Chapter 1
Introducing Interface Design for Remote Autonomous Systems

Abstract This chapter presents a high-level overview of how designers of complex systems can address risks to project success associated with operator performance and user-centered design. Operation Centers for remote, autonomous systems 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 than engineering components. Operators also typically exhibit a wider range of performance than other system components. We propose the Risk-Driven Incremental Commitment Model as the best guide to decision-making when designing interfaces for high-stakes systems. Designers working with relevant stakeholders must assess where to allocate scarce resources during system development. By knowing the technology, users, and tasks for the proposed system, the designers can make informed decisions to reduce the risk of system failure. This chapter introduces key concepts for informed decision-making when designing operation center systems, presents an example system to ground the material, and provides several broadly applicable design guidelines that support the development of user-centered systems in operation centers.

1.1 Introduction

Our increasingly complex society relies on an interconnected network of systems, each responsible for carrying out its own role effectively. The most important components within the systems of systems are called critical systems. Critical systems are defined by the cost of their failure; critical systems are called as such because their failure will lead to loss of life, destruction of the system, or failure for the organization as a whole. For example, failure in central command for the space missions may leave astronauts without the information (and oxygen!) they need if their oxygen tank were to fail a few days into the mission. Air traffic control is another example of a critical system; even minor mistakes can have devastating consequences. Not every critical system, however, needs to be part of a large international organization. A 911 emergency call center is responsible for triaging calls,

J. D. Oury, F. E. Ritter, *Building Better Interfaces for Remote Autonomous Systems*, Human–Computer Interaction Series,
https://doi.org/10.1007/978-3-030-47775-2_1

dispatching appropriate services, and providing support for the caller; loss of the call center means local fire, medical, and police services lose their ability to coordinate and respond.

Whether it's NASA's Christopher C. Kraft Jr. Mission Control Center in Houston, the Indianapolis Air Route Traffic Control Center, or a local 911 dispatcher, these critical systems all contain some form of an operation center at the heart of their operation, and these operation centers are vital communication hubs for the transfer of information. Within any given op center, there are going to be different stakeholders, tasks, and priorities that must be considered in their design. A single room or even a single screen could be the link between the op center and multiple complex systems. Figure 1.1 shows a montage of the types of system components this book addresses. This book primarily examines operation centers that manage remote, autonomous, asynchronous systems.

The book is designed to be useful to managers, designers, and implementors of op centers. Managers can use it to adjust their process to account for a wider range of risks caused by failing to support their users and their tasks. Designers can use it to manage the process, learn about users, and become more aware of useful types of shared representations. Implementers can use it to provide context for seemingly small decisions within an interface that are too minor to be described formally or have not been specified. Where we can, we also identify design principles and aspects of the operator, interface, or process that suggest prescriptive actions to create better interfaces.

This introductory chapter makes the case for including knowledge about users as part of the system and design process. It will then briefly describe a way to include this knowledge (the Risk-Driven Spiral Model) and how this knowledge could be applied to operation centers. The rest of the book 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 used as an executive summary or as 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 that shows how the guidance is applied. Table 1.1 defines some common terms used throughout this book.

The design approach that results from this book will be primarily a human–computer interaction (HCI) approach to make the system usable. Aspects of improving the system through user-centered design (UCD) and making the system more enjoyable (while maintaining usability) with user experience (UX) design will be included as well.

1.2 The Role of Operators

Operators can greatly influence operation center success. In a study of errors in air traffic control, a type of op center, Jones and Endsley (1996) found that seven out of ten times system failures are due to operator error. Their error analysis for

Fig. 1.1 Technological advancement has expanded our ability to use and control complex systems in new ways and from new locations. To make full use of these powerful new systems, usability is paramount. (Image by Kenan Zekić)

Table 1.1 Common terms and definitions

Term	Definition
Operation center (op center)	A centralized location used to monitor and exert control over a system, situation, or event. Can sometimes be used interchangeably with command center or control room
Human–computer interaction (HCI)	A broad term for research into the design and use of computer technology, particularly as it relates to human–machine interactions. HCI typically includes user-centered design and user experience design under its purview
User-centered design (UCD)	A design process focused on fitting the goals, tasks, and needs of the user to support optimal performance for the overall human–machine system
User experience design (UX)	A design process that extends HCI to include all design aspects that are perceived and felt by the user to build systems that are desirable to use in both function and experience

aviation disasters organized the contributing errors by operators using Endsley's (1995) theory of situation awareness. The situation awareness framework predicts operator performance by rating the operator's awareness of necessary information. 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 of air disasters about 75% of the time. Going up in complexity, failing to successfully comprehend the meaning or the importance of information was the primary cause in only about 20% of air disasters. Finally, at the lowest error rate, projection into near-future system states is the key in less than 5% of disasters. Breaking down these failures into more specific types of failure showed 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 events that contribute to operator errors in op centers.

Operators of complex systems use a set of cognitive mechanisms that are fallible in predictable ways. Systems engineers, developers, and designers can begin mitigating the risks associated with fallible cognitive behavior by learning about the factors and mechanisms that influence operator performance and reliability. Not all these mechanisms can be ameliorated by system design, but they do shed light on design opportunities where systems could be improved and better support operators. This book suggests ways to do that.

Modifying op center designs could help reduce these types of system failures by providing the information more clearly, making information more comprehensible, 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.

1.3 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 to 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 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.

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 judgment within the design process. At all times, designers must be aware that interfaces that are hard to read, use, understand, or predict from are constant risks to project success; however these issues are not always easily solvable. Designers will have to use judgment when aspects of the users and their tasks are not fully known. They will also have to use judgment to prioritize tasks or user types and to balance different design requirements. Designers face many challenges when balancing human and system factors, and this book will help guide their decision-making when solutions are not immediately clear.

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 direct solution. Thus, this book summarizes a useful process and design issues to keep in mind when designing operation centers. It goes further, however, by providing a worked example of design and design steps for an example system.

This book 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 a set of ten "rules" about font size, which might need to vary and which will conflict at times with rules about how many objects need to be visible on the interface. And, yet, in providing background knowledge about operators and their tasks, there will inevitably be sensible conclusions that look like and work like guidelines. The design recommendations will often provide "safe" recommendations for designers. Design recommendations will be accompanied by brief supporting details meant to substantiate the information. This self-contained book 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 book 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 book, 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 should be detailed enough to provide further guidance in a standalone format.

1.4 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 on 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 effective systems, including assessing the risks associated with design choices (Pew and Mavor 2007). Figure 1.2 shows the RD-ICM in spiral form. 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 leads to decisions contrary to the dogmatic idea that UX be prioritized at every stage, but this is because UX issues are only explored once their risks are relatively large.

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 the 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.

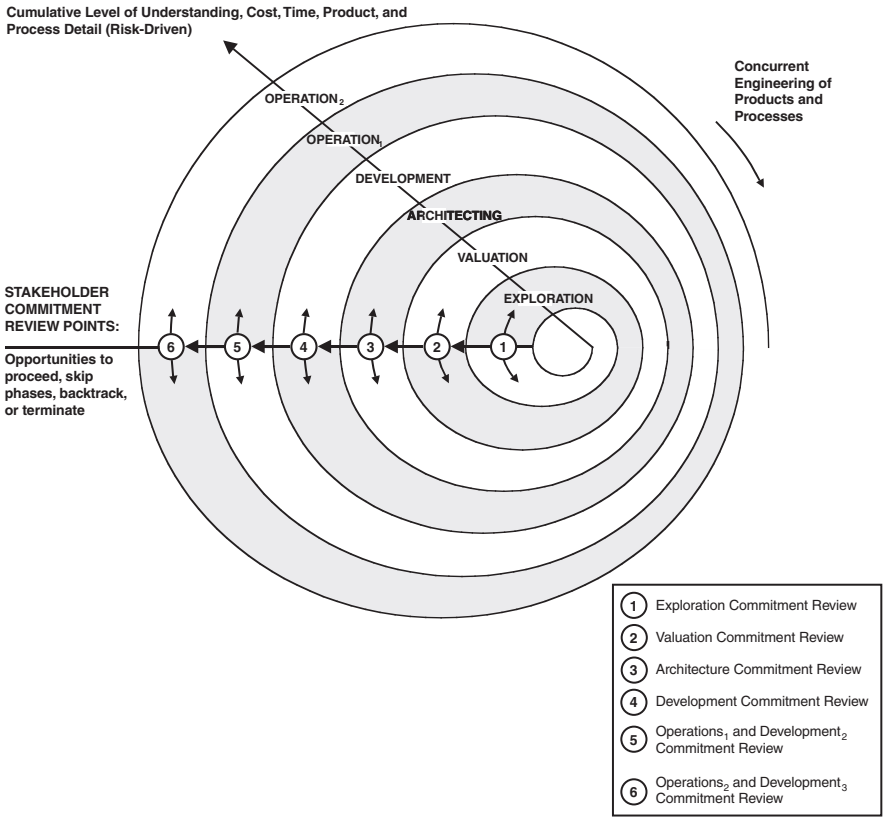

Fig. 1.2 The Risk-Driven Incremental Commitment Model as a spiral of development. (Reprinted from Pew and Mavor 2007, p. 48)

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.

Within the spiral, each stage has phases of (a) stakeholder valuation and evaluation; (b) determination of objectives, alternatives, and constraints; (c) evaluation of alternatives and identification and resolution of risks; and (d) development and verification of the 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., failure to meet expectations for funders or other high-level stakeholders or technical connection issues) may outweigh the risks associated with having a nonideal interface design (e.g., frustrated users). The stakeholders have determined

that functionality (the task-related aspects of the design) should be prioritized over the user experience (UX, the users' feelings, emotions, values, and responses to the system). 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 associated with a frustrating user interface. If the interface fails to convey critical information in a consistent manner to most users, 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 a National Research Council Report (Pew and Mavor 2007), a special issue of the *Journal of Cognitive Engineering and Decision Making* (Pew 2008), and an overview in the *Foundations for Designing User-Centered Systems* textbook (Ritter et al. 2014).

So, if you adopt a risk-driven process that includes human operator-related risks, you still must be able to recognize and reduce these risks. This book 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 op centers. 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.

1.5 The Design Problem Space for Op Centers

This book reviews how the risks of failures due to human performance can be alleviated throughout the design process of interfaces within operation centers. Because 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 common risks and issues that apply to operator interactions with the systems.

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 book, so we address them together.

1.5.1 Know Your Technology

Across the range of stakeholders involved with the design of a system, the most influential stakeholders will likely prioritize system functionality over concerns of operator-related risks like improving user-centered design. While this may irk the designers of human-facing subsystems, this basic fact should influence how the design process is conducted. Thus, system designers should have at least some understanding of how the technology within their system functions.

The underlying, unmanned technology within op centers processes and transmits the information that is presented to an operator. So, the first issue in design is to know what the technology can and cannot do. The technology in an op center is likely built from varied inputs and outputs, ranging from manually entered paper documentation to antenna arrays linked to distant sensors. On its own, a component like an oxygen sensor simply outputs an associated metric. However, once integrated into an environmental monitoring station in an op center, additional design features to support human use (i.e., an interface, optional controls, and memory for time series) become apparent. Interface designers may not need to understand the intricacies of each component but should have some knowledge of the technology associated with their system.

The types of systems built for op centers are likely to differ greatly in their underlying technology and purpose. In some cases, designers can grasp the underlying technology well enough to create effective systems, but this may not always be the case. Building an electrical circuit monitoring system and building a hydrothermal monitoring system may require incorporating subject matter experts into the design process, especially for high-stakes systems like a nuclear power plant.

Finally, designers should understand the tools they need to build interfaces as well. The interface tools need to be able to support the designers in creating usable interfaces, which not all tools support well (Pew and Mavor 2007; Ritter et al. 2014). To our previous example, an electrical circuit monitoring system may require designers to reference an unfamiliar program used by electrical engineers like Pspice (Personal Simulation Program with Integrated Circuit Emphasis). Stakeholders should ensure that system designers can successfully understand and utilize the necessary information.

Understanding the technology within the system and used to build the system will help with the inevitable design choices. The typical issue is where designers should fit the person to the machine vs. fit the machine to the person. Sometimes, technological or personal constraints will prevent designers from optimizing the fit in one direction or another, but knowing the technology will help reduce problems of fit in both directions.

1.5.2 Know Your Users and Their Tasks

On the other hand, designs that do not support users to do their tasks can fail for this reason as well, so system designers need to study the user and how to design for users. The focus of this book is to explain how to know the users of the op centers, the operators, and their tasks. Human operators and their tasks, in many cases, will be as complex as the technology. The only difference is that many technology designers have been trained in technology design, but not in the science of how operators think, learn, and do their tasks. This book notes some of the literature, results, and methods for understanding operators to help in the design process. Similarly, it describes methods for improving the work process, like task analysis (TA), 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 the expected speeds? 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 frequent causes of system failure.

The gold standard in design (Card et al. 1983; Pew and 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 as it can be used, to support the tasks based on the operator's capabilities. Designers who use their own understanding of a system as a reference (instead of that of the actual users) commit the fundamental attribution error and risk-creating systems that are unwieldy or outright unusable by the intended users (Baxter et al. 2014). The fundamental attribution error of design refers to when designers assume all users are just like themselves. As we note in our example system in this book, this is often a mistake and leads 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, or making state information visible only upon query, are all mistakes that are easily avoided, but require knowing the operators and their tasks.

Knowing the frequency and importance of tasks is also important. Common and important tasks should be more easily and safely accomplished than less common and less important tasks. When the two factors of frequency and importance collide, then possible design choices become apparent. At this point designers can assess the situation through the RD-ICM and reduce risk by getting feedback from stakeholders, researching similar design problems, or testing multiple designs depending on the risks associated with each choice.

There are numerous guidelines on how to create task analyses (e.g., Cox 2007; Ritter et al. 2014, Ch. 11). 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.

[1] http://cogulator.io/

TA is a lot like pizza—while the balance of contents may vary in approaches, most versions are usable and enjoyable.

1.5.3 Test Designs Broadly and with Cognitive Walkthroughs

During design and implementation, there may be unknown aspects of the users, their tasks, or the interactions between the two. A way to reduce the risk of system failure is to test the resulting system. The test can be quite simple, for example, simply to see if the tasks can be performed. Alternatively, there are more complex methods, like running a small A/B experiment with two possible designs or measuring task performance with actual users under realistic conditions. Pew and Mavor (2007) review the range of these tests, and there are multiple textbooks describing them (e.g., Cairns and Cox 2008; Lewis and Rieman 1994). Testing interfaces will reduce the range of usability risks, but test methods vary by how much of a time and resource commitment is required to get useful results. Asking someone unfamiliar with the project to review the proposed interface mockup may be essentially free, whereas conducting an A/B test with expected users may take weeks (if not months) to fully set up, run, and analyze, but will be much more useful.

The simplest test is to have naïve operators use the interface and observe them. This approach is explained in many textbooks, including Ritter et al. (2014). Such tests with naïve users could last as little as 10 min and cost next to nothing (i.e., ask a colleague to use the interface and provide comments) or could take multiple months and cost $100 k (i.e., conducting a formal study on task performance under realistic conditions). Stakeholders should consider system requirements and risks to determine how their system should be tested.

We also support using "cognitive walkthroughs" (Polson et al. 1992) to examine the usability of the system. A cognitive walkthrough is a method for evaluating the learnability and usability of an interface by simulating the cognitive activities of a typical user during normal tasks. The typical process for performing cognitive walkthroughs begins with describing the 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, is arranged into a hierarchy. The top-level goals represent the overall task. Each top-level goal is composed of intermediate-level goals (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 compares the goals with the interface and attempts to map how a typical user would accomplish each goal, subtask, and action. If the analyst cannot make some mapping of a goal to the interface, this will suggest an area of the interface that requires improvement or further work. One potential pitfall here can occur if the analyst is too familiar with the interface (relative to a true "typical user"), as they will not see the same problems that 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 provide a coherent information set capable of being understood and followed by the operator (Blackmon et al. 2002; Polson et al. 1992).

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

Whenever detailed time predictions are useful, we recommend using the keystroke-level model (KLM) of Card et al. (1980, 1983). This approach provides time estimates based on the keystrokes, mouse moves, mental operators, system response time, and other possible cognitive operators. The times are engineering estimates (i.e., ± 20%), but basically support fair comparisons of different interfaces. The KLM time predictions suggest where and how time is spent on an interface and can help identify ways to improve performance. The regularity of the interactions across subtasks also suggests how much needs to be 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 and show examples of how potentially abstract principles can be put into practice.

1.6 Example Task: The Mars Water Detection System

This book provides context for readers through a hypothetical use case for a semiautonomous system that searches for water. 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 is a mostly autonomous mobile robot that searches Mars for signs of water, but the WDS sometimes requires human intervention to respond to novel or risky scenarios. 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 high-level 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. The rest of this chapter provides a brief review of the WDS and its design requirements

before concluding with some design recommendations that arise from this chapter. A detailed description is presented in Appendix 1.

1.6.1 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 bidirectional communication between scientists from the Program Office who funded the WDS and the operators responsible for direct interaction with the systems. Figure 1.3 shows a few example interface prototypes for the WDS. While the design will vary depending on the needs of the system, these systems present many different metrics of 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

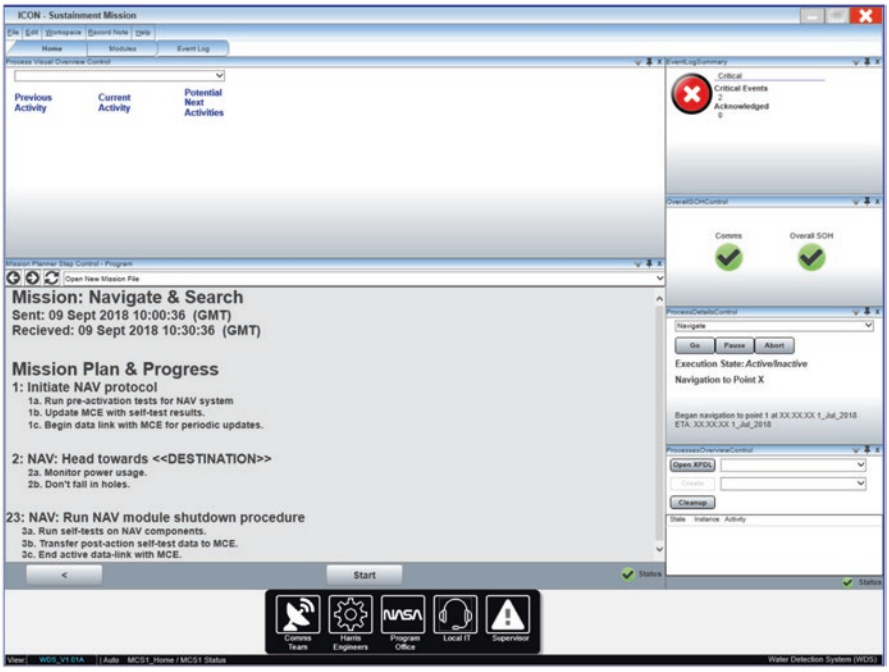

Fig. 1.3 Two example interface designs for the Water Detection System monitoring screen (second example on next page)

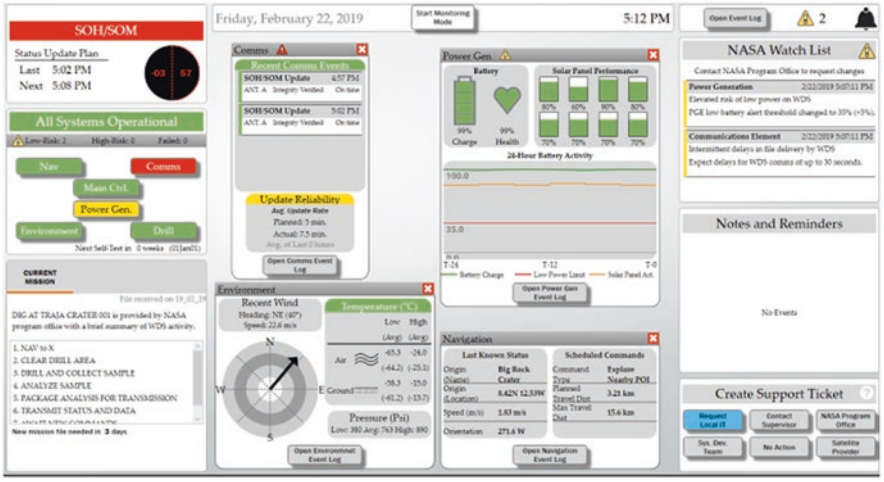

Fig. 1.3 (continued)

able to be completed without direct contact with a supervisor, while others will need direct response from supervisors prior to action.

1.6.2 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, monitoring other subsystems, and linking the subsystems together. The other 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 1.2 and a diagram of the WDS–Earth link is shown in Fig. 1.4.

1.6.3 Example Issues

System designers may be unable to anticipate every risk to system success; however, the Risk-Driven Incremental Commitment Model drives the designers to try to understand what risks are most likely to arise. Table 1.3 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.

Table 1.2 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

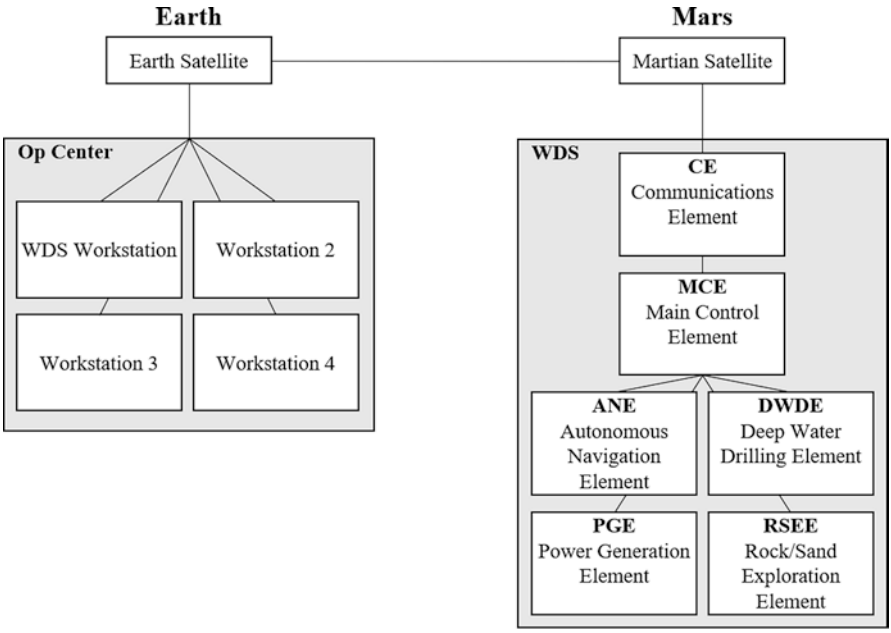

Fig. 1.4 Diagram of the Water Detection System (WDS) and its connection to the operation center

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

Table 1.3 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. The main control element cannot complete all the scheduled commands for the day	Moderate	Communications element sends an alert the NASA operators of the low battery status. Operator must re-task the day's commands because the autonomous navigation element would use all the remaining power	Operator, supervisor
Within the op center, the 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 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

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.

1.7 Principles for Design

Based on the target system description, the example system, and the design process, we can provide an overview of the book 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. We aggregate the most important design principles described in this book in Appendix 3. 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 1.1: Don't Assume the User to Be How You Think You Are
One of the 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 developer, systems engineer, or astronaut, you will almost always

need to adapt your design to meet the operator's system-related needs, capabilities, and wants (in that order).

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. It is therefore important to provide designers and engineers with the ability to consult users and other stakeholders throughout the design process. Methods for learning about users can include talking with them, watching them work, having them use your interfaces, reading their autobiographies, or watching movies about their work environments (whether documentaries or even fictional accounts). Each of these methods for understanding users will gather only a subset of the useful information; casting a wide net can reduce the risk of overgeneralization and improve the breadth of the knowledge gleaned from users.

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 (e.g., Bias and Mayhew 2005; Lewis and Rieman 1994; Pew and Mavor 2007; Ritter et al. 2014).

Principle 1.2: All Design Choices Have Trade-offs—Don't Go in Blind
Most design choices have trade-offs. This basic fact will provide engineers with difficult decisions throughout the design process. For example, increased font size may increase readability by sacrificing some valuable interface "real estate" and limiting the total amount of information displayed. Effectively resolving these difficult design choices requires designers to use knowledge of the tasks and users to make informed decisions. Use of the risk-driven spiral model helps engineers make the best decision given the constraints by consulting with stakeholders and using what others have already learned. Designers will be presented with problems like this, both big and small, throughout the design process, and not every individual design choice is worthy of a full user study.

For example, consider a system that requires operators to search for digital files while performing other tasks. An informed designer may realize that recognition memory (i.e., "Is 'book_manuscriptV47_final.docx' the file you are looking for?") is more robust than recall memory (i.e., "What is the exact name of the file you are looking for?"). While searching for files on a system, it is usually easy and familiar to point and click around a series of folders to find some item, as in the standard desktop operating system. Using a keystroke-based system (like a command line) might be faster, but typically will require more experienced users or more training. Stakeholders should consider which design would be best suited for their system needs, users, and tasks.

As another example, consider a system that tasks operators with monitoring incoming pings and classifying them as friendly, hostile, or unknown. An informed designer will know that speed and accuracy are traded off when improving performance. Emphasizing speed will require sacrificing accuracy (i.e., more errors), and

the inverse is true as well. Stakeholders can use this knowledge to analyze how to reach an acceptable balance between accuracy and speed. Although ideal solutions are not always possible, designers can meet expectations by understanding the expectations for task time and error rate.

Finally, almost any point-and-click system will use menu trees to support navigation. Many studies have explored how users' decision-making, reaction time, and error rate change in response to changing the menu design. The Hick–Hyman Law (Hick 1952; Hyman 1953) predicts that choosing between more options (e.g., five menu choices vs. three menu choices) takes longer, but the menu is more likely to contain the correct choice. Signal detection theory shows a similar trade-off between hits, misses, false alarms, and correct rejections.

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

Principle 1.3: Use and Test Multiple Designs
When designing a new display or component, create and consider multiple versions. Get feedback on the possible designs from a source (or sources) that is as objective as possible.

When you create a new display, particularly high stakes or main displays, you should consider multiple versions. Considering multiple versions of designs tends 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' behavior.

Research by Steven Dow examined the design process in the egg drop task. In this task, designers were given a set of standard materials and asked to design a protective cradle for an egg so it will survive a large vertical drop. Groups that designed more examples and that tested more often had reliably higher distances from which their eggs could be safely dropped. Dow argues that the beneficial outcomes seen from multiple designs will apply to other design tasks, and we agree.

1.8 Conclusion

Throughout the design of an op 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 on which to err. 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. Systems engineers will be neglecting a crucial component of their system if they do not account for the system's compatibility with the human operators. Although 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

Table 1.4 Questions to be answered by this book for systems like the WDS

Process performance
1. Which 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 troubleshooting?
High-throughput reaction times
3. Which features in 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 user interface? 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 user interface?
6. What are the effects of time-on-task (i.e., work shift length) on reaction time and accuracy for a user using the system?
Interface generalizability 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?

target population. The user interface should facilitate high performance without undue stress on the operators.

Table 1.4 notes some questions that designers might have in mind when designing and implementing control rooms, op centers, and other similar systems. The next two chapters will review the psychology and human factors concepts and theories that give rise to the principles described above and should be considered to help answer the questions in Table 1.4. In the conclusion to this book, we will note how these questions have been answered.

References

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.

Bias, R. G., & Mayhew, D. J. (2005). *Cost-justifying usability: An update for the internet age*. San Francisco: Morgan Kaufmann. https://doi.org/10.1016/B978-0-12-095811-5.X5000-7.

Blackmon, M. H., Polson, P. G., Kitajima M, & Lewis, C. (2002). *Cognitive walkthrough for the web*. In CHI 2002: *Proceedings of the Conference on Human Factors in Computing Systems*, (pp. 463–470). New York: ACM Press.

Boehm, B., & Hansen, W. (2001). The spiral model as a tool for evolutionary acquisition. *CrossTalk, 14*(5), 4–11.

Cairns, P., & Cox, A. L. (2008). *Research methods for human-computer interaction* (1st ed., eds. P. Cairns & A. L. Cox). New York: Cambridge University Press.

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/10.1145/358886.358895.

Card, S. K., Moran, T. P., & Newell, A. (1983). *The psychology of human-computer interaction*. Hillsdale: Lawrence Erlbaum.

Cox, D. (2007). Task analysis, usability and engagement. In *Human-computer interaction. Interaction design and usability* (pp. 1072–1081). Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-540-73105-4_117.

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

Endsley, M. R. (1995). Toward a theory of situation awareness in dynamic systems. *Human Factors, 37*(1), 32–64. https://doi.org/10.1518/001872095779049543.

Hick, W. E. (1952). On the rate of gain of information. *Quarterly Journal of Experimental Psychology, 4*(1), 11–26. https://doi.org/10.1080/17470215208416600.

Hyman, R. (1953). Stimulus information as a determinant of reaction time. *Journal of Experimental Psychology, 45*(3), 188–196. https://doi.org/10.1037/h0056940.

Jones, D. G., & Endsley, M. R. (1996). Sources of situation awareness errors in aviation. *Aviation, Space, and Environmental Medicine, 67*(6), 507–512. https://doi.org/10.1039/c4qo00187g.

Lewis, C., & Rieman, J. (1994). *Task-centered user interface design: A practical introduction*. Retrieved from http://www.hcibib.org/tcuid/

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*. Washington, DC: The National Academies Press. https://doi.org/10.17226/11893.

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.

Ritter, F. E., Baxter, G. D., & Churchill, E. F. (2014). *Foundations for designing user-centered systems*. London: Springer. https://doi.org/10.1007/978-1-4471-5134-0.