user can adjust the font, size, and color of the text. System controlled text attribute changes could be used to instantiate the pop-out effect in event logs. **[Level 2], [T], [V1], No need**

   ADG

6) Get information from the system whenever possible. Don't force users to provide information that can be gathered automatically or with the user's permission. **[Level 4], [T], [V1], No need**

   GOMS; CPMGOMS ;FDUCS §10 Errors: An Inherent part of human-system performance

7) Provide reasonable default values and prefill fields with most likely values when appropriate. **[Level 3], [T+], [V2], Maybe**

   GOMS; CPMGOMS; ADG

8) Dynamically validate field values rather than waiting until submission. This reduces the need to backtrack when data entry fails validation. **[Level 3], [T+], [V3], Yes**

   ADG; NN/g

9) Use proper formatting that connects the input format with user expectations. **[Level 3], [T+], [V2], Yes**

   a) Displaying the input for percentages as a percentage, or automatically presenting phone numbers in their standard format.

   b) Entries expecting long text should allow users to view the input with minimal scrolling (and thus less short-term memory usage).

10) Use numeric data entry, especially for critical features, should follow these guides. **[Level 4], [T+], [V3], Yes**

    Thimbleby, H., & Cairns, P. (2010). Reducing number entry errors: solving a widespread, serious problem. Journal of The Royal Society Interface, 7(51), 1429–1439. https://doi.org/10.1098/rsif.2010.0112; Test case from study was for medication dosages entry to reduce risk of killing patient due to operator error.

    a) Always show commas for values above 1,000.

    b) Don't use 'naked' decimal points: 0.5 is better than .5

    c) Avoid showing trailing zeros for values with whole numbers: 1 is better than 1.0

    d) When possible, build in automatic blocking of invalid numbers.

    e) Maximum stakes data entry fields can reduce risk of failure by using slightly larger decimal points and smaller font for numerals after the decimal.

    f) Batching values in groups of 3

Help & Tooltips [Level 4], [T], [V2], No/Maybe.

Ideally, people can figure out how to use your system without a guide. However, even in a highly intuitive interface, users sometimes need help learning advanced and secondary features. When called for, your program can offer assistance in the form of help tags and other forms of help documentation. Help tags allow you to provide temporary, context-sensitive help, whereas documentation allows you to provide a more thorough discussion of the topic.

Isaksen, H., Iversen, M., Kaasbøll, J., & Kanjo, C. (2017). Methods for Evaluation of Tooltips. In M. Kurosu (Ed.), HCI 2017: Human-Computer Interaction, User Interface Design, Development and Multimodality (Vol. 10271, pp. 297–312). https://doi.org/10.1007/978-3-319-58071-5_23; Mildly testable but expensive. Instead, help and tooltips simply provide a useful. A study on tooltips found that explicit evaluation was costly. Instead, making tooltips should just be assumed.

1) Describe only the control that's directly beneath the pointer.
2) Add help tags to app-specific or system-specific controls. Skip tags on common features like resize controls, scrollers, or others.
3) Focus on the action that a control initiates. A good rule-of-thumb is to start tool tips with a verb.
4) Use the fewest number of words possible.
    a) Try to limit tags to a maximum of 60 or 75 characters depending on your system needs. [Level 2]
    b) Requiring more text to explain a feature may indicate that the interface is overly complicated. [Level 1]
5) In general, don't reference a tag's corresponding control. Typically, the help tag's location (directly adjacent to the control) will provide sufficient context for the user.
6) Use sentence fragments with sentence-style capitalization. This emphasizes brevity without overly sacrificing readability for users.
7) Consider offering context-sensitive help tags.

Keyboard Interactions [Level 4.5], [T+], [V3], Yes

The keyboard is an essential input device for entering text, navigating, and initiating actions. Some users will prefer to almost exclusively use the keyboard for performing some or all tasks.

GOMS, general wide support

1) Respect standard keyboard shortcuts and create program-specific shortcuts for frequently used commands.
2) Add full keyboard access mode support for all custom interface elements.
    a) Full keyboard access mode lets users navigate and activate windows, menus, interface elements, and system features using the keyboard alone.
    b) Tab is an important command to switching between areas and fields.
3) Enable expected shortcuts for standard menu items. Strive for consistency across all applications and systems for common actions.
4) Define new keyboard shortcuts only for things people do regularly.
    a) Unexpected shortcut design can easily confuse users, and tt rarely makes sense to redefine a common shortcut.
    b) The WDS and similar systems could log commands to know which keyboard shortcuts and commands are most common. This would help improve keystroke accelerator generation.
5) Use a standardized hierarchy for assigning modifier keys (i.e., *ctrl*, *alt, shift*) when creating a new shortcut.
    a) Maintain a consistent order using modifiers and writing out commands with modifiers.

6) Provide keystroke accelerators for nearly all commands

7) Keystroke accelerators are displayed in a help screen as a set and on menus and perhaps tool tips.

8) At a convenient time, like starting or stopping or loading or paused, note a keystroke accelerator of the day.

9) Prefer to create "sets" of commands centered around a single action key with multiple modifier keys. For example, *Control-P* may activate the "print" command, and *Shift-Control-P* may activate the "page layout" menu which complements the "print" command.

10) Determine which keyboard shortcuts are common and/or reserved with your system to ensure that your application does not interfere with prior knowledge from the users regarding how to interact with systems of this type.

Providing User Feedback [Level 4], [T-], [V2]

Feedback tells people what an app is doing and helps them understand the results of actions and what they can do next.

FOK; CWT

1) Unobtrusively integrate status and other types of feedback into your interface. If a notification does not provide immediately actionable information, the operator should be able to continue their current task uninterrupted. **[Level 4] [T], [V3], Yes**

LR §3.2.2 Interruptions

2) Avoid unnecessary alerts by carefully assessing whether new information is worth disrupting the operator's current task, so they can address the situation. If deemed important, ensure that the alert is disruptive so ensure the user responds. **[Level 4] [T], [V3], Yes**

LR §3.2.2 Interruptions

3) Warn people when they initiate a task that can cause an unexpected and irreversible loss of data. Avoid being overzealous (i.e., notifications for clearing the recycle bin on desktop), but try to strike a balance between user expectations and task requirements. **[Level 3], [T-], [V1], No**

4) Inform the user when a command can't be carried out. **[Level 3], [T-], [V1], No**

5) Clearly note time constraints for alert triggers, postponing an alert response, and other important tasks. **[Level 4], [T], [V2], Maybe**

6) If it makes sense, allow users to adjust time constraints for how alerts are provided. For example, a user (or supervisor) may wish to make a certain alert type occur more or less often. **[Level 2], [T-], [V1], No**

7) Allow users to set up new alerts when it makes sense. **[Level 3], [T-], [V2], Maybe**

Badging or Icons as Updates [Level 3], [T], [V3], Yes

Icons and other programs can display small, meaningful icons to indicate new, noncritical information like events or minor alerts.

Unstudied other than to note that even common icons only have a 70% recognition rate on average. See Ghayas, S., Sulaiman, S., Khan, M., & Jaafar, J. (2013). The effects of icon characteristics on users' perception. In International Visual Informatics Conference (pp. 652–663).

1) Use badging for notification purposes only for focused, simple information. Avoid using icons as updates for complex, quickly changing information (e.g., air quality or wind speed). **[Level 3], [T+], [V3], Yes**

2) Badging should supplement direct presentation of information within the application. If a badge indicates some alert, that same alert should be presented within the application in text form. **[Level 3], [T-], [V2], Maybe**

3) Ensure badges update quickly in response to user activity such as dismissal or acknowledgement of some alert. **[Level 2], [T-], [V1], No need**

4) Prefer short and/or concrete words (vs. long and/or abstract where these will do, they are faster to read and easier to interpret. The button that says "word" is clearer to ask about than the button "that some squiggly lines that seem to form a point…." **[Level 4], [T], [V2], Maybe**

FDUCS §7.3 How Users Read; Stroop on Automatic Processing of Words

Notifications [Level 3], [T+], [V3], Yes

System notifications provide timely and important information anytime. Notifications may occur when a message arrives, an event occurs, new data is available, or the status of something has changed.

1) Use distinct notification styles to differentiate between minor notifications and alerts. Alerts should remain visible until dismissed by the user while notifications can disappear after a few seconds. **[Level 2], [T+], [V3], Yes**

2) Notifications should be useful and informative: use complete sentences and standard grammatical style, avoid repetitive notifications that clutter the view, and ensure key information (like origin) is clearly displayed. **[Level 4], [T], [V2], Maybe**

FDUCS §7.3 How Users Read

3) If possible, ensure that responses prompted by the notification are not overly specific or difficult to accomplish once the notification is dismissed. **[Level 3], [T-], [V2], Maybe**

LR §3.3 Working Memory and Cognition

4) Adapt notification behavior for different contexts. Consider using cognitive counter-measures to correct behavior in risky situations. **[Level 3], [T+], [V3], Yes**

   a) If the user is on the home page, then a notification about new events may be useful; if the user is already on the event log page then displaying a popup will likely be annoying compared to other methods of informing the user of new information.

   b) Critical events can implement cognitive counter-measures to capture the attention of the operator. Cognitive counter-measures are temporary, major changes to the interface intended to temporarily break their focus, so they will reorient onto the important task. For example, a low battery alert that occurs during manual control of an unmanned vehicle could clear the screen of all features, prominently display the low battery alert until cleared before resuming normal operation. This eliminates the risk of "tunnel-vision" causing the signal to be missed.

   Directly tested for the exact scenario described. Extremely relevant to WDS interface design. Dehais, F., Causse, M., & Tremblay, S. (2011). Mitigation of conflicts with automation: Use of cognitive countermeasures. Human Factors, 53(5), 448–460. https://doi.org/10.1177/0018720811418635

c) Critical events should use dual-coded alerts such as a visual and audio indicator, or multiple visual indicators.

5) Provide intuitive, beneficial action buttons on pop-up notifications and alerts. Limit buttons for user response to 2 buttons if possible. **[Level 3], [T], [V2], Maybe**

a) Use the buttons to perform common, time-saving tasks. This will help reduce how often the operator needs to change views for simple tasks.

Color [Level 4] [T+], [V2], No/Maybe

Color is a great way to provide status information, give feedback in response to user actions, and help people visualize data.

1) Use color judiciously for communication. Limit the number of colors used for communication to less than five. **[Level 3], [T], [V2], Maybe**

ADG; LR §3.1.5 Principle 7

2) Provide adequate support for colorblind users. Colorblindness is common enough that, when possible, designers and engineers should ensure that the standard design supports colorblind users. **[Level 4], [T+], [V2], Maybe**

LR §3.1 Perception

3) Color contrast should be between foreground and background colors should be at least 4.5:1 if not a higher contrast of 7:1. **[Level 3], [T], [V1], No**

ADG.

4) Test the application's color scheme under appropriate lighting conditions. A system used in a brightly-lit room will have different requirements than one used in a dark room. **[Level 4], [T+], [V2], No**

LR §1.4

Visual feature index

Most applications should be built using components from your preferred graphic design kit like Java Swing or others. This will provide a programming framework that defines common interface elements. This framework lets applications achieve a consistent appearance across the system, while at the same time offering a high level of customization. The following interface elements are a common set of flexible and familiar features that can provide a design framework for building nearly any system.

Windows and Views

Alerts

An alert appears when the system or program needs to warn the user about an error condition, or a potentially hazardous situation or consequence. A major alert should be modal within an application; once the alert is received, the program is locked into an "alert response" mode that requires user input regarding the alert before enabling any other actions. Minor alerts should be displayed differently than major alerts.

1) Minimize alerts. Alerts disrupt the operator and should be reserved for important situations. The infrequency of alerts helps ensure that operators take them seriously. **[Level 3], [T+], [V3], Yes**

2) Ensure that each alert offers critical information and useful choices. **[Level 3], [T], [V2], Yes**

   a) Avoid using alerts to merely provide information.

   b) Users become annoyed at alerts and interruptions that don't provide actionable information.

   c) Avoid displaying alerts for common, undoable actions.

3) Use a standardized alert display. Consistency will help users understand the meaning of the alerts by supporting learned responses to different alert displays. **[Level 4], [T], [V3], Maybe**

   Rieman, J., Young, R. M., & Howes, A. (1996). A dual-space model of iteratively deepening exploratory learning. International Journal of Human Computer Studies, 44(6), 743–775. https://doi.org/10.1006/ijhc.1996.0032

4) Provide a clear, succinct alert message that gives the user what, why, and where for a given alert. **[Level 2], [T], [V2], Maybe**

   ADG

   a) Consider phrasing a message as a question when the default action has negative consequences.

   b) Supplement alert messages with informative text. Use this space to elaborate on consequences, suggest solutions, and explain why the user should care.

5) Avoid using alert buttons that require explanation. **[Level 3], [T-], V1], No**

   ADG; LR §2.2.2 ;CWT

   a) If the text and button titles are clear, there should be no need to explain the buttons.

   b) If guidance is needed, preserve capitalization when referencing buttons and don't enclose button titles in quotes.

   c) Give alert buttons succinct, logical titles. Best titles will use one- or two-word verb phrases that describe the result of clicking the button. Avoid using "yes and no" as the options.

   d) Label cancellation buttons appropriately.

   e) Include a Cancel button when there's a destructive button (i.e., delete file)

6) Generally, prefer two-button alerts. Single button alerts inform but give no control; alerts with three or more buttons create complexity. **[Level 2], [T], [V2], Maybe**

   ADG

7) Ensure that the default button title reflects the action the button performs. Avoid using OK unless the alert is purely informational. Specific button titles like Erase, Convert, Clear, or Delete help users understand the action. **[Level 3], [T-], [V1], Maybe**

   CWT; ADG

8) Place buttons where people expect them. In general, the default (or most likely) button should be on the right. Cancel is usually on the left. **[Level 2], [T], [V1], No**

   ADG; Others

9) Consider offering time-saving keyboard shortcuts for all buttons. For example, Enter (or return) can a default "Accept" button for situations are not high stakes. Clearly indicate defaults by using bold, underlined text on the default choice. **[Level 2], [T], [V2], Maybe**

Boxes [Level 2], [T-], [V1], No

A box is a type of view that creates distinct, logical groupings of controls, text fields, and other interface elements. For example, a preferences window may include boxes that visually group related settings together. By default, a box has a border and a title, either of which can be disabled if it makes sense for your sub-display. The title, if displayed, can be positioned above (the default) or below the box.

1) Avoid nesting boxes. Nested boxes waste space and reduce the effectiveness of boxes overall for grouping information.

   ADG

2) Use sentence-style capitalization in box titles. Don't end box titles with a colon.

   APA guidelines; FDUCS §7.3 How Users Read

Dialogs

A dialog is a type of window that elicits a response from the user. Many dialogs—like the Print dialog, for example—let people provide several responses at once. Dialogs are presented in three ways: *document-modal*, *app-modal*, and *modeless*.

A document-modal dialog is attached to a document as a sheet and prevents the user from doing anything in the document until the dialog is dismissed. The user can still switch to other documents and apps. A Save dialog is an example of a document-modal dialog.

An app-modal dialog prevents the user from doing anything in the app until the dialog is dismissed. The user can still switch to other apps. An Open dialog is an example of an app-modal dialog.

A modeless dialog is usually referred to as a *panel*. The user can continue interacting with documents and apps uninterrupted. The standard Find dialog is an example of a modeless dialog.

Data Entry for Dialogs

Dialogs are intended to be small, transient windows that don't require in-depth user interaction, so it's important to ensure that data entry is efficient.

1) Provide default values for controls and fields whenever possible. **[Level 5]**

2) Set the initial focus to the first location that accepts user input. **[Level 5]**

3) Make static text selectable. For example, users may want to copy an error message or IP address.

4) Check for errors during data entry. The best time to check is immediately after the user moves onto the next field. Waiting until they hit the submit button can annoy the user.

5) Whenever possible, minimize the potential for invalid input.

Layout

6) Use disclosure control to provide information or functionality that's only occasionally needed.

7) Position buttons as expected. **[Level 1]**

   a) Buttons in the bottom right of a dialog should dismiss the dialog

   b) An action button, which initiates the dialog's primary action, should be farthest to the right.

c) A cancel button should be to the immediate left of the action button.

d) If a third button is needed, it should be to the left of the cancel button.

e) If a help button is shown, it should be the furthest left button.

8) Separate destructive buttons from nondestructive buttons.

a) For example, Don't Save should be far enough away from Save to ensure accidents are rare.

b) Destructive buttons should require intentional effort.

c) Ideally, 24 points of separation is best.

Dialog Dismissal

9) Provide a default button only when the user's most likely action is harmless. Users may simply hit Return (or Enter) to dismiss an alert or dialog. This should never trigger an important event. If it's important enough, they should have to select a response.

10) Provide a default button only when the Return key isn't already used by text fields on the dialog.

11) Include a Cancel button that responds to the standard cancellation keyboard shortcuts. A Cancel button provides a clear, safe way out of the dialog and returns the computer to its previous state.

12) Ensure the Cancel button undoes all applied changes.

Outline View [Level 3], [T+], [V3], Yes/Maybe

An outline view presents hierarchical data—like folders and the items they contain—cleanly and efficiently in a scrolling list of cells that are organized into columns and rows. At minimum, an outline view includes one column that contains the primary hierarchical data: parent containers and their children. Subsequent columns may be added, as needed, to display additional attributes that supplement the primary data. Event logs could be presented in outline view as an alternative to the typical table view.

1) Outline View should be used for hierarchical data whereas Table View should be used for non-hierarchical data. Event logs have some underlying hierarchical traits, but presentation style should depend on the task being performed. **[Level 3], [T], [V3], Maybe**

Bakke, E., Karger, D. R., & Miller, R. C. (2013). Automatic layout of structured hierarchical reports. IEEE Transactions on Visualization and Computer Graphics, 19(12), 2586–2595. https://doi.org/10.1109/TVCG.2013.137; The outline view is just one way to present data. There could be valuable testing done on how best to present complex sets of events from the WDS and other systems based on the mental model of the user.

2) The data hierarchy structure should be viewable within the first column only. **[Level 1] [T], [V1], No**

ADG

3) If deemed appropriate, operators should be able to click column headings to sort an outline view. Clicking again should sort the column in the reverse-order of the initial click. **[Level 2], [T], [V1], No**

ADG

4) Support ease-of-use by providing clear, noun-based column headings, allowing operators to resize columns, and ensuring that rows are easily distinguished. **[Level 2], [T], [V1], No**

5) Long text strings within a cell should be truncated in some way. This can be done with an ellipsis in the middle, with the ends unaffected, or with a trailing ellipsis that prioritizes early text. **[Level 3], [T+], [V2], Yes**

6) Search fields should be provided to allow operators to quickly find specific items. **[Level 3], [T+], [V3], Yes**

Panels

A panel is an auxiliary window containing controls, options, or information related to the active document or selection. A panel appears less prominent than a main window and can behave like a normal window or be configured to float above other open windows—even modal windows. Panels can also adopt a darker, translucent (HUD-style) appearance when the user experience calls for it.

1) Use a panel to provide quick access to important controls or information related to content.
2) As an alternative to panels, you could also implement Popovers, Sidebars, Split Views, or a Toolbar
3) Title panels with appropriate text that describes the purpose with nouns or noun phrases.
4) Link the visibility of a panel to whether the associated application is currently active. Inactive applications should shouldn't have visible panels.
5) Consider using HUD-Style panels for highly visual content.
    a) HUD Panels are translucent and typically have a darkened background.
    b) Use simple controls and interactions for HUD panels. Avoid making the user type, for example.
    c) Keep HUD Panels fairly plain with minimal color and other distracting features.

Popover [Level 2], [T-], [V1], No

A popover is a view that appears above other content onscreen when you click a control or view. Popovers typically integrate an arrow pointing to its origin. Popovers can close in response to a user interaction (transient behavior), in response to a user's interaction with the view or element from which the popover emerged (semi-transient behavior). A popover can also be made detachable. A detachable popover becomes a separate window when dragged by the user, allowing it to remain visible onscreen while the user interacts with other content.

ADG

1) Popovers are for limited information or functionality and typically disappear following user interaction. Avoid using popovers for complex tasks and functions.
2) Use popovers to streamline interfaces by moving simple interactions from static regions into context dependent popover views.
3) Popover behavior should be intuitive based on the popover's function.

    NN/g

    a) Typically, this means exiting automatically after completing a task or clicking outside the popover rather than requiring a Close button.
    b) Ensure Popovers don't obscure the screen element that caused it to appear.
    c) Only display a single Popover on the screen at one time.

Scroll view [Level 3], [T+], [V2], Yes/Maybe

A scroll view lets people browse content (i.e., a large event log) that's larger than the view's visible area. A scroll view itself has no appearance, but can display horizontal and vertical scroll bars, each of which consists of a track containing a draggable control known as a knob. The height of a knob reflects the quantity of scrollable content.

1) Don't have nested scrolling views. **[Level 2], [T], [V1], No**

2) Ensure scroll bars and sliders have distinct appearances. **[Level 2], [T], [V1], No**

3) Avoid requiring horizontal and vertical scrolling on the same interface and prefer vertical scrolling over horizontal. **[Level 4], [T+], [V3], Yes**

    GOMS; Bakke, E., Karger, D. R., & Miller, R. C. (2013). Automatic layout of structured hierarchical reports. IEEE Transactions on Visualization and Computer Graphics, 19(12), 2586–2595. https://doi.org/10.1109/TVCG.2013.137; Event logs are complex sets of data that need searched by users and determining the best way to present them could be valuable.

4) If possible, avoid requiring the use of scrolling to view all content. This must be balanced against over crowding an interface. **[Level 4], [T+], [V3], Yes**

    GOMS; LR §3.3 Working Memory and Cognition; Scrolling requires the user to store more information in working memory rather than "maintaining" that information on the screen.

Split view

A split view manages the presentation of two or more panes of content. Each pane can contain any variety of elements, including buttons, tables, column views, text fields, and even other split views. The panes of a split view can be arranged horizontally or vertically and are separated by a divider that can typically be dragged to resize the panes. Each pane can have a minimum and maximum size, which affects how much it can be resized. Many apps let the user hide specific panes on request.

ADG

1) Allow panes to be hidden when it makes sense. For example, hiding a pane may help reduce distractions during focused work.

2) Provide multiple ways to access hidden panes. Provide toolbar buttons or menu items with keyboard shortcuts.

3) Ensure minimum and maximum pane sizes set based on the system's requirements and functions.

4) Use Thin dividers (1pt width) for most dividers. If the designer wants to indicate a stronger visual distinction between panes then use a Thick divider (9pt width).

Tab Views [Level 3], [T], [V2], No/Maybe

A tab view presents multiple mutually exclusive panes of content in the same area. A tab view includes a tabbed control (which is similar in appearance to a segmented control) and a content area. Each segment of a tabbed control is known as a tab, and clicking a tab displays its corresponding pane in the content area. Although the amount of content can vary from pane to pane, switching tabs doesn't change the overall size of the tab view or its parent window.

1) Use a tab view to present closely related, equally important content areas. **[Level 2], [T-], [V1], No need**

2) Provide between two and six tabs in tab view. If more tabs are necessary, consider alternative views. **[Level 2], [T], [V2], Maybe**

    ADG

3) Controls within a pane using tab view should only affect content within that tab. **[Level 2], [T], [V1], No need**

ADG; CWT

4) Provide a label for each tab that describes the content of its pane. **[Level 2], [T-], [V1], No need**

5) Ensure switching between tabs requires only a single action like pressing a button, using a keystroke (e.g., tab), or clicking. **[Level 3], [T+], [V1], No need**

ADG; GOMS

Menus [Level 3], [T+], [V2], Yes/Maybe

A menu presents a list of items—commands, attributes, or states—from which a user can choose. An item within a menu is known as a menu item, and may be configured to initiate an action, toggle a state on or off, or display a submenu of additional menu items when selected or in response to an associated keyboard shortcut. Menus can also include separators, and menu items can contain icons and symbols, like checkmarks.

CWT; GOMS; ADG

1) Use title-style capitalization for all text. **[Level 2], [T], [V1], No need**

APA Guidelines; ADG

2) Ensure menu titles are intuitive so users will anticipate the types of items the menu contains. **[Level 4], [T], [V2], Maybe**

ADG; NN/g; CWT; Information scent research

3) Keep menus enabled, even when menu items are unavailable. **[Level 3], [T], [V2], Maybe**

ADG; CWT; Mendel, J., & Pak, R. (2009). The Effect of Interface Consistency and Cognitive Load on User Performance in an Information Search Task. Proceedings of the Human Factors and Ergonomics Society Annual Meeting, 53(22), 1684–1688. https://doi.org/10.1177/154193120905302206

   a) This tells users that a particular function is unavailable at the moment.

   b) Unavailable menu items also allow users to learn about other functions in the system, even if the actions aren't possible.

4) Make menu titles as short as possible without sacrificing clarity. **[Level 3], [T], [V1], No need**

FDUCS §7.3 How Users Read; ADG

5) Only use text for menu items. Icons are confusing and unnecessary. **[Level 3], [T], [V1], Maybe**

FDUCS §7.3 How Users Read; Ghayas, S., Sulaiman, S., Khan, M., & Jaafar, J. (2013). The effects of icon characteristics on users' perception. In International Visual Informatics Conference (pp. 652–663).

6) Ensure the menu titles and text make sense according to their function. **[Level 2], [T], [V2], No**

ADG; NN/g

   a) Use verbs and verb phrases for menu items that initiate actions.

   b) Use adjectives and adjective phrases for menu items that toggle attribute states.

   c) Avoid articles in menu item titles.

7) Use keyboard shortcuts for frequently used items in the menu bar. Make sure keyboard shortcuts are shown next to the functions. **[Level 4.5], [T+], [V3], Yes**

8) Avoid using submenus when possible. **[Level 4], [T+], [V2], Maybe**

FDUCS §7.3.4

a) If necessary to include a submenu, only have a single additional level to the menu.

b) Avoid having more than five items in a submenu.

c) Only consolidate related menu items into submenus. For example, Sort By Name, Sort By Date, and Sort By Length could be merged into a single command Sort By with a submenu for Date, Name, and Length.

9) Group items within a menu in a logical manner. **[Level 4], [T+], [V2], No**

FDUCS §7.3.4; GOMS; St. Amant, R., Horton, T. E., & Ritter, F. E. (2004). Model-based evaluation of cell phone menu interaction. Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, 6(1), 343–350. https://doi.org/10.1145/985692.985736

a) Group closely related items together (Find and Find Next)

b) Arrange sets of closely related items by frequency of use. Put frequently used items at the top of the list.

c) Separate items that initiate actions from items that set attributes.

10) Avoid scrolling menus. [Level 4], [T+], [V3], Yes

GOMS; LR §3.3 Working Memory and Cognition; Scrolling requires the user to store more information in working memory rather than "maintaining" that information on the screen.

11) If icons are necessary for your menus (like for a toggled setting), use a standard, limited set of clear symbols like a checkmark. **[Level 2], [T], [V2], Maybe**

ADG

Contextual menus

A contextual menu, or shortcut menu, gives people access to frequently used commands related to the current context. Contextual menus are typically brought up by using a right-click on the item. Contextual menus often provide a limited set of useful actions that are frequently used in a particular situation.

1) Follow the standards and best practices of typical menu design

2) Include only the most commonly used commands that are appropriate in the current context.

3) Always make contextual menu items available in the menu bar as well.

Buttons

Checkbox [Level 3], [T], [V1], No

A checkbox is a type of button that lets the user choose between two opposite states, actions, or values. A selected checkbox is considered on when it contains a checkmark and off when it's empty. A checkbox is almost always followed by a title unless it appears in a checklist.

ADG; Tufte

1) Ensure the label or title implies two opposite states. If the titled/labeled checkbox is difficult to make unambiguous, consider using two binary-titled radio buttons instead.

2) Checkboxes should be within a view, not a window frame (i.e., toolbars and status bars).

3) Consider using a label for describing a set of several checkboxes if their relationship isn't evident.

4) Checkboxes should usually be arranged vertically.

5) Checkboxes can use a hierarchical arrangement with indentation to show relationships between parent and child checkboxes.

6) Parent checkboxes should use a mixed state [-] if the child checkboxes have mixed settings.

Gradient button

A gradient button initiates an immediate action related to a view, like adding or removing table rows. Gradient buttons contain icons—not text—and can be configured to behave as push buttons, toggles, or pop-up buttons. They usually reside in close proximity to (either within or beneath) their associated view.

ADG

1) Gradient buttons only below in views, not in window frames.

2) Use standard system-provided icons for gradient buttons to ensure users are familiar with the symbols and meaning.

3) Gradient buttons should be clearly linked to a particular view and shouldn't need a label.

Help button [Level 3], [T+], [V3], Yes

A help button appears within a view and opens application-specific help documentation when clicked. All help buttons are circular, consistently sized buttons that contain a question mark icon.

1) Use system-provided help buttons and ensure the help buttons have a consistent response.

2) Only include one help button per window.

3) Position help buttons as expected

   a) Dialog with dismissal buttons (e.g., OK and Cancel): lower-left corner aligned with dismissal buttons.

   b) Dialog without dismissal buttons: Lower-left or lower-right corner.

   c) Preference window or pane: Lower-left or lower-right corner.Pop-up buttonA pop-up button (often referred to as a pop-up menu) is a type of button that, when clicked, displays a menu containing a list of mutually exclusive choices. A pop-up button includes a double-arrow indicator that alludes to the direction in which the menu will appear. The menu appears on top of the button. Like other types of menus, a pop-up button's menu can include separators and symbols like checkmarks. After the menu is revealed, it remains open until the user chooses a menu item, clicks outside of the menu, switches to another app, or quits the app; or until the system displays an alert.

Push Buttons [Level 2.5], [T-], [V1], No

A push button appears within a view and initiates an instantaneous app-specific action, such as printing a document or deleting a file. Push buttons contain text—not icons—and often open a separate window, dialog, or app so the user can complete a task.

ADG; NN/g

1) Design the options to ensure a likely default button is clear.
2) Push buttons should only be in views, not window frames.
3) Only use text for push buttons, not icons.
4) Give push buttons clear labels with verbs to describe the effect of clicking the button.
5) Be specific when possible. “Select Text File” is much clearer than “Import.”
6) Include a trailing ellipsis in the title when a push button opens another window, dialog, or application.
7) Push buttons should be similar in size (when appropriate) for aesthetics and clarity.

Radio button [Level 2.5], [T-], [V1], No

A radio button is a small, circular button followed by a title. Typically presented in groups of two to five, radio buttons provide the user a set of related but mutually exclusive choices. A radio button’s state is either on (a filled circle) or off (an empty circle). A radio button can also permit a mixed state (a circle containing a dash) that’s partially on and partially off. However, it’s better to use checkboxes when your app requires a mixed state.

ADG; NN/g; GOMS; General support from work on visual scanning

1) Ensure radio buttons have meaningful titles.
2) Use a standard button instead of a radio button if initiating an action.
3) Radio buttons should only be used in views, not window frames.
4) Labels can help clarify the connection between a set of radio buttons.
5) Avoid horizontally placed radio buttons, but if necessary then use consistent spacing.
6) If more than five choices are necessary, consider using a pop-up button instead.
7) In almost every case, select a radio button by default. Default buttons reduce confusion and can allow engineers to imply the best course of action to the user.

Fields and Labels

Combo box

A combo box combines a text field with a pull-down button in a single control. The user can enter a custom value into the field or click the button to choose from a list of predefined values. When the user enters a custom value, it’s not added to the list of choices.

ADG; NN/g; CWT; LR §2.2.2; Rieman, J., Young, R. M., & Howes, A. (1996). A dual-space model of iteratively deepening exploratory learning. International Journal of Human Computer Studies, 44(6), 743–775. https://doi.org/10.1006/ijhc.1996.0032

1) Populate the field with a meaningful default value from the list.

2) Use an introductory label to let the user know what types of items to expect.

3) Provide useful, relevant choices for the user to select. Ensure that the options are all standalone selections because combo boxes shouldn't allow multiple selection.

Labels [Level 3.5], [T], [V2], Yes

A label is a static text field that describes an onscreen interface element or provides a short message. Although people can't edit labels, they can sometimes copy label contents.

1) Ensure labels are legible, clear, and consistent **[Level 3], [T], [V2], Maybe/No**

   a) Typically labels for controls should end with a colon. An exception to this rule is when the label and control form a complete sentence.

   b) Use system-provided, standardized label colors to communicate relative importance.

2) Make sure label text is selectable where possible, and make logs copiable so users can copy useful text onto other locations. **[Level 3.5], [T+], [V2], Yes/Maybe**

   ADG; GOMS; CWT; LR §3.3 Working Memory and Cognition

3) Labels and other text must use a consistent vocabulary, syntax, and grammar. Even minor changes can have a negative impact on the mental model and understanding of the user. **[Level 4], [T+], [V3], Yes**

   Mendel, J., & Pak, R. (2009). The Effect of Interface Consistency and Cognitive Load on User Performance in an Information Search Task. Proceedings of the Human Factors and Ergonomics Society Annual Meeting, 53(22), 1684–1688. https://doi.org/10.1177/154193120905302206

4) If users will be exposed to many labels at once, use colors and icons to help differentiate items for faster, more accurate search. **[Level 4.5], [T], [V3], No need/Maybe**

   NN/g; https://www.nngroup.com/articles/visual-indicators-differentiators/

Search field [Level 3], [T+], [V3], Yes

A search field is a style of text field optimized for performing text-based searches in a large collection of values. Many windows include a search field in the toolbar, but a search field can also be displayed in the body area of a window. A search field typically displays a magnifying glass icon and can also include placeholder text and a cancellation button.

ADG; NN/G; CWT; Others

1) Ensure search fields have a distinct look that users can instantly recognize and distinguish from other similar features like text fields. **[Level 3], [T], [V1], Maybe**

   ADG; Consistency and Cognitive Load on User Performance in an Information Search Task. Proceedings of the Human Factors and Ergonomics Society Annual Meeting, 53(22), 1684–1688. https://doi.org/10.1177/154193120905302206

   a) Placeholder text can remind users of the types of information are searchable.

2) Determine an appropriate time to begin searching. Consider whether to show search results dynamically or only after the user initiates the search. **[Level 3], [T+], [V2], Yes/Maybe**

   ADG; NN/g; See: https://www.nngroup.com/articles/suggested-employee-search/

3) Scope bars, a type of toolbar for filtering searches, will help users trim down unnecessary information during searches that may bring up large amounts of data. **[Level 2.5], [T], [V2], Maybe**

   ADG; NN/g; CWT

a) Plan scope bar functions around the tasks. Searching documentation for a page might not need detailed search filters, however searching an event log with 1000s of entries may require users to input multiple filters.

b) Some general useful filters for event logs include date range, module origin, text, and severity.

c) More advanced or specialized filters could include number of results shown, reverse filters, and options for pre-set filter categories (e.g., alarms from past 24 hours from only core modules).

d) Include a "not" function for searches to support more detailed searching behavior.

4) Searches with no results found should be clearly communicated to the operator. **[Level 3], [T+], [V1], No**

ADG; LR: §2.2.1 Stage 1 – Perception

5) Filtering for date ranges should have multiple input methods like text-view and calendar-view. **[Level 2], [T+], [V2], Maybe**

ADG; NN/g

6) Ensure that date formats are clear

Text/Character field [Level 3], [T], [V2], Yes

A text field is a rectangular area in which the user enters or edits one or more lines of text. A text field can contain plain or styled text. Text fields are the base category for search fields, labels, and other related features.

ADG; NN/g

1) When providing a user-provided data entry field, use a clear label with useful hints close by to communicate the purpose of the text field. **[Level 2], [T], [V1], No**

LR §3.3 Working Memory and Cognition; Disappearing placeholder text can strain working memory, particularly when distracted.

2) Perform field validation after the user finishes typing into the field. Don't wait until the user tries to submit the data. **[Level 3], [T], [V2], Yes**

ADG; NN/g; Others; The value of this is dependent on what is being typed. For numerical entry, this is more important.

3) Number formatters help users provide accurate numerical data by making the text easier to read and comprehend. **See Data Entry above.**

4) Ensure that text fields allow users to easily view the full content in the field. Consider enabling resizing of text fields or providing another method to view the full text. **[Level 4], [T], [V1], Maybe**

LR §3.3 Working Memory and Cognition; Being unable to view the full text field forces operators to store information within working memory rather than simply view it if they want the full picture.

5) When possible, match the size of the text field to the expected size of the input. A text field for a five-digit zip code can be static and just slightly wider than the text. A text field for paragraph-length entries should show (at the very least) multiple lines and potentially include a method for resizing the text field. **[Level 3], [T], [V1], No**

6) A page with multiple text fields should ensure the layout is clean and clear. **[Level 4], [T], [V1], Maybe**

Mendel, J., & Pak, R. (2009). The Effect of Interface Consistency and Cognitive Load on User Performance in an Information Search Task. Proceedings of the Human Factors and Ergonomics Society Annual Meeting, 53(22), 1684–1688. https://doi.org/10.1177/154193120905302206

a) Evenly space multiple text fields.

b) Prefer a vertical layout over horizontal.

c) Prefer consistent text field widths when appropriate. This can be used to signal relationships between text fields. For example, "first name" and "last name" can be one width while the "address" and "city" fields can be another width.

7) Ensure that "tabbing" between fields follows a logical, intuitive path. **[Level 3], [T], [V1], Maybe**

FOK; Mendel, J., & Pak, R. (2009). The Effect of Interface Consistency and Cognitive Load on User Performance in an Information Search Task. Proceedings of the Human Factors and Ergonomics Society Annual Meeting, 53(22), 1684–1688. https://doi.org/10.1177/154193120905302206

8) Provide access to an "other" option when the task is complicated. This provides users a method for completing the task when the options don't align exactly. **[Level 2], [T], [V1], Maybe**

ADG; Consistency and Cognitive Load on User Performance in an Information Search Task. Proceedings of the Human Factors and Ergonomics Society Annual Meeting, 53(22), 1684–1688. https://doi.org/10.1177/154193120905302206

Date/Time picker [Level 3], [T], [V3], Yes

A date picker lets the user choose a date, a time, a date and time, or a range of dates. Date and time can be presented in a textual format using text fields, as a graphical format using a calendar view and/or clock view, or as a display showing both at once.

KLM; CWT

1) Ensure that the formatting of time and date displays matches the needs of the user and system.

2) The date and time format should be consistent across the system (or all systems).

3) Ensure the detail shown by the display matches the needs of the task. Scheduling an in-person meeting requires less precision than scheduling access to a super computer.

4) Present dates and times in a familiar format for the user. Ensure that cultural and international differences are considered during the design.

Segmented control

A segmented control is a horizontal set of two or more segments, each of which functions as a button—usually configured as a toggle. Segmented controls provide closely related choices that affect an object, state, or view. Like buttons, segments can contain text or icons. A segmented control can enable single choice or multiple choices.

ADG; LR §2.2.2 Stage 2 – Comprehension

1) In general, try to keep segment size consistent.

2) Consider using labels to add clarity. Labels can introduce a segmented control, clarify its purpose, and help ensure that icons are understood by the user.

3) Segmented controls should follow the toolbar design guidelines when possible.

4) Segmented controls should not be used as a replacement for tab view controls within a primary window. Segmented controls can be used for view switching within a toolbar or inspector pane, however.

5) Segmented controls should not be used for Add or Remove actions. Instead, use gradient buttons.

6) Segmented control labels should use nouns or noun phrases.

7) Segmented controls that use text within the control don't need an additional label, however icons should be accompanied by labels.

8) Avoid including text and icons within a single segmented control.

Level Indicators [Level 3.5], [T+], [V3], Yes

A level indicator graphically represents of a specific value within a range of numeric values. It's similar to a slider in purpose, but more visual and doesn't contain a distinct control for selecting a value—clicking and dragging across the level indicator itself to select a value is supported, however. A level indicator can also include tick marks, making it easy for the user to pinpoint a specific value in the range. A capacity indicator illustrates the current level in relation to a finite capacity. Capacity indicators are often used when communicating factors like disk and battery usage.

1) The fill color for capacity indicators should be used to alert users about significant values like low battery or low disk space.

2) Large ranges of data should use continuous indicators and tick marks to provide additional information about the data value.

3) Use the quantity and width of discrete indicators to convey additional context information to the user. Don't use tick marks on discrete indicators since they already include that information in their display.

4) Be sure to label at least the first and last tick marks if they are used on a continuous indicator.

Progress Indicators [Level 4], [T], [V2], No

Don't make people sit around staring at a static screen waiting for your app to load content or perform lengthy data processing operations. Use progress indicators to let people know your app hasn't stalled and to give them some idea of how long they'll be waiting.

There are two general kinds of progress indicators: bar indicators and spinning indicators. Bar indicators (or progress bars) use a horizontal bar that fills from left to right to show the progress of some action. Spinning indicators use a circular form to show progress through filling the circle as progress continues.

ADG; Ghafurian, M. (2017). Impatience in dynamic decision-making: Its moderation, and implications for user interface design. The Pennsylvania State University.

1) Progress indicators should only be shown within a view, not in window frame areas like toolbars and status bars.

2) Progress indicators should be in consistent locations across the system.

3) If possible and useful, allow users to halt processing for an action without causing negative side effects.

4) Only use determinate progress indicators for tasks with well-defined durations. Be sure to differentiate between processes that have a determinate length and processes that have an indeterminate length.

5) Always report progress accurately. Users will be frustrated by a progress bar that does not represent the progress in a useful, accurate manner. For example, avoid making a progress bar that jumps to 90% completion within the first 10 seconds, but takes five minutes to complete the final 10% of the task.

6) Hide determine progress indicators once they are completely filled, however make sure the user realizes that the task is complete. If it disappears too quickly, they may wonder if that task was actually complete.

7) Labels for progress bars can provide useful context about the current state of the system. Use a trailing ellipsis on labels to indicate that the task is an ongoing process.

8) Spinning progress indicators should be used to communicate the status of a background operation or to save space on the screen.

9) In general, determinate progress indicators are preferred over indeterminate indicators.

10) Don't switch between spinning indicators and progress bars for the same task.

11) Try to keep indeterminate progress bars in motion to ensure that the user knows that something is happening. This prevents users determine whether the task is progressing or if the system has stalled.

12) Spinning progress indicators typically won't need labels.

For designers

Guidelines will not cover all decisions

Guidelines cannot cover all instances. There may be edge cases or places where unexpectedly questions arise about design. For example, another item to add, another task to add, a different type of screen or user. The implementer will often be asked to make short-term, rapid design decisions without the necessary time or resources to fully analyze the situation. For example, a customer may determine that the power module requires a view showing power over time in addition to the current power level. Should the power-over-time view be shown in addition to the current power level or merged into a single view? Should the power-over-time view change the line's color to show low power alerts or use a horizontal threshold line instead? Providing implementers, designers, and engineers with additional training will allow them to make good design decisions throughout the design process.

Even design guidance and even designs will not always provide enough information to implement a system. Better systems are built when the implementer is at least sympathetic to and perhaps even has studied a bit about the domain they are implementing. In the same way that architects that understand how buildings are built provide better and easier to build building, and architectural engineers build betters buildings if they have studied architecture, engineers that understand their users will build better interfaces.

Study the user [Level 4, FDUCS, PM]

Thus, interface implementers should study the user slightly to be prepared for when explicitly or implicitly, decisions have to be made while implementing the interface. This might be 10-25 hours a year.

Study design [Level 4, FDUCS, PM]

Interface design and implementation is a process and procedural skill like any engineering discipline, similar to writing code or writing English or even medical practice. Professionals in this area should get continuing education in the process of design. This might be 10-25 hours a year.