9 Safety-Guided Design

In the examples of STPA in the last chapter, the development of the design was assumed to occur independently. Most of the time, hazard analysis is done after the major design decisions have been made. But STPA can be used in a proactive way to help guide the design and system development, rather than as simply a hazard analysis technique on an existing design. This integrated design and analysis process is called *safety-guided design* (figure 9.1).

As the systems we build and operate increase in size and complexity, the use of sophisticated system engineering approaches becomes more critical. Important system-level (emergent) properties, such as safety, must be built into the design of these systems; they cannot be effectively added on or simply measured afterward. Adding barriers or protection devices after the fact is not only enormously more expensive, it is also much less effective than designing safety in from the beginning (see *Safeware*, chapter 16). This chapter describes the process of safety-guided design, which is enhanced by defining accident prevention as a control problem rather than a "prevent failures" problem. The next chapter shows how safety engineering and safety-guided design can be integrated into basic system engineering processes.

9.1 The Safety-Guided Design Process

One key to having a cost-effective safety effort is to embed it into a system engineering process from the very beginning and to design safety into the system as the design decisions are made. Once again, the process starts with the fundamental activities in chapter 7. After the hazards and system-level safety requirements and constraints have been identified; the design process starts:

1. Try to eliminate the hazards from the conceptual design.
2. If any of the hazards cannot be eliminated, then identify the potential for their control at the system level.

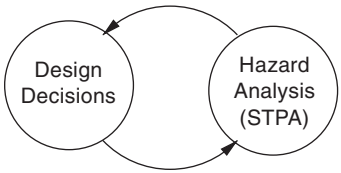


Figure 9.1
Safety-guided design entails tightly intertwining the design decisions and their analysis to support better decision making.

3. Create a system control structure and assign responsibilities for enforcing safety constraints. Some guidance for this process is provided in the operations and management chapters.
4. Refine the constraints and design in parallel.
   a. Identify potentially hazardous control actions by each of system components that would violate system design constraints using STPA step 1. Restate the identified hazard control actions as component design constraints.
   b. Using STPA Step 2, determine what factors could lead to a violation of the safety constraints.
   c. Augment the basic design to eliminate or control potentially unsafe control actions and behaviors.
   d. Iterate over the process, that is, perform STPA steps 1 and 2 on the new augmented design and continue to refine the design until all hazardous scenarios are eliminated, mitigated, or controlled.

The next section provides an example of the process. The rest of the chapter discusses safe design principles for physical processes, automated controllers, and human controllers.

9.2 An Example of Safety-Guided Design for an Industrial Robot

The process of safety-guided design and the use of STPA to support it is illustrated here with the design of an experimental Space Shuttle robotic Thermal Tile Processing System (TTPS) based on a design created for a research project at CMU [57].

The goal of the TTPS system is to inspect and waterproof the thermal protection tiles on the belly of the Space Shuttle, thus saving humans from a laborious task, typically lasting three to four months, that begins within minutes after the Shuttle

lands and ends just prior to launch. Upon landing at either the Dryden facility in California or Kennedy Space Center in Florida, the orbiter is brought to either the Mate-Demate Device (MDD) or the Orbiter Processing Facility (OPF). These large structures provide access to all areas of the orbiters.

The Space Shuttle is covered with several types of heat-resistant tiles that protect the orbiter's aluminum skin during the heat of reentry. While the majority of the upper surfaces are covered with flexible insulation blankets, the lower surfaces are covered with silica tiles. These tiles have a glazed coating over soft and highly porous silica fibers. The tiles are 95 percent air by volume, which makes them extremely light but also makes them capable of absorbing a tremendous amount of water. Water in the tiles causes a substantial weight problem that can adversely affect launch and orbit capabilities for the shuttles. Because the orbiters may be exposed to rain during transport and on the launch pad, the tiles must be waterproofed. This task is accomplished through the use of a specialized hydrophobic chemical, DMES, which is injected into each tile. There are approximately 17,000 lower surface tiles covering an area that is roughly 25m × 40m.

In the standard process, DMES is injected into a small hole in each tile by a handheld tool that pumps a small quantity of chemical into the nozzle. The nozzle is held against the tile and the chemical is forced through the tile by a pressurized nitrogen purge for several seconds. It takes about 240 hours to waterproof the tiles on an orbiter. Because the chemical is toxic, human workers have to wear heavy suits and respirators while injecting the chemical and, at the same time, maneuvering in a crowded work area. One goal for using a robot to perform this task was to eliminate a very tedious, uncomfortable, and potentially hazardous human activity.

The tiles must also be inspected. A goal for the TTPS was to inspect the tiles more accurately than the human eye and therefore reduce the need for multiple inspections. During launch, reentry, and transport, a number of defects can occur on the tiles in the form of scratches, cracks, gouges, discoloring, and erosion of surfaces. The examination of the tiles determines if they need to be replaced or repaired. The typical procedures involve visual inspection of each tile to see if there is any damage and then assessment and categorization of the defects according to detailed checklists. Later, work orders are issued for repair of individual tiles.

Like any design process, safety-guided design starts with identifying the goals for the system and the constraints under which the system must operate. The high-level goals for the TTPS are to:

1. Inspect the thermal tiles for damage caused during launch, reentry, and transport
2. Apply waterproofing chemicals to the thermal tiles

Environmental constraints delimit how these goals can be achieved and identifying those constraints, particularly the safety constraints, is an early goal in safety-guided design.

The environmental constraints on the system design stem from physical properties of the Orbital Processing Facility (OPF) at KSC, such as size constraints on the physical system components and the necessity of any mobile robotic components to deal with crowded work areas and for humans to be in the area. Example work area environmental constraints for the TTPS are:

EA1: The work areas of the Orbiter Processing Facility (OPF) can be very crowded. The facilities provide access to all areas of the orbiters through the use of intricate platforms that are laced with plumbing, wiring, corridors, lifting devices, and so on. After entering the facility, the orbiters are jacked up and leveled. Substantial structure then swings around and surrounds the orbiter on all sides and at all levels. With the exception of the jack stands that support the orbiters, the floor space directly beneath the orbiter is initially clear but the surrounding structure can be very crowded.

EA2: The mobile robot must enter the facility through personnel access doors 1.1 meters (42″) wide. The layout within the OPF allows a length of 2.5 meters (100″) for the robot. There are some structural beams whose heights are as low as 1.75 meters (70″), but once under the orbiter the tile heights range from about 2.9 meters to 4 meters. The compact roll-in form of the mobile system must maneuver these spaces and also raise its inspection and injection equipment up to heights of 4 meters to reach individual tiles while still meeting a 1 millimeter accuracy requirement.

EA3: Additional constraints involve moving around the crowded workspace. The robot must negotiate jack stands, columns, work stands, cables, and hoses. In addition, there are hanging cords, clamps, and hoses. Because the robot might cause damage to the ground obstacles, cable covers will be used for protection and the robot system must traverse these covers.

Other design constraints on the TTPS include:

- Use of the TTPS must not negatively impact the flight schedules of the orbiters more than that of the manual system being replaced.
- Maintenance costs of the TTPS must not exceed x dollars per year.
- Use of the TTPS must not cause or contribute to an unacceptable loss (accident) as defined by Shuttle management.

As with many systems, prioritizing the hazards by severity is enough in this case to assist the engineers in making decisions during design. Sometimes a preliminary

hazard analysis is performed using a risk matrix to determine how much effort will be put into eliminating or controlling the hazards and in making tradeoffs in design. Likelihood, at this point, is unknowable but some type of surrogate, like mitigatibility, as demonstrated in section 10.3.4, could be used. In the TTPS example, severity plus the NASA policy described earlier is adequate. To decide not to consider some of the hazards at all would be pointless and dangerous at this stage of development as likelihood is not determinable. As the design proceeds and decisions must be made, specific additional information may be found to be useful and acquired at that time. After the system design is completed, if it is determined that some hazards cannot be adequately handled or the compromises required to handle them are too great; then the limitations would be documented (as described in chapter 10) and decisions would have to be made at that point about the risks of using the system. At that time, however, the information necessary to make those decisions will more likely be available than before the development process begins.

After the hazards are identified, system-level safety-related requirements and design constraints are derived from them. As an example, for hazard H7 (inadequate thermal protection), a system-level safety design constraint is that the mobile robot processing must not result in any tiles being missed in the inspection or waterproofing process. More detailed design constraints will be generated during the safety-guided design process.

To get started, a general system architecture must be selected (figure 9.2). Let's assume that the initial TTPS architecture consists of a mobile base on which tools will be mounted, including a manipulator arm that performs the processing and contains the vision and waterproofing tools. This very early decision may be changed after the safety-guided design process starts, but some very basic initial assumptions are necessary to get going. As the concept development and detailed design process proceeds, information generated about hazards and design tradeoffs may lead to changes in the initial configuration. Alternatively, multiple design configurations may be considered in parallel.

In the initial candidate architecture (control structure), a decision is made to introduce a human operator in order to supervise robot movement as so many of the hazards are related to movement. At the same time, it may be impractical for an operator to monitor all the activities so the first version of the system architecture is to have the TTPS control system in charge of the non-movement activities and to have both the TTPS and the control room operator share control of movement. The safety-guided design process, including STPA, will identify the implications of this decision and will assist in analyzing the allocation of tasks to the various components to determine the safety tradeoffs involved.

In the candidate starting architecture (control structure), there is an automated robot work planner to provide the overall processing goals and tasks for the

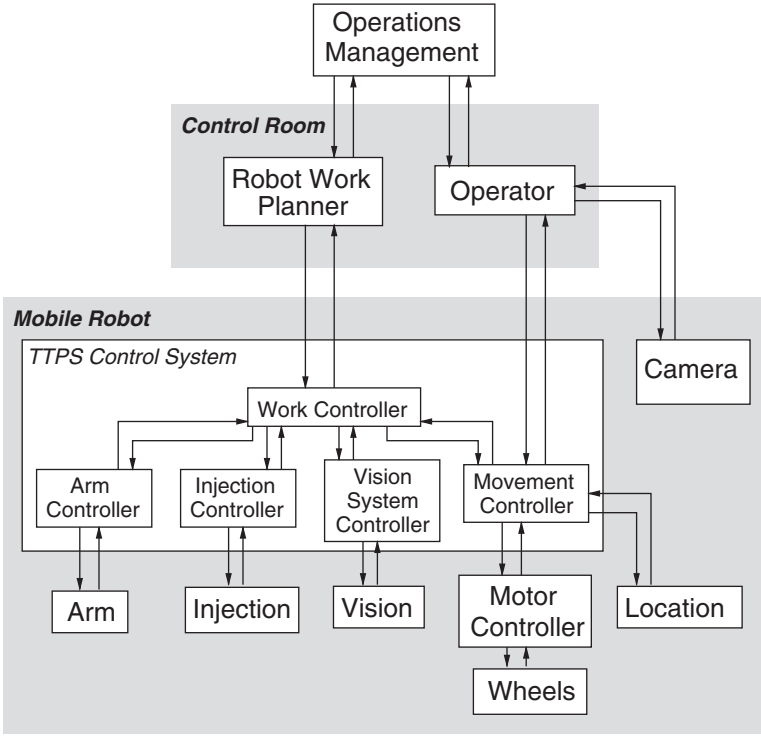


Figure 9.2
A candidate structure for the TTPS.

TTPS. A location system is needed to provide information to the movement controller about the current location of the robot. A camera is used to provide information to the human controller, as the control room will be located at a distance from the orbiter. The role of the other components should be obvious.

The proposed design has two potential movement controllers, so coordination problems will have to be eliminated. The operator could control all movement, but that may be considered impractical given the processing requirements. To assist with this decision process, engineers may create a *concept of operations* and perform a human task analysis [48, 122].

The safety-guided design process, including STPA, will identify the implications of the basic decisions in the candidate tasks and will assist in analyzing the allocation of tasks to the various components to determine the safety tradeoffs involved.

The design process is now ready to start. Using the information already specified, particularly the general functional responsibilities assigned to each component,

designers will identify potentially hazardous control actions by each of the system components that could violate the safety constraints, determine the causal factors that could lead to these hazardous control actions, and prevent or control them in the system design. The process thus involves a top-down identification of scenarios in which the safety constraints could be violated. The scenarios can then be used to guide more detailed design decisions.

In general, safety-guided design involves first attempting to eliminate the hazard from the design and, if that is not possible or requires unacceptable tradeoffs, reducing the likelihood the hazard will occur, reducing the negative consequences of the hazard if it does occur, and implementing contingency plans for limiting damage. More about design procedures is presented in the next section.

As design decisions are made, an STPA-based hazard analysis is used to inform these decisions. Early in the system design process, little information is available, so the hazard analysis will be very general at first and will be refined and augmented as additional information emerges through the system design activities.

For the example, let's focus on the robot instability hazard. The first goal should be to eliminate the hazard in the system design. One way to eliminate potential instability is to make the robot base so heavy that it cannot become unstable, no matter how the manipulator arm is positioned. A heavy base, however, could increase the damage caused by the base coming into contact with a human or object or make it difficult for workers to manually move the robot out of the way in an emergency situation. An alternative solution is to make the base long and wide so the moment created by the operation of the manipulator arm is compensated by the moments created by base supports that are far from the robot's center of mass. A long and wide base could remove the hazard but may violate the environmental constraints in the facility layout, such as the need to maneuver through doors and in the crowded OPF.

The environmental constraint EA2 above implies a maximum length for the robot of 2.5 meters and a width no larger than 1.1 meter. Given the required maximum extension length of the manipulator arm and the estimated weight of the equipment that will need to be carried on the mobile base, a calculation might show that the length of the robot base is sufficient to prevent any longitudinal instability, but that the width of the base is not sufficient to prevent lateral instability.

If eliminating the hazard is determined to be impractical (as in this case) or not desirable for some reason, the alternative is to identify ways to control it. The decision to try to control it may turn out not to be practical or later may seem less satisfactory than increasing the weight (the solution earlier discarded). All decisions

should remain open as more information is obtained about alternatives and backtracking is an option.

At the initial stages in design, we identified only the general hazards—for example, instability of the robot base and the related system design constraint that the mobile base must not be capable of falling over under worst-case operational conditions. As design decisions are proposed and analyzed, they will lead to additional refinements in the hazards and the design constraints.

For example, a potential solution to the stability problem is to use lateral stabilizer legs that are deployed when the manipulator arm is extended but must be retracted when the robot base moves. Let's assume that a decision is made to at least consider this solution. That potential design decision generates a new refined hazard from the high-level stability hazard (H2):

H2.1: The manipulator arm is extended while the stabilizer legs are not fully extended.

Damage to the mobile base or other equipment around the OPF is another potential hazard introduced by the addition of the legs if the mobile base moves while the stability legs are extended. Again, engineers would consider whether this hazard could be eliminated by appropriate design of the stability legs. If it cannot, then that is a second additional hazard that must be controlled in the design with a corresponding design constraint that the mobile base must not move with the stability legs extended.

There are now two new refined hazards that must be translated into design constraints:

1. The manipulator arm must never be extended if the stabilizer legs are not extended.
2. The mobile base must not move with the stability legs extended.

STPA can be used to further refine these constraints and to evaluate the resulting designs. In the process, the safety control structure will be refined and perhaps changed. In this case, a controller must be identified for the stabilizer legs, which were previously not in the design. Let's assume that the legs are controlled by the TTPS movement controller (figure 9.3).

Using the augmented control structure, the remaining activities in STPA are to identify potentially hazardous control actions by each of the system components that could violate the safety constraints, determine the causal factors that could lead to these hazardous control actions, and prevent or control them in the system design. The process thus involves a top-down identification of scenarios in which the safety

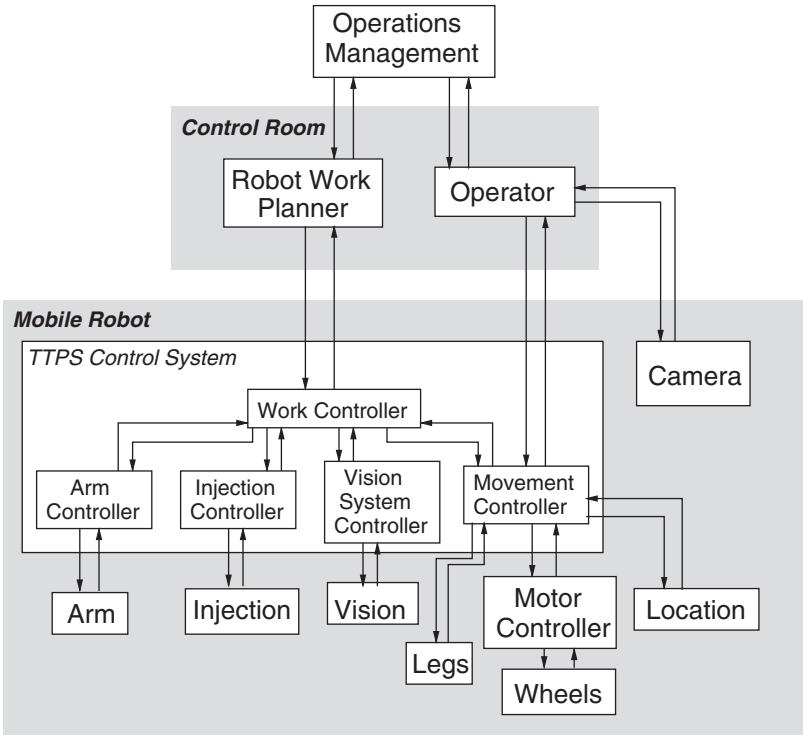


Figure 9.3
A refined control structure for the TTPS.

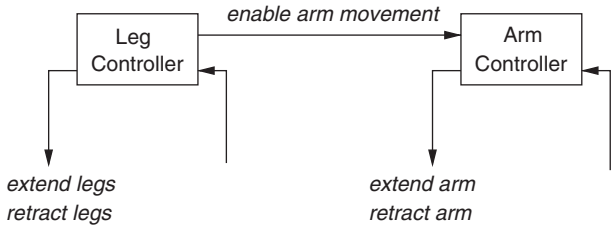
constraints could be violated so that they can be used to guide more detailed design decisions.

The unsafe control actions associated with the stability hazard are shown in figure 9.4. Movement and thermal tile processing hazards are also identified in the table. Combining similar entries for H1 in the table leads to the following unsafe control actions by the leg controller with respect to the instability hazard:

1. The leg controller does not command a deployment of the stabilizer legs before the arm is extended.
2. The leg controller commands a retraction of the stabilizer legs before the manipulator arm is fully stowed.
3. The leg controller commands a retraction of the stabilizer legs after the arm has been extended or commands a retraction of the stabilizer legs before the manipulator arm is stowed.

HAZARD1: Arm extended while legs retracted

HAZARD2: Legs extended during movement



Command	Not Providing Causes Hazard	Providing Causes Hazard	Timing/Sequencing Causes Hazard	Stopped Too Soon or Applied Too Long
extend legs	Legs not extended before arm extended **H1**	Extend legs during movement **H2**	Extend arm before legs extended **H1**	Stop before fully extended **H1**
retract legs	Not retracted before movement **H2**	Retract while arm extended **H1**	Retract legs before arm fully stowed **H1**	Stop while still partially extended **H1**

Command	Not Providing Causes Hazard	Providing Causes Hazard	Timing/Sequencing Causes Hazard	Stopped Too Soon or Applied Too Long
extend arm	Not Hazardous	Extend arm when legs retracted **H1**	Extend arm before legs fully extended **H1**	(tile processing hazard)
retract arm	Not retracted before movement **H2**	(tile processing hazard)	(tile processing hazard)	Stop retraction before arm fully stowed and movement starts or legs retracted **H1 H2**

Figure 9.4
STPA step 1 for the stability and movement hazards related to the leg and arm control.

4. The leg controller stops extension of the stabilizer legs before they are fully extended.

and by the arm controller:

1. The arm controller extends the manipulator arm when the stabilizer legs are not extended or before they are fully extended.

The inadequate control actions can be restated as system safety constraints on the controller behavior (whether the controller is automated or human):

1. The leg controller must ensure the stabilizer legs are fully extended before arm movements are enabled.
2. The leg controller must not command a retraction of the stabilizer legs when the manipulator arm is not in a fully stowed position.
3. The leg controller must command a deployment of the stabilizer legs before arm movements are enabled; the leg controller must not command a retraction of the stabilizer legs before the manipulator arm is stowed.
4. The leg controller must not stop the leg extension until the legs are fully extended.

Similar constraints will be identified for all hazardous commands: for example, the arm controller must not extend the manipulator arm before the stabilizer legs are fully extended.

These system safety constraints might be enforced through physical interlocks, human procedures, and so on. Performing STPA step 2 will provide information during detailed design (1) to evaluate and compare the different design choices, (2) to design the controllers and design fault tolerance features for the system, and (3) to guide the test and verification procedures (or training for humans). As design decisions and safety constraints are identified, the functional specifications for the controllers can be created.

To produce detailed scenarios for the violation of safety constraints, the control structure is augmented with process models. The preliminary design of the process models comes from the information necessary to ensure the system safety constraints hold. For example, the constraint that the arm controller must not enable manipulator movement before the stabilizer legs are completely extended implies there must be some type of feedback to the arm controller to determine when the leg extension has been completed.

While a preliminary functional decomposition of the system components is created to start the process, as more information is obtained from the hazard analysis and the system design continues, this decomposition may be altered to optimize fault tolerance and communication requirements. For example, at this point the need

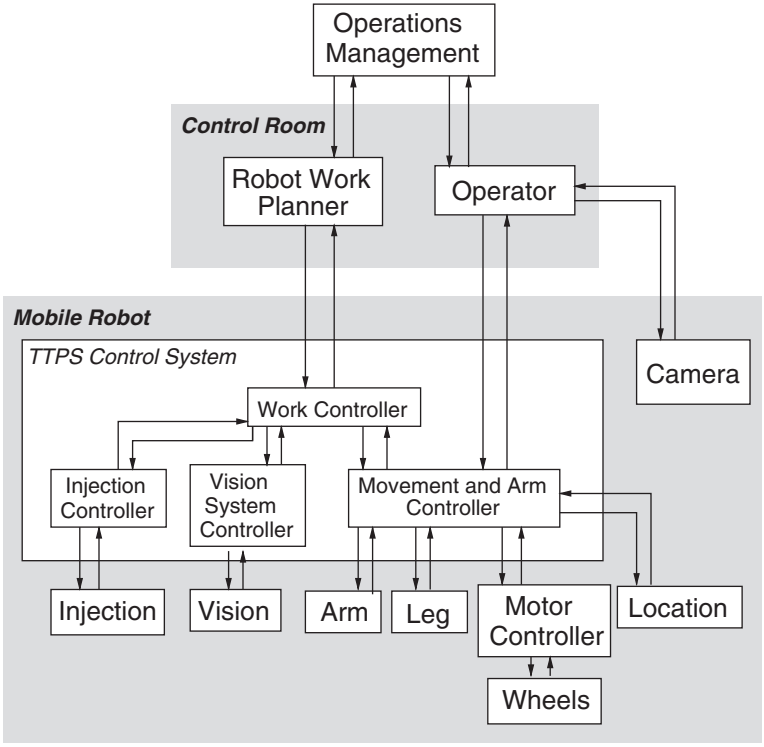


Figure 9.5
A further refined control structure for the TTPS.

for the process models of the leg and arm controllers to be consistent and the communication required to achieve this goal may lead the designers to decide to combine the leg and arm controllers (figure 9.5).

Causal factors for the stability hazard being violated can be determined using STPA step 2. Feedback about the position of the legs is clearly critical to ensure that the process model of the state of the stabilizer legs is consistent with the actual state. The movement and arm controller cannot assume the legs are extended simply because a command was issued to extend them. The command may not be executed or may only be executed partly. One possible scenario, for example, involves an external object preventing the complete extension of the stabilizer legs. In that case, the robot controller (either human or automated) may assume the stabilizer legs are extended because the extension motors have been powered up (a common type of design error). Subsequent movement of the manipulator arm would then violate the identified safety constraints. Just as the analysis assists in refining the component safety constraints (functional requirements), the causal analysis can be used to

further refine those requirements and to design the control algorithm, the control loop components, and the feedback necessary to implement them.

Many of the causes of inadequate control actions are so common that they can be restated as general design principles for safety-critical control loops. The requirement for feedback about whether a command has been executed in the previous paragraph is one of these. The rest of this chapter presents those general design principles.

9.3 Designing for Safety

Hazard analysis using STPA will identify application-specific safety design constraints that must be enforced by the control algorithm. For the thermal-tile processing robot, a safety constraint identified above is that the manipulator arm must never be extended if the stabilizer legs are not fully extended. Causal analysis (step 2 of STPA) can identify specific causes for the constraint to be violated and design features can be created to eliminate or control them.

More general principles of safe control algorithm functional design can also be identified by using the general causes of accidents as defined in STAMP (and used in STPA step 2), general engineering principles, and common design flaws that have led to accidents in the past.

Accidents related to software or system logic design often result from incompleteness and unhandled cases in the functional design of the controller. This incompleteness can be considered a requirements or functional design problem. Some requirements completeness criteria were identified in *Safeware* and specified using a state machine model. Here those criteria plus additional design criteria are translated into functional design principles for the components of the control loop.

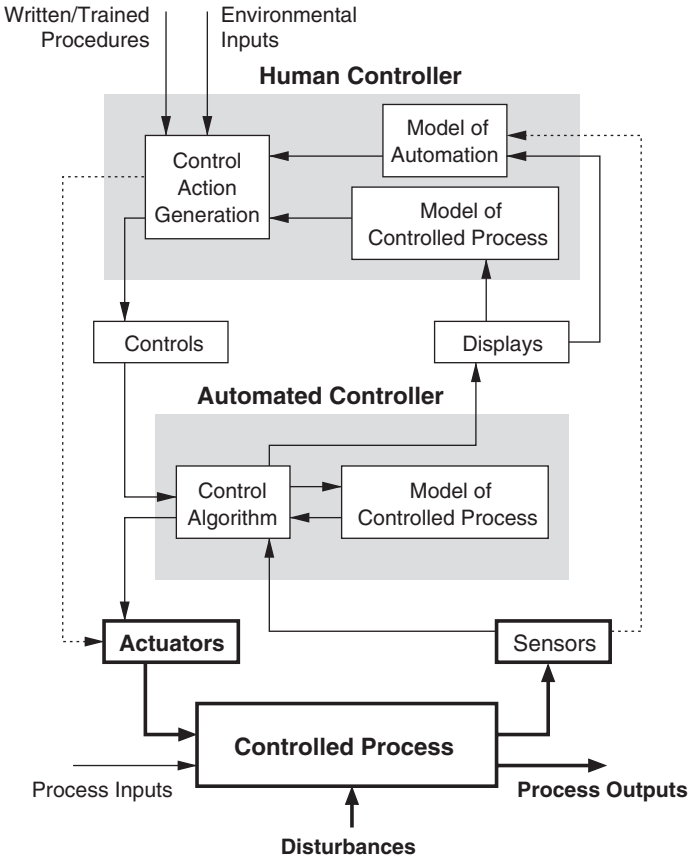
In STAMP, accidents are caused by inadequate control. The controllers can be human or physical. This section focuses on design principles for the components of the control loop that are important whether a human is in the loop or not. Section 9.4 describes extra safety-related design principles that apply for systems that include human controllers. We cannot "design" human controllers, but we can design the environment or context in which they operate, and we can design the procedures they use, the control loops in which they operate, the processes they control, and the training they receive.

9.3.1 Controlled Process and Physical Component Design

Protection against component failure accidents is well understood in engineering. Principles for safe design of common hardware systems (including sensors and actuators) with standard safety constraints are often systematized and encoded in checklists for an industry, such as mechanical design or electrical design. In addition,

most engineers have learned about the use of redundancy and overdesign (safety margins) to protect against component failures.

These standard design techniques are still relevant today but provide little or no protection against component interaction accidents. The added complexity of redundant designs may even increase the occurrence of these accidents. Figure 9.6 shows the design precedence described in *Safeware*. The highest precedence is to eliminate the hazard. If the hazard cannot be eliminated, then its likelihood of occurrence should be reduced, the likelihood of it leading to an accident should be reduced and, at the lowest precedence, the design should reduce the potential damage incurred. Clearly, the higher the precedence level, the more effective and less costly will be the safety design effort. As there is little that is new here that derives from using the STAMP causality model, the reader is referred to *Safeware* and standard engineering references for more information.



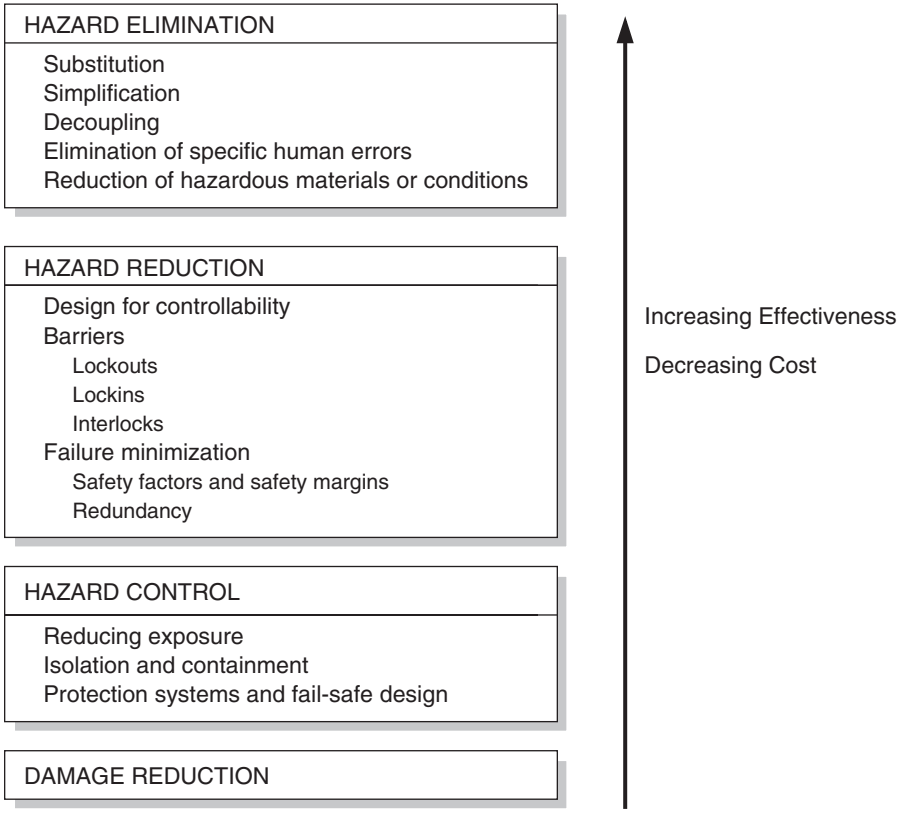


Figure 9.6
Basic system safety design precedence.

9.3.2 Functional Design of the Control Algorithm

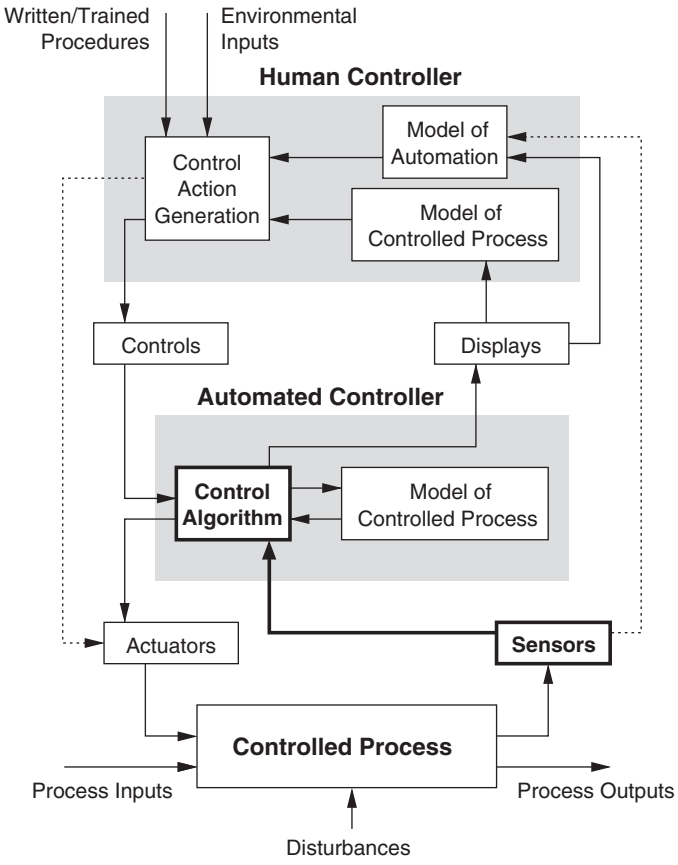
Design for safety includes more than simply the physical components but also the control components. We start by considering the design of the control algorithm.

The controller algorithm is responsible for processing inputs and feedback, initializing and updating the process model, and using the process model plus other knowledge and inputs to produce control outputs. Each of these is considered in turn.

Designing and Processing Inputs and Feedback

The basic function of the algorithm is to implement a feedback control loop, as defined by the controller responsibilities, along with appropriate checks to detect internal or external failures or errors.

Feedback is critical for safe control. Without feedback, controllers do not know whether their control actions were received and performed properly or whether



these commands were effective in achieving the controllers' goals. Feedback is also critical in detecting errors and failures, both errors in the controllers' own actions and failures or faults in the controlled system. Finally, feedback is important in updating process models and in learning about the system and how it will respond to a variety of situations.

Updating process models requires feedback about the current state of the system and any changes that occur. In a system where rapid response is necessary, timing requirements must be placed on the feedback information that the controller uses to make decisions. In addition, when task performance requires or implies a need for the controller to assess timeliness of information, the feedback should include time and date information.

Hazard analysis using STPA will provide information about the types of feedback needed and when. Some additional guidance can be provided to the designer, once again, using general safety design principles.

The controller must be designed to respond appropriately to the arrival of any possible (i.e., detectable by the sensors) input at any time as well as the lack of an expected input over a given time period. Humans are better (and more flexible) than automated controllers at this task. Often automation is not designed to handle input arriving unexpectedly, for example, a target detection report from a radar that was previously sent a message to shut down.

All inputs should be checked for out-of-range or unexpected values and a response designed into the control algorithm. A surprising number of losses still occur due to software not being programmed to handle unexpected inputs.

In addition, the time bounds (minimum and maximum) for every input should be checked and appropriate behavior provided in case the input does not arrive within these bounds. There should also be a response for the non-arrival of an input within a given amount of time (a timeout) for every variable in the process model. The controller must also be designed to respond to excessive inputs (overload conditions) in a safe way.

Because sensors and input channels can fail, there should be a minimum-arrival-rate check for each physically distinct communication path, and the controller should have the ability to query its environment with respect to inactivity over a given communication path. Traditionally these queries are called *sanity* or *health checks*. Care needs to be taken, however, to ensure that the design of the response to a health check is distinct from the normal inputs and that potential hardware failures cannot impact the sanity checks. As an example of the latter, in June 1980 warnings were received at the U.S. command and control headquarters that a major nuclear attack had been launched against the United States [180]. The military prepared for retaliation, but the officers at command headquarters were able to ascertain from direct contact with warning sensors that no incoming missile had been detected and the alert was canceled. Three days later, the same thing happened again. The false alerts were caused by the failure of a computer chip in a multiplexor system that formats messages sent out continuously to command posts indicating that communication circuits are operating properly. This *health check* message was designed to report that there were 000 ICBMs and 000 SLBMs detected. Instead, the integrated circuit failure caused some of the zeros to be replaced with twos. After the problem was diagnosed, the message formats were changed to report only the status of the communication system and nothing about detecting ballistic missiles. Most likely, the developers thought it would be easier to have one common message format but did not consider the impact of erroneous hardware behavior.

STAMP identifies inconsistency between the process model and the actual system state as a common cause of accidents. Besides incorrect feedback, as in the example early warning system, a common way for the process model to become

inconsistent with the state of the actual process is for the controller to assume that an output command has been executed when it has not. The TTPS controller, for example, assumes that because it has sent a command to extend the stabilizer legs, the legs will, after a suitable amount of time, be extended. If commands cannot be executed for any reason, including time outs, controllers have to know about it. To detect errors and failures in the actuators or controlled process, there should be an input (feedback) that the controller can use to detect the effect of any output on the process.

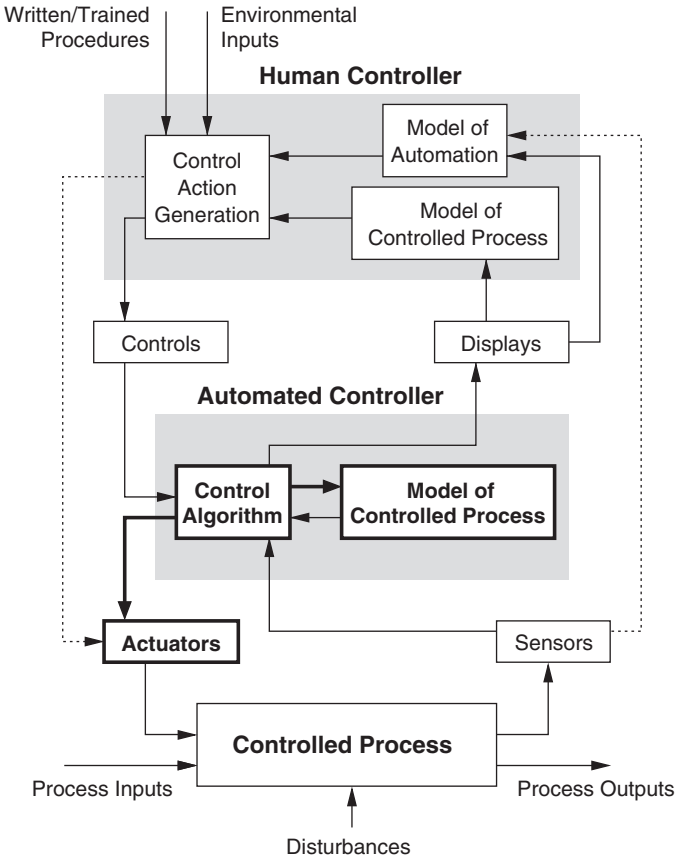
This feedback, however, should not simply be an indication that the command arrived at the controlled process—for example, the command to open a valve was received by the valve, but that the valve actually opened. An explosion occurred in a U.S. Air Force system due to overpressurization when a relief valve failed to open after the operator sent a command to open it. Both the position indicator light and open indicator light were illuminated on the control board. Believing the primary valve had opened, the operator did not open the secondary valve, which was to be used if the primary valve failed. A post-accident examination discovered that the indicator light circuit was wired to indicate presence of a signal at the valve, but it did not indicate valve position. The indicator therefore showed only that the activation button had been pushed, not that the valve had opened. An extensive quantitative safety analysis of this design had assumed a low probability of simultaneous failure for the two relief valves, but it ignored the possibility of a design error in the electrical wiring; the probability of the design error was not quantifiable. Many other accidents have involved a similar design flaw, including Three Mile Island.

When the feedback associated with an output is received, the controller must be able to handle the normal response as well as deal with feedback that is missing, too late, too early, or has an unexpected value.

Initializing and Updating the Process Model

Because the process model is used by the controller to determine what control commands to issue and when, the accuracy of the process model with respect to the controlled process is critical. As noted earlier, many software-related losses have resulted from such inconsistencies. STPA will identify which process model variables are critical to safety; the controller design must ensure that the controller receives and processes updates for these variables in a timely manner.

Sometimes normal updating of the process model is done correctly by the controller, but problems arise in initialization at startup and after a temporary shutdown. The process model must reflect the actual process state at initial startup and after a restart. It seems to be common, judging from the number of incidents and accidents that have resulted, for software designers to forget that the world



continues to change even though the software may not be operating. When the computer controlling a process is temporarily shut down, perhaps for maintenance or updating of the software, it may restart with the assumption that the controlled process is still in the state it was when the software was last operating. In addition, assumptions may be made about when the operation of the controller will be started, which may be violated. For example, an assumption may be made that a particular aircraft system will be powered up and initialized before takeoff and appropriate default values used in the process model for that case. In the event it was not started at that time or was shut down and then restarted after takeoff, the default startup values in the process model may not apply and may be hazardous.

Consider the mobile tile-processing robot at the beginning of this chapter. The mobile base may be designed to allow manually retracting the stabilizer legs if an emergency occurs while the robot is servicing the tiles and the robot must be physically moved out of the way. When the robot is restarted, the controller may assume

that the stabilizer legs are still extended and arm movements may be commanded that would violate the safety constraints.

The use of an *unknown* value can assist in protecting against this type of design flaw. At startup and after temporary shutdown, process variables that reflect the state of the controlled process should be initialized with the value *unknown* and updated when new feedback arrives. This procedure will result in resynchronizing the process model and the controlled process state. The control algorithm must also account, of course, for the proper behavior in case it needs to use a process model variable that has the *unknown* value.

Just as timeouts must be specified and handled for basic input processing as described earlier, the maximum time the controller waits until the first input after startup needs to be determined and what to do if this time limit is violated. Once again, while human controllers will likely detect such a problem eventually, such as a failed input channel or one that was not restarted on system startup, computers will patiently wait forever if they are not given instructions to detect such a timeout and to respond to it.

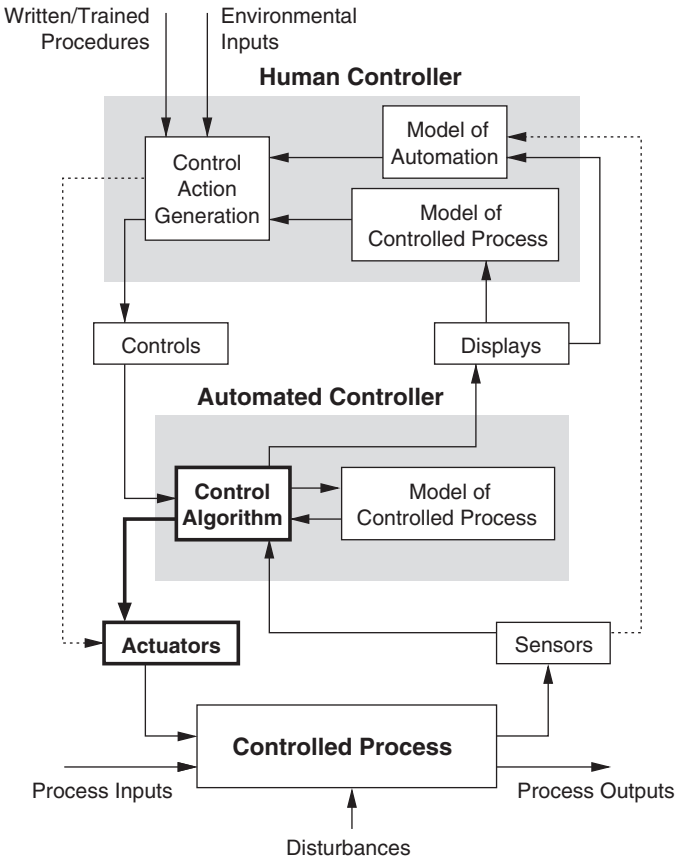
In general, the system and control loop should start in a safe state. Interlocks may need to be initialized or checked to be operational at system startup, including startup after temporarily overriding the interlocks.

Finally the behavior of the controller with respect to input received before startup, after shutdown, or while the controller is temporarily disconnected from the process (offline) must be considered and it must be determined if this information can be safely ignored or how it will be stored and later processed if it cannot. One factor in the loss of an aircraft that took off from the wrong runway at Lexington Airport, for example, is that information about temporary changes in the airport taxiways was not reflected in the airport maps provided to the crew. The information about the changes, which was sent by the National Flight Data Center, was received by the map-provider computers at a time when they were not online, leading to airport charts that did not match the actual state of the airport. The document control system software used by the map provider was designed to only make reports of information received during business hours Monday through Friday [142].

Producing Outputs

The primary responsibility of the process controller is to produce commands to fulfill its control responsibilities. Again, the STPA hazard analysis and safety-guided design process will produce the application-specific behavioral safety requirements and constraints on controller behavior to ensure safety. But some general guidelines are also useful.

One general safety constraint is that the behavior of an automated controller should be deterministic: it should exhibit only one behavior for arrival of any input



in a particular state. While it is easy to design software with nondeterministic behavior and, in some cases, actually has some advantages from a software point of view, nondeterministic behavior makes testing more difficult and, more important, much more difficult for humans to learn how an automated system works and to monitor it. If humans are expected to control or monitor an automated system or an automated controller, then the behavior of the automation should be deterministic.

Just as inputs can arrive faster than they can be processed by the controller, the absorption rate of the actuators and recipients of output from the controller must be considered. Again, the problem usually arises when a fast output device (such as a computer) is providing input to a slower device, such as a human. Contingency action must be designed when the output absorption rate limit is exceeded.

Three additional general considerations in the safe design of controllers are data age, latency, and fault handling.

Data age: No inputs or output commands are valid forever. The control loop design must account for inputs that are no longer valid and should not be used by the controller and for outputs that cannot be executed immediately. All inputs used in the generation of output commands must be properly limited in the time they are used and marked as obsolete once that time limit has been exceeded. At the same time, the design of the control loop must account for outputs that are not executed within a given amount of time. As an example of what can happen when data age is not properly handled in the design, an engineer working in the cockpit of a B-lA aircraft issued a CLOSE WEAPONS BAY DOOR command during a test. At the time, a mechanic working on the door had activated a mechanical inhibit on it. The CLOSE DOOR command was not executed, but it remained active. Several hours later, when the door maintenance was completed, the mechanical inhibit was removed. The door closed unexpectedly, killing the worker [64].

Latency: Latency is the time interval during which receipt of new information cannot change an output even though it arrives prior to the output. While latency time can be reduced by using various types of design techniques, it cannot be eliminated completely. Controllers need to be informed about the arrival of feedback affecting previously issued commands and, if possible, provided with the ability to undo or to mitigate the effects of the now unwanted command.

Fault-handling: Most accidents involve off-nominal processing modes, including startup and shutdown and fault handling. The design of the control loop should assist the controller in handling these modes and the designers need to focus particular attention on them.

The system design may allow for performance degradation and may be designed to fail into safe states or to allow partial shutdown and restart. Any fail-safe behavior that occurs in the process should be reported to the controller. In some cases, automated systems have been designed to fail so gracefully that human controllers are not aware of what is going on until they need to take control and may not be prepared to do so. Also, hysteresis needs to be provided in the control algorithm for transitions between off-nominal and nominal processing modes to avoid *ping-ponging* when the conditions that caused the controlled process to leave the normal state still exist or recur.

Hazardous functions have special requirements. Clearly, interlock failures should result in the halting of the functions they are protecting. In addition, the control algorithm design may differ after failures are detected, depending on whether the controller outputs are hazard-reducing or hazard-increasing. A hazard-increasing output is one that moves the controlled process to a more hazardous state, for example, arming a weapon. A hazard-reducing output is a command that leads to a

reduced risk state, for example, safing a weapon or any other command whose purpose is to maintain safety.

If a failure in the control loop, such as a sensor or actuator, could inhibit the production of a hazard-reducing command, there should be multiple ways to trigger such commands. On the other hand, multiple inputs should be required to trigger commands that can lead to hazardous states so they are not inadvertently issued. Any failure should inhibit the production of a hazard-increasing command. As an example of the latter condition, loss of the ability of the controller to receive input, such as failure of a sensor, that might inhibit the production of a hazardous output should prevent such an output from being issued.

9.4 Special Considerations in Designing for Human Controllers

The design principles in section 9.3 apply when the controller is automated or human, particularly when designing procedures for human controllers to follow. But humans do not always follow procedures, nor should they. We use humans to control systems because of their flexibility and adaptability to changing conditions and to the incorrect assumptions made by the designers. Human error is an inevitable and unavoidable consequence. But appropriate design can assist in reducing human error and increasing safety in human-controlled systems.

Human error is not random. It results from basic human mental abilities and physical skills combined with the features of the tools being used, the tasks assigned, and the operating environment. We can use what is known about human mental abilities and design the other aspects of the system—the tools, the tasks, and the operating environment—to reduce and control human error to a significant degree. The previous section described general principles for safe design. This section focuses on additional design principles that apply when humans control, either directly or indirectly, safety-critical systems.

9.4.1 Easy but Ineffective Approaches

One simple solution for engineers is to simply use human factors checklists. While many such checklists exist, they often do not distinguish among the qualities they enhance, which may not be related to safety and may even conflict with safety. The only way such universal guidelines could be useful is if all design qualities were complementary and achieved in exactly the same way, which is not the case. Qualities are conflicting and require design tradeoffs and decisions about priorities.

Usability and safety, in particular, are often conflicting; an interface that is easy to use may not necessarily be safe. As an example, a common guideline is to ensure that a user must enter data only once and that the computer can access that data if

needed later for the same task or for different tasks [192]. Duplicate entry, however, is required for the computer to detect entry errors unless the errors are so extreme that they violate reasonableness criteria. A small slip usually cannot be detected and such entry errors have led to many accidents. Multiple entry of critical data can prevent such losses.

As another example, a design that involves displaying data or instructions on a screen for an operator to check and verify by pressing the *enter* button minimizes the typing an operator must do. Over time, however, and after few errors are detected, operators will get in the habit of pressing the enter key multiple times in rapid succession. This design feature has been implicated in many losses. For example, the Therac-25 was a linear accelerator that overdosed multiple patients during radiation therapy. In the original Therac-25 design, operators were required to enter the treatment parameters at the treatment site as well as on the computer console. After the operators complained about the duplication, the parameters entered at the treatment site were instead displayed on the console and the operator needed only to press the *return* key if they were correct. Operators soon became accustomed to pushing the return key quickly the required number of times without checking the parameters carefully.

The second easy but not very effective solution is to write procedures for human operators to follow and then assume the engineering job is done. Enforcing the following of procedures is unlikely, however, to lead to a high level of safety.

Dekker notes what he called the "Following Procedures Dilemma" [50]. Operators must balance between adapting procedures in the face of unanticipated conditions versus sticking to procedures rigidly when cues suggest they should be adapted. If human controllers choose the former, that is, they adapt procedures when it appears the procedures are wrong, a loss may result when the human controller does not have complete knowledge of the circumstances or system state. In this case, the humans will be blamed for deviations and nonadherence to the procedures. On the other hand, if they stick to procedures (the control algorithm provided) rigidly when the procedures turn out to be wrong, they will be blamed for their inflexibility and the application of the rules in the wrong context. Hindsight bias is often involved in identifying what the operator should have known and done.

Insisting that operators always follow procedures does not guarantee safety although it does usually guarantee that there is someone to blame—either for following the procedures or for not following them—when things go wrong. Safety comes from controllers being skillful in judging when and how procedures apply. As discussed in chapter 12, organizations need to monitor adherence to procedures not simply to enforce compliance but to understand how and why the gap between procedures and practice grows and to use that information to redesign both the system and the procedures [50].

Section 8.5 of chapter 8 describes important differences between human and automated controllers. One of these differences is that the control algorithm used by humans is dynamic. This dynamic aspect of human control is why humans are kept in systems. They provide the flexibility to deviate from procedures when it turns out the assumptions underlying the engineering design are wrong. But with this flexibility comes the possibility of unsafe changes in the dynamic control algorithm and raises new design requirements for engineers and system designers to understand the reason for such unsafe changes and prevent them through appropriate system design.

Just as engineers have the responsibility to understand the hazards in the physical systems they are designing and to control and mitigate them, engineers also must understand how their system designs can lead to human error and how they can design to reduce errors.

Designing to prevent human error requires some basic understanding about the role humans play in systems and about human error.

9.4.2 The Role of Humans in Control Systems

Humans can play a variety of roles in a control system. In the simplest cases, they create the control commands and apply them directly to the controlled process. For a variety of reasons, particularly speed and efficiency, the system may be designed with a computer between the human controller and the system. The computer may exist only in the feedback loop to process and present data to the human operator. In other systems, the computer actually issues the control instructions with the human operator either providing high-level supervision of the computer or simply monitoring the computer to detect errors or problems.

An unanswered question is what is the best role for humans in safety-critical process control. There are three choices beyond direct control: the human can monitor an automated control system, the human can act as a backup to the automation, or the human and automation can both participate in the control through some type of partnership. These choices are discussed in depth in *Safeware* and are only summarized here.

Unfortunately for the first option, humans make very poor monitors. They cannot sit and watch something without active control duties for any length of time and maintain vigilance. Tasks that require little active operator behavior may result in lowered alertness and can lead to complacency and overreliance on the automation. Complacency and lowered vigilance are exacerbated by the high reliability and low failure rate of automated systems.

But even if humans could remain vigilant while simply sitting and monitoring a computer that is performing the control tasks (and usually doing the right thing), Bainbridge has noted the irony that automatic control systems are installed because

they can do the job better than humans, but then humans are assigned the task of monitoring the automated system [14]. Two questions arise:

1. The human monitor needs to know what the correct behavior of the controlled or monitored process should be; however, in complex modes of operation—for example, where the variables in the process have to follow a particular trajectory over time—evaluating whether the automated control system is performing correctly requires special displays and information that may only be available from the automated system being monitored. How will human monitors know when the computer is wrong if the only information they have comes from that computer? In addition, the information provided by an automated controller is more indirect, which may make it harder for humans to get a clear picture of the system: Failures may be silent or masked by the automation.
2. If the decisions can be specified fully, then a computer can make them more quickly and accurately than a human. How can humans monitor such a system? Whitfield and Ord found that, for example, air traffic controllers' appreciation of the traffic situation was reduced at the high traffic levels made feasible by using computers [198]. In such circumstances, humans must monitor the automated controller at some metalevel, deciding whether the computer's decisions are acceptable rather than completely correct. In case of a disagreement, should the human or the computer be the final arbiter?

Employing humans as backups is equally ineffective. Controllers need to have accurate process models to control effectively, but not being in active control leads to a degradation of their process models. At the time they need to intervene, it may take a while to "get their bearings"—in other words, to update their process models so that effective and safe control commands can be given. In addition, controllers need both manual and cognitive skills, but both of these decline in the absence of practice. If human backups need to take over control from automated systems, they may be unable to do so effectively and safely. Computers are often introduced into safety-critical control loops because they increase system reliability, but at the same time, that high reliability can provide little opportunity for human controllers to practice and maintain the skills and knowledge required to intervene when problems *do* occur.

It appears, at least for now, that humans will have to provide direct control or will have to share control with automation unless adequate confidence can be established in the automation to justify eliminating monitors completely. Few systems exist today where such confidence can be achieved when safety is at stake. The problem then becomes one of finding the correct partnership and allocation of tasks between humans and computers. Unfortunately, this problem has not been solved, although some guidelines are presented later.

One of the things that make the problem difficult is that it is not just a matter of splitting responsibilities. Computer control is changing the cognitive demands on human controllers. Humans are increasingly supervising a computer rather than directly monitoring the process, leading to more cognitively complex decision making. Automation logic complexity and the proliferation of control modes are confusing humans. In addition, whenever there are multiple controllers, the requirements for cooperation and communication are increased, not only between the human and the computer but also between humans interacting with the same computer, for example, the need for coordination among multiple people making entries to the computer. The consequences can be increased memory demands, new skill and knowledge requirements, and new difficulties in the updating of the human's process models.

A basic question that must be answered and implemented in the design is who will have the final authority if the human and computers disagree about the proper control actions. In the loss of an Airbus 320 while landing at Warsaw in 1993, one of the factors was that the automated system prevented the pilots from activating the braking system until it was too late to prevent crashing into a bank built at the end of the runway. This automation feature was a protection device included to prevent the reverse thrusters accidentally being deployed in flight, a presumed cause of a previous accident. For a variety of reasons, including water on the runway causing the aircraft wheels to hydroplane, the criteria used by the software logic to determine that the aircraft had landed were not satisfied by the feedback received by the automation [133]. Other incidents have occurred where the pilots have been confused about who is in control, the pilot or the automation, and found themselves fighting the automation [181].

One common design mistake is to set a goal of automating everything and then leaving some miscellaneous tasks that are difficult to automate for the human controllers to perform. The result is that the operator is left with an arbitrary collection of tasks for which little thought was given to providing support, particularly support for maintaining accurate process models. The remaining tasks may, as a consequence, be significantly more complex and error-prone. New tasks may be added, such as maintenance and monitoring, that introduce new types of errors. Partial automation, in fact, may not reduce operator workload but merely change the type of demands on the operator, leading to potentially increased workload. For example, cockpit automation may increase the demands on the pilots by creating a lot of data entry tasks during approach when there is already a lot to do. These automation interaction tasks also create "heads down" work at a time when increased monitoring of nearby traffic is necessary.

By taking away the easy parts of the operator's job, automation may make the more difficult ones even harder [14]. One causal factor here is that taking away or

changing some operator tasks may make it difficult or even impossible for the operators to receive the feedback necessary to maintain accurate process models.

When designing the automation, these factors need to be considered. A basic design principle is that automation should be designed to augment human abilities, not replace them, that is, to aid the operator, not to take over.

To design safe automated controllers with humans in the loop, designers need some basic knowledge about human error related to control tasks. In fact, Rasmussen has suggested that the term *human error* be replaced by considering such events as *human–task mismatches*.

9.4.3 Human Error Fundamentals

Human error can be divided into the general categories of slips and mistakes [143, 144]. Basic to the difference is the concept of *intention* or desired action. A *mistake* is an error in the intention, that is, an error that occurs during the planning of an action. A *slip*, on the other hand, is an error in carrying out the intention. As an example, suppose an operator decides to push button A. If the operator instead pushes button B, then it would be called a slip because the action did not match the intention. If the operator pushed A (carries out the intention correctly), but it turns out that the intention was wrong, that is, button A should *not* have been pushed, then this is called a mistake.

Designing to prevent slips involves applying different principles than designing to prevent mistakes. For example, making controls look very different or placing them far apart from each other may reduce slips, but not mistakes. In general, designing to reduce mistakes is more difficult than reducing slips, which is relatively straightforward.

One of the difficulties in eliminating planning errors or mistakes is that such errors are often only visible in hindsight. With the information available at the time, the decisions may seem reasonable. In addition, planning errors are a necessary side effect of human problem-solving ability. Completely eliminating mistakes or planning errors (if possible) would also eliminate the need for humans as controllers.

Planning errors arise from the basic human cognitive ability to solve problems. Human error in one situation is human ingenuity in another. Human problem solving rests on several unique human capabilities, one of which is the ability to create hypotheses and to test them and thus create new solutions to problems not previously considered. These hypotheses, however, may be wrong. Rasmussen has suggested that human error is often simply unsuccessful experiments in an unkind environment, where an unkind environment is defined as one in which it is not possible for the human to correct the effects of inappropriate variations in performance

before they lead to unacceptable consequences [166]. He concludes that human performance is a balance between a desire to optimize skills and a willingness to accept the risk of exploratory acts.

A second basic human approach to problem solving is to try solutions that worked in other circumstances for similar problems. Once again, this approach is not always successful but the inapplicability of old solutions or plans (learned procedures) may not be determinable without the benefit of hindsight.

The ability to use these problem-solving methods provides the advantages of human controllers over automated controllers, but success is not assured. Designers, if they understand the limitations of human problem solving, can provide assistance in the design to avoid common pitfalls and enhance human problem solving. For example, they may provide ways for operators to obtain extra information or to test hypotheses safely. At the same time, there are some additional basic human cognitive characteristics that must be considered.

Hypothesis testing can be described in terms of basic feedback control concepts. Using the information in the process model, the controller generates a hypothesis about the controlled process. A test composed of control actions is created to generate feedback useful in evaluating the hypothesis, which in turn is used to update the process model and the hypothesis.

When controllers have no accurate diagnosis of a problem, they must make provisional assessments of what is going on based on uncertain, incomplete, and often contradictory information [50]. That provisional assessment will guide their information gathering, but it may also lead to over attention to confirmatory evidence when processing feedback and updating process models while, at the same time, discounting information that contradicts their current diagnosis. Psychologists call this phenomenon *cognitive fixation*. The alternative is called *thematic vagabonding*, where the controller jumps around from explanation to explanation, driven by the loudest or latest feedback or alarm and never develops a coherent assessment of what is going on. Only hindsight can determine whether the controller should have abandoned one explanation for another: Sticking to one assessment can lead to more progress in many situations than jumping around and not pursuing a consistent planning process.

Plan continuation is another characteristic of human problem solving related to cognitive fixation. Commitment to a preliminary diagnosis can lead to sticking with the original plan even though the situation has changed and calls for a different plan. Orisanu [149] notes that early cues that suggest an initial plan is correct are usually very strong and unambiguous, helping to convince people to continue the plan. Later feedback that suggests the plan should be abandoned is typically more ambiguous and weaker. Conditions may deteriorate gradually. Even when

controllers receive and acknowledge this feedback, the new information may not change their plan, especially if abandoning the plan is costly in terms of organizational and economic consequences. In the latter case, it is not surprising that controllers will seek and focus on confirmatory evidence and will need a lot of contradictory evidence to justify changing their plan.

Cognitive fixation and plan continuation are compounded by stress and fatigue. These two factors make it more difficult for controllers to juggle multiple hypotheses about a problem or to project a situation into the future by mentally simulating the effects of alternative plans [50].

Automated tools can be designed to assist the controller in planning and decision making, but they must embody an understanding of these basic cognitive limitations and assist human controllers in overcoming them. At the same time, care must be taken that any simulation or other planning tools to assist human problem solving do not rest on the same incorrect assumptions about the system that led to the problems in the first place.

Another useful distinction is between errors of omission and errors of commission. Sarter and Woods [181] note that in older, less complex aircraft cockpits, most pilot errors were *errors of commission* that occurred as a result of a pilot control action. Because the controller, in this case the pilot, took a direct action, he or she is likely to check that the intended effect of the action has actually occurred. The short feedback loops allow the operators to repair most errors before serious consequences result. This type of error is still the prevalent one for relatively simple devices.

In contrast, studies of more advanced automation in aircraft find that *errors of omission* are the dominant form of error [181]. Here the controller does not implement a control action that is required. The operator may not notice that the automation has done something because that automation behavior was not explicitly invoked by an operator action. Because the behavioral changes are not expected, the human controller is less likely to pay attention to relevant indications and feedback, particularly during periods of high workload.

Errors of omission are related to the change of human roles in systems from direct controllers to monitors, exception handlers, and supervisors of automated controllers. As their roles change, the cognitive demands may not be reduced but instead may change in their basic nature. The changes tend to be more prevalent at high-tempo and high-criticality periods. So while some types of human errors have declined, new types of errors have been introduced.

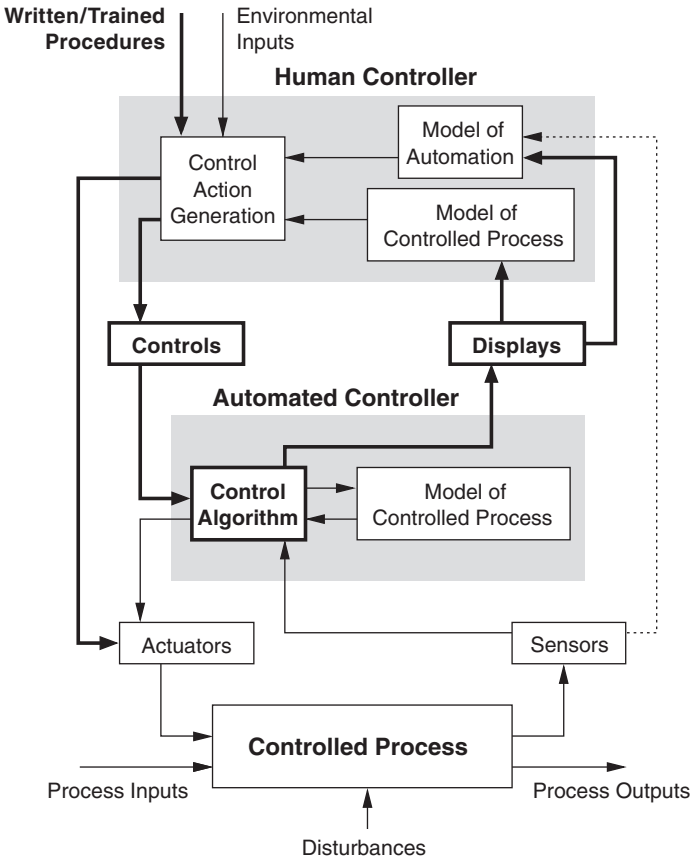
The difficulty and perhaps impossibility of eliminating human error does not mean that greatly improved system design in this respect is not possible. System design can be used to take advantage of human cognitive capabilities and to minimize the errors that may result from them. The rest of the chapter provides some

principles to create designs that better support humans in controlling safety-critical processes and reduce human errors.

9.4.4 Providing Control Options

If the system design goal is to make humans responsible for safety in control systems, then they must have adequate flexibility to cope with undesired and unsafe behavior and not be constrained by inadequate control options. Three general design principles apply: design for redundancy, design for incremental control, and design for error tolerance.

Design for redundant paths: One helpful design feature is to provide multiple physical devices and logical paths to ensure that a single hardware failure or software error cannot prevent the operator from taking action to maintain a safe system state and avoid hazards. There should also be multiple ways to change



from an unsafe to a safe state, but only one way to change from an unsafe to a safe state.

Design for incremental control: Incremental control makes a system easier to control, both for humans and computers, by performing critical steps incrementally rather than in one control action. The common use of incremental *arm*, *aim*, *fire* sequences is an example. The controller should have the ability to observe the system and get feedback to test the validity of the assumptions and models upon which the decisions are made. The system design should also provide the controller with compensating control actions to allow modifying or aborting previous control actions before significant damage is done. An important consideration in designing for controllability in general is to lower the time pressures on the controllers, if possible.

The design of incremental control algorithms can become complex when a human controller is controlling a computer, which is controlling the actual physical process, in a stressful and busy environment, such as a military aircraft. If one of the commands in an incremental control sequence cannot be executed within a specified period of time, the human operator needs to be informed about any delay or postponement or the entire sequence should be canceled and the operator informed. At the same time, interrupting the pilot with a lot of messages that may not be critical at a busy time could also be dangerous. Careful analysis is required to determine when multistep controller inputs can be preempted or interrupted before they are complete and when feedback should occur that this happened [90].

Design for error tolerance: Rasmussen notes that people make errors all the time, but we are able to detect and correct them before adverse consequences occur [165]. System design can limit people's ability to detect and recover from their errors. He defined a system design goal of *error tolerant systems*. In these systems, errors are observable (within an appropriate time limit) and they are reversible before unacceptable consequences occur. The same applies to computer errors: they should be observable and reversible.

The general goal is to allow controllers to monitor their own performance. To achieve this goal, the system design needs to:

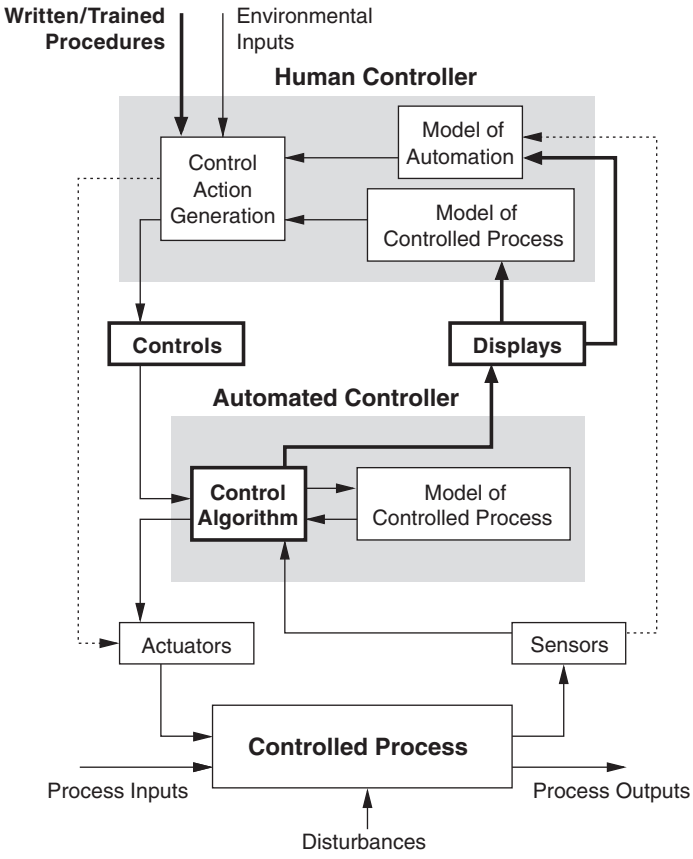
1. Help operators monitor their actions and recover from errors.
2. Provide feedback about actions operators took and their effects, in case the actions were inadvertent. Common examples are echoing back operator inputs or requiring confirmation of intent.
3. Allow for recovery from erroneous actions. The system should provide control options, such as compensating or reversing actions, and enough time for recovery actions to be taken before adverse consequences result.

Incremental control, as described earlier, is a type of error-tolerant design technique.

9.4.5 Matching Tasks to Human Characteristics

In general, the designer should tailor systems to human requirements instead of the opposite. Engineered systems are easier to change in their behavior than are humans.

Because humans without direct control tasks will lose vigilance, the design should combat lack of alertness by designing human tasks to be stimulating and varied, to provide good feedback, and to require active involvement of the human controllers in most operations. Maintaining manual involvement is important, not just for alertness but also in getting the information needed to update process models.



Maintaining active engagement in the tasks means that designers must distinguish between providing help to human controllers and taking over. The human tasks should not be oversimplified and tasks involving passive or repetitive actions should be minimized. Allowing latitude in how tasks are accomplished will not only reduce monotony and error proneness, but can introduce flexibility to assist operators in improvising when a problem cannot be solved by only a limited set of behaviors. Many accidents have been avoided when operators jury-rigged devices or improvised procedures to cope with unexpected events. Physical failures may cause some paths to become nonfunctional and flexibility in achieving goals can provide alternatives.

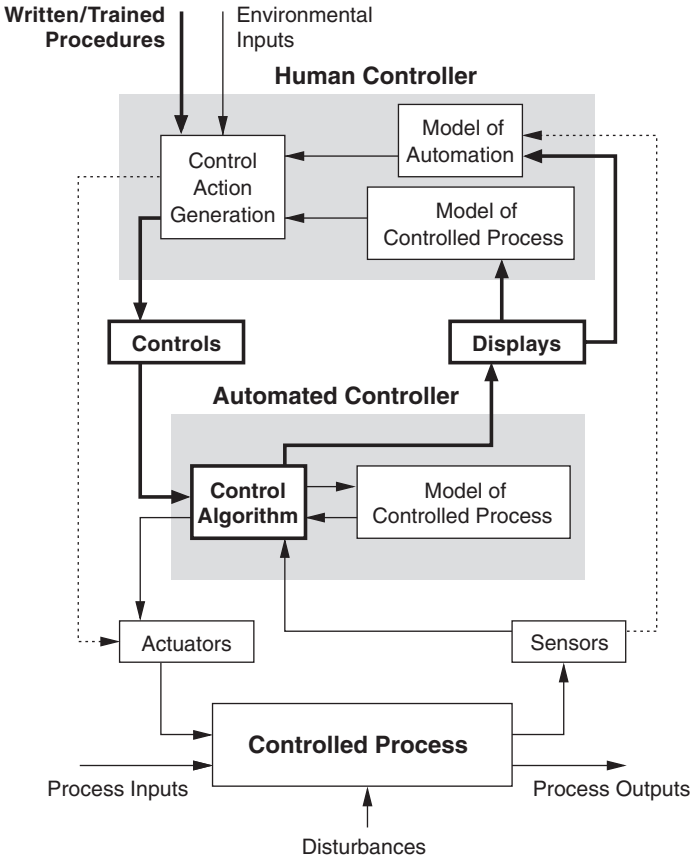
Designs should also be avoided that require or encourage *management by exception*, which occurs when controllers wait for alarm signals before taking action. Management by exception does not allow controllers to prevent disturbances by looking for early warnings and trends in the process state. For operators to anticipate undesired events, they need to continuously update their process models. Experiments by Swaanenburg and colleagues found that management by exception is not the strategy adopted by human controllers as their normal supervisory mode [196]. Avoiding management by exception requires active involvement in the control task and adequate feedback to update process models. A display that provides only an overview and no detailed information about the process state, for example, may not provide the information necessary for detecting imminent alarm conditions.

Finally, if designers expect operators to react correctly to emergencies, they need to design to support them in these tasks and to help fight some basic human tendencies described previously such as cognitive fixation and plan continuation. The system design should support human controllers in decision making and planning activities during emergencies.

9.4.6 Designing to Reduce Common Human Errors

Some human errors are so common and unnecessary that there is little excuse for not designing to prevent them. Care must be taken though that the attempt to reduce erroneous actions does not prevent the human controller from intervening in an emergency when the assumptions made during design about what should and should not be done turn out to be incorrect.

One fundamental design goal is to make safety-enhancing actions easy, natural, and difficult to omit or do wrong. In general, the design should make it more difficult for the human controller to operate unsafely than safely. If safety-enhancing actions are easy, they are less likely to be bypassed intentionally or accidentally. Stopping an unsafe action or leaving an unsafe state should be possible with a single keystroke that moves the system into a safe state. The design should make fail-safe actions easy and natural, and difficult to avoid, omit, or do wrong.



In contrast, two or more unique operator actions should be required to start any potentially hazardous function or sequence of functions. Hazardous actions should be designed to minimize the potential for inadvertent activation; they should not, for example, be initiated by pushing a single key or button (see the preceding discussion of incremental control).

The general design goal should be to enhance the ability of the human controller to act safely while making it more difficult to behave unsafely. Initiating a potentially unsafe process change, such as a spacecraft launch, should require multiple keystrokes or actions while stopping a launch should require only one.

Safety may be enhanced by using procedural safeguards, where the operator is instructed to take or avoid specific actions, or by designing safeguards into the system. The latter is much more effective. For example, if the potential error involves leaving out a critical action, either the operator can be instructed to always take that action or the action can be made an integral part of the process. A typical error

during maintenance is not to return equipment (such as safety interlocks) to the operational mode. The accident sequence at Three Mile Island was initiated by such an error. An action that is isolated and has no immediate relation to the "gestalt" of the repair or testing task is easily forgotten. Instead of stressing the need to be careful (the usual approach), change the system by integrating the act physically into the task, make detection a physical consequence of the tool design, or change operations planning or review. That is, change design or management rather than trying to change the human [162].

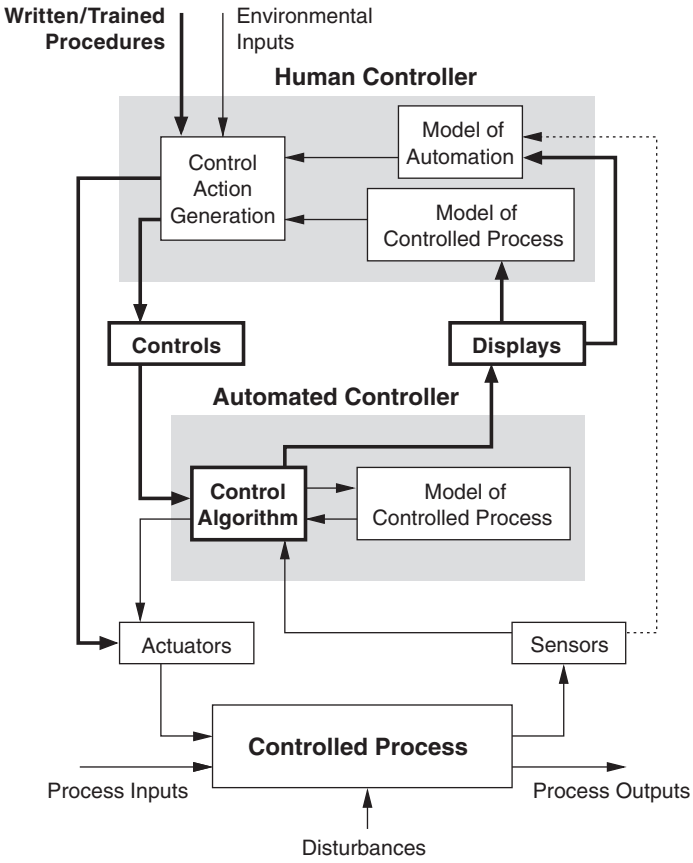
To enhance decision making, references should be provided for making judgments, such as marking meters with safe and unsafe limits. Because humans often revert to stereotype and cultural norms, such norms should be followed in design. Keeping things simple, natural, and similar to what has been done before (not making gratuitous design changes) is a good way to avoid errors when humans are working under stress, are distracted, or are performing tasks while thinking about something else.

To assist in preventing sequencing errors, controls should be placed in the sequence in which they are to be used. At the same time, similarity, proximity, interference, or awkward location of critical controls should be avoided. Where operators have to perform different classes or types of control actions, sequences should be made as dissimilar as possible.

Finally, one of the most effective design techniques for reducing human error is to design so that the error is not physically possible or so that errors are obvious. For example, valves can be designed so they cannot be interchanged by making the connections different sizes or preventing assembly errors by using asymmetric or male and female connections. Connection errors can also be made obvious by color coding. Amazingly, in spite of hundreds of deaths due to misconnected tubes in hospitals that have occurred over decades, such as a feeding tube inadvertently connected to a tube that is inserted in a patient's vein, regulators, hospitals, and tube manufacturers have taken no action to implement this standard safety design technique [80].

9.4.7 Support in Creating and Maintaining Accurate Process Models

Human controllers who are supervising automation have two process models to maintain: one for the process being controlled by the automation and one for the automated controller itself. The design should support human controllers in maintaining both of these models. An appropriate goal here is to provide humans with the facilities to experiment and learn about the systems they are controlling, either directly or indirectly. Operators should also be allowed to maintain manual involvement to update process models, to maintain skills, and to preserve self-confidence. Simply observing will degrade human supervisory skills and confidence.



When human controllers are supervising automated controllers, the automation has extra design requirements. The control algorithm used by the automation must be learnable and understandable. Two common design flaws in automated controllers are inconsistent behavior by the automation and unintended side effects.

Inconsistent Behavior

Carroll and Olson define a consistent design as one where a similar task or goal is associated with similar or identical actions [35]. Consistent behavior on the part of the automated controller makes it easier for the human providing supervisory control to learn how the automation works, to build an appropriate process model for it, and to anticipate its behavior.

An example of inconsistency, detected in an A320 simulator study, involved an aircraft go-around below 100 feet above ground level. Sarter and Woods found that pilots failed to anticipate and realize that the autothrust system did not arm when

they selected takeoff/go-around (TOGA) power under these conditions because it did so under all other circumstances where TOGA power is applied [181].

Another example of inconsistent automation behavior, which was implicated in an A320 accident, is a protection function that is provided in all automation configurations except the specific mode (in this case altitude acquisition) in which the autopilot was operating [181].

Human factors for critical systems have most extensively been studied in aircraft cockpit design. Studies have found that consistency is most important in high-tempo, highly dynamic phases of flight where pilots have to rely on their automatic systems to work as expected without constant monitoring. Even in more low-pressure situations, consistency (or predictability) is important in light of the evidence from pilot surveys that their normal monitoring behavior may change on high-tech flight decks [181].

Pilots on conventional aircraft use a highly trained instrument-scanning pattern of recurrently sampling a given set of basic flight parameters. In contrast, some A320 pilots report that they no longer scan anymore but allocate their attention within and across cockpit displays on the basis of expected automation states and behaviors. Parameters that are not expected to change may be neglected for a long time [181]. If the automation behavior is not consistent, errors of omission may occur where the pilot does not intervene when necessary.

In section 9.3.2, determinism was identified as a safety design feature for automated controllers. Consistency, however, requires more than deterministic behavior. If the operator provides the same inputs but different outputs (behaviors) result for some reason other than what the operator has done (or may even know about), then the behavior is inconsistent from the operator viewpoint even though it is deterministic. While the designers may have good reasons for including inconsistent behavior in the automated controller, there should be a careful tradeoff made with the potential hazards that could result.

Unintended Side Effects

Incorrect process models can result when an action intended to have one effect has an additional side effect not easily anticipated by the human controller. An example occurred in the Sarter and Woods A320 aircraft simulator study cited earlier. Because the approach to the destination airport is such a busy time for the pilots and the automation requires so much heads down work, pilots often program the automation as soon as the air traffic controllers assign them a runway. Sarter and Woods found that the experienced pilots in their study were not aware that entering a runway change *after* entering data for the assigned approach results in the deletion by the automation of all the previously entered altitude and speed constraints, even though they may still apply.

Once again, there may be good reason for the automation designers to include such side effects, but they need to consider the potential for human error that can result.

Mode Confusion

Modes define mutually exclusive sets of automation behaviors. Modes can be used to determine how to interpret inputs or to define required controller behavior. Four general types of modes are common: controller operating modes, supervisory modes, display modes, and controlled process modes.

Controller operating modes define sets of related behavior in the controller, such as shutdown, nominal behavior, and fault-handling.

Supervisory modes determine who or what is controlling the component at any time when multiple supervisors can assume control responsibilities. For example, a flight guidance system in an aircraft may be issued direct commands by the pilot(s) or by another computer that is itself being supervised by the pilot(s). The movement controller in the thermal tile processing system might be designed to be in either manual supervisory mode (by a human controller) or automated mode (by the TTPS task controller). Coordination of control actions among multiple supervisors can be defined in terms of these supervisory modes. Confusion about the current supervisory mode can lead to hazardous system behavior.

A third type of common mode is a *display mode*. The display mode will affect the information provided on the display and how the user interprets that information.

A final type of mode is the operating mode of the controlled process. For example, the mobile thermal tile processing robot may be in a moving mode (between work areas) or in a work mode (in a work area and servicing tiles, during which time it may be controlled by a different controller). The value of this mode may determine whether various operations—for example, extending the stabilizer legs or the manipulator arm—are safe.

Early automated systems had a fairly small number of independent modes. They provided a passive background on which the operator would act by entering target data and requesting system operations. They also had only one overall mode setting for each function performed. Indications of currently active mode and of transitions between modes could be dedicated to one location on the display.

The consequences of breakdown in mode awareness were fairly small in these system designs. Operators seemed able to detect and recover from erroneous actions relatively quickly before serious problems resulted. Sarter and Woods conclude that, in most cases, mode confusion in these simpler systems are associated with errors of commission, that is, with errors that require a controller action in order for the problem to occur [181]. Because the human controller has taken an explicit action,

he or she is likely to check that the intended effect of the action has actually occurred. The short feedback loops allow the controller to repair most errors quickly, as noted earlier.

The flexibility of advanced automation allows designers to develop more complicated, mode-rich systems. The result is numerous mode indications often spread over multiple displays, each containing just that portion of mode status data corresponding to a particular system or subsystem. The designs also allow for interactions across modes. The increased capabilities of automation can, in addition, lead to increased delays between user input and feedback about system behavior.

These new mode-rich systems increase the need for and difficulty of maintaining mode awareness, which can be defined in STAMP terms as keeping the controlled-system operating mode in the controller's process model consistent with the actual controlled system mode. A large number of modes challenges human ability to maintain awareness of active modes, armed modes, interactions between environmental status and mode behavior, and interactions across modes. It also increases the difficulty of error or failure detection and recovery.

Calling for systems with fewer or less complex modes is probably unrealistic. Simplifying modes and automation behavior often requires tradeoffs with precision or efficiency and with marketing demands from a diverse set of customers [181]. Systems with accidental (unnecessary) complexity, however, can be redesigned to reduce the potential for human error without sacrificing system capabilities. Where tradeoffs with desired goals are required to eliminate potential mode confusion errors, system and interface design, informed by hazard analysis, can help find solutions that require the fewest tradeoffs. For example, accidents most often occur during transitions between modes, particularly normal and nonnormal modes, so they should have more stringent design constraints applied to them.

Understanding more about particular types of mode confusion errors can assist with design. Two common types leading to problems are interface interpretation modes and indirect mode changes.

Interface Interpretation Mode Confusion: Interface mode errors are the classic form of mode confusion error:

1. *Input-related errors:* The software interprets user-entered values differently than intended.
2. *Output-related errors:* The software maps multiple conditions onto the same output, depending on the active controller mode, and the operator interprets the interface incorrectly.

A common example of an input interface interpretation error occurs with many word processors where the user may think they are in INSERT mode but instead they

are in INSERT AND DELETE mode or in COMMAND mode and their input is interpreted in a different way and results in different behavior than they intended.

A more complex example occurred in what is believed to be a cause of an A320 aircraft accident. The crew directed the automated system to fly in the TRACK/FLIGHT PATH ANGLE mode, which is a combined mode related to both lateral (TRACK) and vertical (FLIGHT PATH ANGLE) navigation:

> When they were given radar vectors by the air traffic controller, they may have switched from the TRACK to the HDG SEL mode to be able to enter the heading requested by the controller. However, pushing the button to change the lateral mode also automatically changes the vertical mode from FLIGHT PATH ANGLE to VERTICAL SPEED—the mode switch button affects both lateral and vertical navigation. When the pilots subsequently entered "33" to select the desired flight path angle of 3.3 degrees, the automation interpreted their input as a desired vertical speed of 3300 ft. This was not intended by the pilots who were not aware of the active "interface mode" and failed to detect the problem. As a consequence of the too steep descent, the airplane crashed into a mountain [181].

An example of an output interface mode problem was identified by Cook et al. [41] in a medical operating room device with two operating modes: warmup and normal. The device starts in warmup mode when turned on and changes from normal mode to warmup mode whenever either of two particular settings is adjusted by the operator. The meaning of alarm messages and the effect of controls are different in these two modes, but neither the current device operating mode nor a change in mode is indicated to the operator. In addition, four distinct alarm-triggering conditions are mapped onto two alarm messages so that the same message has different meanings depending on the operating mode. In order to understand what internal condition triggered the message, the operator must infer which malfunction is being indicated by the alarm.

Several design constraints can assist in reducing interface interpretation errors. At a minimum, any mode used to control interpretation of the supervisory interface should be annunciated to the supervisor. More generally, the current operating mode of the automation should be displayed at all times. In addition, any change of operating mode should trigger a change in the current operating mode reflected in the interface and thus displayed to the operator, that is, the annunciated mode must be consistent with the internal mode.

A stronger design choice, but perhaps less desirable for various reasons, might be not to condition the interpretation of the supervisory interface on modes at all. Another possibility is to simplify the relationships between modes, for example in the A320, the lateral and vertical modes might be separated with respect to the heading select mode. Other alternatives are to make the required inputs different to lessen confusion (such as 3.3 and 3,300 rather than 33), or the mode indicator on the control panel could be made clearer as to the current mode. While simply

annunciating the mode may be adequate in some cases, annunciations can easily to missed for a variety of reasons and additional design features should be considered.

Mode Confusion Arising from Indirect Mode Changes: Indirect mode changes occur when the automation changes mode without an explicit instruction or direct command by the operator. Such transitions may be triggered on conditions in the automation, such as preprogrammed envelope protection. They may also result from sensor input to the computer about the state of the computer-controlled process, such as achievement of a preprogrammed target or an armed mode with a preselected mode transition. An example of the latter is a mode in which the autopilot might command leveling off of the plane once a particular altitude is reached: the operating mode of the aircraft (leveling off) is changed when the altitude is reached without a direct command to do so by the pilot. In general, the problem occurs when activating one mode can result in the activation of different modes depending on the system status at the time.

There are four ways to trigger a mode change:

1. The automation supervisor explicitly selects a new mode.
2. The automation supervisor enters data (such as a target altitude) or a command that leads to a mode change:
   a. Under all conditions.
   b. When the automation is in a particular state
   c. When the automation's controlled system model or environment is in a particular state.
3. The automation supervisor does not do anything, but the automation logic changes mode as a result of a change in the system it is controlling.
4. The automation supervisor selects a mode change but the automation does something else, either because of the state of the automation at the time or the state of the controlled system.

Again, errors related to mode confusion are related to problems that human supervisors of automated controllers have in maintaining accurate process models. Changes in human controller behavior in highly automated systems, such as the changes in pilot scanning behavior described earlier, are also related to these types of mode confusion error.

Behavioral expectations about the automated controller behavior are formed based on the human supervisors' knowledge of the input to the automation and on their process models of the automation. Gaps or misconceptions in this model

may interfere with predicting and tracking indirect mode transitions or with understanding the interactions among modes.

An example of an accident that has been attributed to an indirect mode change occurred while an A320 was landing in Bangalore, India [182]. The pilot's selection of a lower altitude while the automation was in the ALTITUDE ACQUISITION mode resulted in the activation of the OPEN DESCENT mode, where speed is controlled only by the pitch of the aircraft and the throttles go to idle. In that mode, the automation ignores any preprogrammed altitude constraints. To maintain pilot-selected speed without power, the automation had to use an excessive rate of descent, which led to the aircraft crashing short of the runway.

Understanding how this could happen is instructive in understanding just how complex mode logic can get. There are three different ways to activate OPEN DESCENT mode on the A320:

1. Pull the altitude knob after selecting a lower altitude.
2. Pull the speed knob when the aircraft is in EXPEDITE mode.
3. Select a lower altitude while in ALTITUDE ACQUISITION mode.

It was the third condition that is suspected to have occurred. The pilot must not have been aware the aircraft was within 200 feet of the previously entered target altitude, which triggers ALTITUDE ACQUISITION mode. He therefore may not have expected selection of a lower altitude at that time to result in a mode transition and did not closely monitor his mode annunciations during this high workload time. He discovered what happened ten seconds before impact, but that was too late to recover with the engines at idle [182].

Other factors contributed to his not discovering the problem until too late, one of which is the problem in maintaining consistent process models when there are multiple controllers as discussed in the next section. The pilot flying (PF) had disengaged his flight director[1] during approach and was assuming the pilot not flying (PNF) would do the same. The result would have been a mode configuration in which airspeed is automatically controlled by the autothrottle (the SPEED mode), which is the recommended procedure for the approach phase of flight. The PNF never turned off his flight director, however, and the OPEN DESCENT mode became active when a lower altitude was selected. This indirect mode change led to the hazardous state and eventually the accident, as noted earlier. But a complicating factor was that each pilot only received an indication of the status of his own flight

1. The flight director is automation that gives visual cues to the pilot via an easily interpreted display of the aircraft's flight path. The preprogrammed path, automatically computed, furnishes the steering commands necessary to obtain and hold a desired path.

director and not all the information necessary to determine whether the desired mode would be engaged. The lack of feedback and resulting incomplete knowledge of the aircraft state (incorrect aircraft process model) contributed to the pilots not detecting the unsafe state in time to correct it.

Indirect mode transitions can be identified in software designs. What to do in response to identifying them or deciding not to include them in the first place is more problematic and the tradeoffs and mitigating design features must be considered for each particular system. The decision is just one of the many involving the benefits of complexity in system design versus the hazards that can result.

Coordination of Multiple Controller Process Models

When multiple controllers are engaging in coordinated control of a process, inconsistency between their process models can lead to hazardous control actions. Careful design of communication channels and coordinated activity is required. In aircraft, this coordination, called crew resource management, is accomplished through careful design of the roles of each controller to enhance communication and to ensure consistency among their process models.

A special case of this problem occurs when one human controller takes over for another. The handoff of information about both the state of the controlled process and any automation being supervised by the human must be carefully designed.

Thomas describes an incident involving loss of communication for an extended time between ground air traffic control and an aircraft [199]. In this incident, a ground controller had taken over after a controller shift change. Aircraft are passed from one air traffic control sector to another through a carefully designed set of exchanges, called a *handoff*, during which the aircraft is told to switch to the radio frequency for the new sector. When, after a shift change the new controller gave an instruction to a particular aircraft and received no acknowledgment, the controller decided to take no further action; she assumed that the lack of acknowledgment was an indication that the aircraft had already switched to the new sector and was talking to the next controller.

Process model coordination during shift changes is partially controlled in a *position relief briefing*. This briefing normally covers all aircraft that are currently on the correct radio frequency or have not checked in yet. When the particular flight in question was not mentioned in the briefing, the new controller interpreted that as meaning that the aircraft was no longer being controlled by this station. She did not call the next controller to verify this status because the aircraft had not been mentioned in the briefing.

The design of the air traffic control system includes redundancy to try to avoid errors—if the aircraft does not check in with the next controller, then that controller

would call her. When she saw the aircraft (on her display) leave her airspace and no such call was received, she interpreted that as another indication that the aircraft was indeed talking to the next controller.

A final factor implicated in the loss of communication was that when the new controller took over, there was little traffic at the aircraft's altitude and no danger of collision. Common practice for controllers in this situation is to initiate an early handoff to the next controller. So although the aircraft was only halfway through her sector, the new controller assumed an early handoff had occurred.

An additional causal factor in this incident involves the way controllers track which aircraft have checked in and which have already been handed off to the next controller. The old system was based on printed flight progress strips and included a requirement to mark the strip when an aircraft had checked in. The new system uses electronic flight progress strips to display the same information, but there is no standard method to indicate the check-in has occurred. Instead, each individual controller develops his or her own personal method to keep track of this status. In this particular loss of communication case, the controller involved would type a symbol in a comment area to mark any aircraft that she had already handed off to the next sector. The controller that was relieved reported that he usually relied on his memory or checked a box to indicate which aircraft he was communicating with.

That a carefully designed and coordinated process such as air traffic control can suffer such problems with coordinating multiple controller process models (and procedures) attests to the difficulty of this design problem and the necessity for careful design and analysis.

9.4.8 Providing Information and Feedback

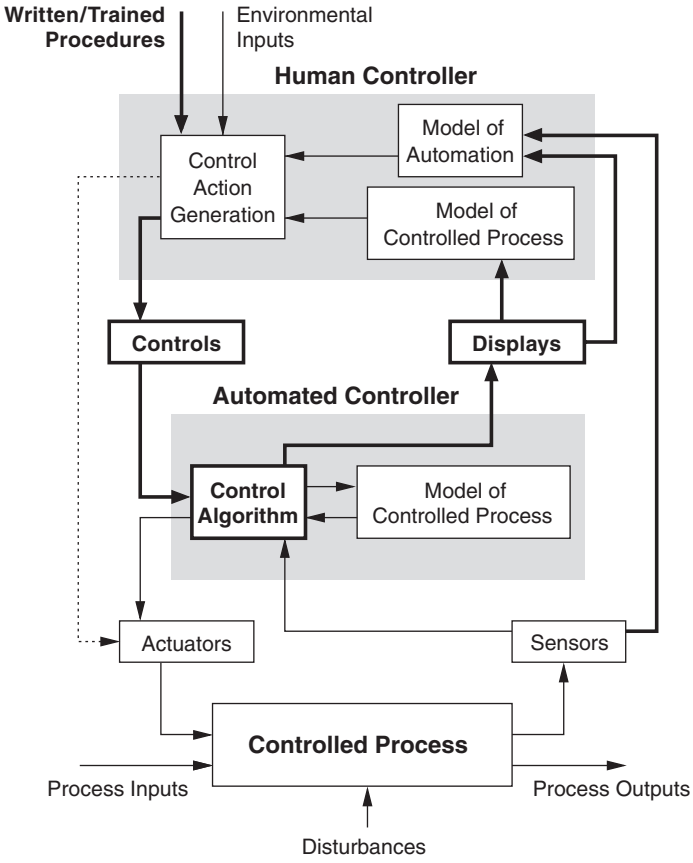
Designing feedback in general was covered in section 9.3.2. This section covers feedback design principles specific to human controllers. Important problems in designing feedback include what information should be provided, how to make the feedback process more robust, and how the information should be presented to human controllers.

Types of Feedback

Hazard analysis using STPA will provide information about the types of feedback needed and when. Some additional guidance can be provided to the designer, once again, using general safety design principles.

Two basic types of feedback are needed:

1. *The state of the controlled process:* This information is used to (1) update the controllers' process models and (2) to detect faults and failures in the other parts of the control loop, system, and environment.



2. *The effect of the controllers' actions:* This feedback is used to detect human errors. As discussed in the section on design for error tolerance, the key to making errors observable—and therefore remediable—is to provide feedback about them. This feedback may be in the form of information about the effects of controller actions, or it may simply be information about the action itself on the chance that it was inadvertent.

Updating Process Models

Updating process models requires feedback about the current state of the system and any changes that occur. In a system where rapid response by operators is necessary, timing requirements must be placed on the feedback information that the controller uses to make decisions. In addition, when task performance requires or implies need for the controller to assess timeliness of information, the feedback display should include time and date information associated with data.

When a human controller is supervising or monitoring automation, the automation should provide an indication to the controller and to bystanders that it is functioning. The addition of a light to the power interlock example in chapter 8 is a simple example of this type of feedback. For robot systems, bystanders should be signaled when the machine is powered up or warning provided when a hazardous zone is entered. An assumption should not be made that humans will not have to enter the robot's area. In one fully automated plant, an assumption was made that the robots would be so reliable that the human controllers would not have to enter the plant often and, therefore, the entire plant could be powered down when entry was required. The designers did not provide the usual safety features such as elevated walkways for the humans and alerts, such as aural warnings, when a robot was moving or about the move. After plant startup, the robots turned out to be so unreliable that the controllers had to enter the plant and bail them out several times during a shift. Because powering down the entire plant had such a negative impact on productivity, the humans got into the habit of entering the automated area of the plant without powering everything down. The inevitable occurred and someone was killed [72].

The automation should provide information about its internal state (such as the state of sensors and actuators), its control actions, its assumptions about the state of the system, and any anomalies that might have occurred. Processing requiring several seconds should provide a status indicator so human controllers can distinguish automated system processing from failure. In one nuclear power plant, the analog component that provided alarm annunciation to the operators was replaced with a digital component performing the same function. An argument was made that a safety analysis was not required because the replacement was "like for like." Nobody considered, however, that while the functional behavior might be the same, the failure behavior could be different. When the previous analog alarm annunciator failed, the screens went blank and the failure was immediately obvious to the human operators. When the new digital system failed, however, the screens froze, which was not immediately apparent to the operators, delaying critical feedback that the alarm system was not operating.

While the detection of nonevents is relatively simple for automated controllers—for instance, watchdog timers can be used—such detection is very difficult for humans. The *absence* of a signal, reading, or key piece of information is not usually immediately obvious to humans and they may not be able to recognize that a missing signal can indicate a change in the process state. In the Turkish Airlines flight TK 1951 accident at Amsterdam's Schiphol Airport in 2009, for example, the pilots did not notice the absence of a critical mode shift [52]. The design must ensure that lack of important signals will be registered and noticed by humans.

While safety interlocks are being overridden for test or maintenance, their status should be displayed to the operators and testers. Before allowing resumption of

normal operations, the design should require confirmation that the interlocks have been restored. In one launch control system being designed by NASA, the operator could turn off alarms temporarily. There was no indication on the display, however, that the alarms had been disabled. If a shift change occurred and another operator took over the position, the new operator would have no way of knowing that alarms were not being annunciated.

If the information an operator needs to efficiently and safety control the process is not readily available, controllers will use experimentation to test their hypotheses about the state of the controlled system. If this kind of testing can be hazardous, then a safe way for operators to test their hypotheses should be provided rather than simply forbidding it. Such facilities will have additional benefits in handling emergencies.

The problem of feedback in emergencies is complicated by the fact that disturbances may lead to failure of sensors. The information available to the controllers (or to an automated system) becomes increasingly unreliable as the disturbance progresses. Alternative means should be provided to check safety-critical information as well as ways for human controllers to get additional information the designer did not foresee would be needed in a particular situation.

Decision aids need to be designed carefully. With the goal of providing assistance to the human controller, automated systems may provide feedforward (as well as feedback) information. Predictor displays show the operator one or more future states of the process parameters, as well as their present state or value, through a fast-time simulation, a mathematical model, or other analytic method that projects forward the effects of a particular control action or the progression of a disturbance if nothing is done about it.

Incorrect feedforward information can lead to process upsets and accidents. Humans can become dependent on automated assistance and stop checking whether the advice is reasonable if few errors occur. At the same time, if the process (control algorithm) truly can be accurately predetermined along with all future states of the system, then it should be automated. Humans are usually kept in systems when automation is introduced because they can vary their process models and control algorithms when conditions change or errors are detected in the original models and algorithms. Automated assistance such as predictor displays may lead to overconfidence and complacency and therefore overreliance by the operator. Humans may stop performing their own mental predictions and checks if few discrepancies are found over time. The operator then will begin to rely on the decision aid.

If decision aids are used, they need to be designed to reduce overdependence and to support operator skills and motivation rather than to take over functions in the name of support. Decision aids should provide assistance only when requested

and their use should not become routine. People need to practice making decisions if we expect them to do so in emergencies or to detect erroneous decisions by automation.

Detecting Faults and Failures

A second use of feedback is to detect faults and failures in the controlled system, including the physical process and any computer controllers and displays. If the operator is expected to monitor a computer or automated decision making, then the computer must make decisions in a manner and at a rate that operators can follow. Otherwise they will not be able to detect faults and failures reliably in the system being supervised. In addition, the loss of confidence in the automation may lead the supervisor to disconnect it, perhaps under conditions where that could be hazardous, such as during critical points in the automatic landing of an airplane. When human supervisors can observe on the displays that proper corrections are being made by the automated system, they are less likely to intervene inappropriately, even in the presence of disturbances that cause large control actions.

For operators to anticipate or detect hazardous states, they need to be continuously updated about the process state so that the system progress and dynamic state can be monitored. Because of the poor ability of humans to perform monitoring over extended periods of time, they will need to be involved in the task in some way, as discussed earlier. If possible, the system should be designed to fail obviously or to make graceful degradation obvious to the supervisor.

The status of safety-critical components or state variables should be highlighted and presented unambiguously and completely to the controller. If an unsafe condition is detected by an automated system being supervised by a human controller, then the human controller should be told what anomaly was detected, what action was taken, and the current system configuration. Overrides of potentially hazardous failures or any clearing of the status data should not be permitted until all of the data has been displayed and probably not until the operator has acknowledged seeing it. A system may have a series of faults that can be overridden safely if they occur singly, but multiple faults could result in a hazard. In this case, the supervisor should be made aware of all safety-critical faults prior to issuing an override command or resetting a status display.

Alarms are used to alert controllers to events or conditions in the process that they might not otherwise notice. They are particularly important for low-probability events. The overuse of alarms, however, can lead to management by exception, overload and the incredulity response.

Designing a system that encourages or forces an operator to adopt a management-by-exception strategy, where the operator waits for alarm signals before taking

action, can be dangerous. This strategy does not allow operators to prevent disturbances by looking for early warning signals and trends in the process state.

The use of computers, which can check a large number of system variables in a short amount of time, has made it easy to add alarms and to install large numbers of them. In such plants, it is common for alarms to occur frequently, often five to seven times an hour [196]. Having to acknowledge a large number of alarms may leave operators with little time to do anything else, particularly in an emergency [196]. A shift supervisor at the Three Mile Island (TMI) hearings testified that the control room never had less than 52 alarms lit [98]. During the TMI incident, more than a hundred alarm lights were lit on the control board, each signaling a different malfunction, but providing little information about sequencing or timing. So many alarms occurred at TMI that the computer printouts were running hours behind the events and, at one point jammed, losing valuable information. Brooks claims that operators commonly suppress alarms in order to destroy historical information when they need real-time alarm information for current decisions [26]. Too many alarms can cause confusion and a lack of confidence and can elicit exactly the wrong response, interfering with the operator's ability to rectify the problems causing the alarms.

Another phenomenon associated with alarms is the incredulity response, which leads to not believing and ignoring alarms after many false alarms have occurred. The problem is that in order to issue alarms early enough to avoid drastic countermeasures, the alarm limits must be set close to the desired operating point. This goal is difficult to achieve for some dynamic processes that have fairly wide operating ranges, leading to the problem of spurious alarms. Statistical and measurement errors may add to the problem.

A great deal has been written about alarm management, particularly in the nuclear power arena, and sophisticated disturbance and alarm analysis systems have been developed. Those designing alarm systems should be familiar with current knowledge about such systems. The following are just a few simple guidelines:

- *Keep spurious alarms to a minimum:* This guideline will reduce overload and the incredulity response.
- *Provide checks to distinguish correct from faulty instruments:* When response time is not critical, most operators will attempt to check the validity of the alarm [209]. Providing information in a form where this validity check can be made quickly and accurately, and not become a source of distraction, increases the probability of the operator acting properly.
- *Provide checks on alarm system itself:* The operator has to know whether the problem is in the alarm or in the system. Analog devices can have simple checks such as "press to test" for smoke detectors or buttons to test the bulbs in a

lighted gauge. Computer-displayed alarms are more difficult to check; checking usually requires some additional hardware or redundant information that does not come through the computer. One complication comes in the form of alarm analysis systems that check alarms and display a prime cause along with associated effects. Operators may not be able to perform validity checks on the complex logic necessarily involved in these systems, leading to overreliance [209]. Weiner and Curry also worry that the priorities might not always be appropriate in automated alarm analysis and that operators may not recognize this fact.

- *Distinguish between routine and safety-critical alarms:* The form of the alarm, such as auditory cues or message highlighting, should indicate degree or urgency. Alarms should be categorized as to which are the highest priority.
- *Provide temporal information about events and state changes:* Proper decision making often requires knowledge about the timing and sequencing of events. Because of system complexity and built-in time delays due to sampling intervals, however, information about conditions or events is not always timely or even presented in the sequence in which the events actually occurred. Complex systems are often designed to sample monitored variables at different frequencies: some variables may be sampled every few seconds while, for others, the intervals may be measured in minutes. Changes that are negated within the sampling period may not be recorded at all. Events may become separated from their circumstances, both in sequence and time [26].
- *Require corrective action when necessary:* When faced with a lot of undigested and sometimes conflicting information, humans will first try to figure out what is going wrong. They may become so involved in attempts to save the system that they wait too long to abandon the recovery efforts. Alternatively, they may ignore alarms they do not understand or they think are not safety critical. The system design may need to ensure that the operator cannot clear a safety-critical alert without taking corrective action or without performing subsequent actions required to complete an interrupted operation. The Therac-25, a linear accelerator that massively overdosed multiple patients, allowed operators to proceed with treatment five times after an error message appeared simply by pressing one key [115]. No distinction was made between errors that could be safety-critical and those that were not.
- *Indicate which condition is responsible for the alarm:* System designs with more than one mode or where more than one condition can trigger the alarm for a mode, must clearly indicate which condition is responsible for the alarm. In the Therac-25, one message meant that the dosage given was either too low or too high, without providing information to the operator

about which of these errors had occurred. In general, determining the cause of an alarm may be difficult. In complex, tightly coupled plants, the point where the alarm is first triggered may be far away from where the fault actually occurred.

- *Minimize the use of alarms when they may lead to management by exception:* After studying thousands of near accidents reported voluntarily by aircraft crews and ground support personnel, one U.S. government report recommended that the altitude alert signal (an aural sound) be disabled for all but a few long-distance flights [141]. Investigators found that this signal had caused decreased altitude awareness in the flight crew, resulting in more frequent overshoots—instead of leveling off at 10,000 feet, for example, the aircraft continues to climb or descend until the alarm sounds. A study of such overshoots noted that they rarely occur in bad weather, when the crew is most attentive.

Robustness of the Feedback Process

Because feedback is so important to safety, robustness must be designed into feedback channels. The problem of feedback in emergencies is complicated by the fact that disturbances may lead to failure of sensors. The information available to the controllers (or to an automated system) becomes increasingly unreliable as the disturbance progresses.

One way to prepare for failures is to provide alternative sources of information and alternative means to check safety-critical information. It is also useful for the operators to get additional information the designers did not foresee would be needed in a particular situation. The emergency may have occurred because the designers made incorrect assumptions about the operation of the controlled system, the environment in which it would operate, or the information needs of the controller.

If automated controllers provide the only information about the controlled system state, the human controller supervising the automation can provide little oversight. The human supervisor must have access to independent sources of information to detect faults and failures, except in the case of a few failure modes such as total inactivity. Several incidents involving the command and control warning system at NORAD headquarters in Cheyenne Mountain involved situations where the computer had bad information and thought the United States was under nuclear attack. Human supervisors were able to ascertain that the computer was incorrect through direct contact with the warning sensors (satellites and radars). This direct contact showed the sensors were operating and had received no evidence of incoming missiles [180]. The error detection would not have been possible if the humans

could only get information about the sensors from the computer, which had the wrong information. Many of these direct sensor inputs are being removed in the mistaken belief that only computer displays are required.

The main point is that human supervisors of automation cannot monitor its performance if the information used in monitoring is not independent from the thing being monitored. There needs to be provision made for failure of computer displays or incorrect process models in the software by providing alternate sources of information. Of course, any instrumentation to deal with a malfunction must not be disabled by the malfunction, that is, common-cause failures must be eliminated or controlled. As an example of the latter, an engine and pylon came off the wing of a DC-10, severing the cables that controlled the leading edge flaps and also four hydraulic lines. These failures disabled several warning signals, including a flap mismatch signal and a stall warning light [155]. If the crew had known the slats were retracted and had been warned of a potential stall, they might have been able to save the plane.

Displaying Feedback to Human Controllers

Computer displays are now ubiquitous in providing feedback information to human controllers, as are complaints about their design.

Many computer displays are criticized for providing too much data (data overload) where the human controller has to sort through large amounts of data to find the pieces needed. Then the information located in different locations may need to be integrated. Bainbridge suggests that operators should not have to page between displays to obtain information about abnormal states in the parts of the process other than the one they are currently thinking about; neither should they have to page between displays that provide information needed for a single decision process.

These design problems are difficult to eliminate, but performing a task analysis coupled with a hazard analysis can assist in better design as will making all the information needed for a single decision process visible at the same time, placing frequently used displays centrally, and grouping displays of information using the information obtained in the task analysis. It may also be helpful to provide alternative ways to display information or easy ways to request what is needed.

Much has been written about how to design computer displays, although a surprisingly large number of displays still seem to be poorly designed. The difficulty of such design is increased by the problem that, once again, conflicts can exist. For example, intuition seems to support providing information to users in a form that can be quickly and easily interpreted. This assumption is true if rapid reactions are required. Some psychological research, however, suggests that cognitive processing

for meaning leads to better information retention: A display that requires little thought and work on the part of the operator may not support acquisition of the knowledge and thinking skills needed in abnormal conditions [168].

Once again, the designer needs to understand the tasks the user of the display is performing. To increase safety, the displays should reflect what is known about how the information is used and what kinds of displays are likely to cause human error. Even slight changes in the way information is presented can have dramatic effects on performance.

This rest of this section concentrates only on a few design guidelines that are especially important for safety. The reader is referred to the standard literature on display design for more information.

Safety-related information should be distinguished from non-safety-related information and highlighted. In addition, when safety interlocks are being overridden, their status should be displayed. Similarly, if safety-related alarms are temporarily inhibited, which may be reasonable to allow so that the operator can deal with the problem without being continually interrupted by additional alarms, the inhibit status should be shown on the display. Make warning displays brief and simple.

A common mistake is to make all the information displays digital simply because the computer is a digital device. Analog displays have tremendous advantages for processing by humans. For example, humans are excellent at pattern recognition, so providing scannable displays that allow operators to process feedback and diagnose problems using pattern recognition will enhance human performance. A great deal of information can be absorbed relatively easily when it is presented in the form of patterns.

Avoid displaying absolute values unless the human requires the absolute values. It is hard to notice changes such as events and trends when digital values are going up and down. A related guideline is to provide references for judgment. Often, for example, the user of the display does not need the absolute value but only the fact that it is over or under a limit. Showing the value on an analog dial with references to show the limits will minimize the required amount of extra and error-prone processing by the user. The overall goal is to minimize the need for extra mental processing to get the information the users of the display need for decision making or for updating their process models.

Another typical problem occurs when computer displays must be requested and accessed sequentially by the user, which makes greater memory demands upon the operator, negatively affecting difficult decision-making tasks [14]. With conventional instrumentation, all process information is constantly available to the operator: an overall view of the process state can be obtained by a glance at the console. Detailed readings may be needed only if some deviation from normal conditions is detected.

The alternative, a process overview display on a computer console, is more time consuming to process: To obtain additional information about a limited part of the process, the operator has to select consciously among displays.

In a study of computer displays in the process industry, Swaanenburg and colleagues found that most operators considered a computer display more difficult to work with than conventional parallel interfaces, especially with respect to getting an overview of the process state. In addition, operators felt the computer overview displays were of limited use in keeping them updated on task changes; instead, operators tended to rely to a large extent on group displays for their supervisory tasks. The researchers conclude that a group display, showing different process variables in reasonable detail (such as measured value, setpoint, and valve position), clearly provided the type of data operators preferred. Keeping track of the progress of a disturbance is very difficult with sequentially presented information [196]. One general lesson to be learned here is that the operators of the system need to be involved in display design decisions: The designers should not just do what is easiest to implement or satisfies their aesthetic senses.

Whenever possible, software designers should try to copy the standard displays with which operators have become familiar, and which were often developed for good psychological reasons, instead of trying to be creative or unique. For example, icons with a standard interpretation should be used. Researchers have found that icons often pleased system designers but irritated users [92]. Air traffic controllers, for example, found the arrow icons for directions on a new display useless and preferred numbers. Once again, including experienced operators in the design process and understanding why the current analog displays have developed as they have will help to avoid these basic types of design errors.

An excellent way to enhance human interpretation and processing is to design the control panel to mimic the physical layout of the plant or system. For example, graphical displays allow the status of valves to be shown within the context of piping diagrams and even the flow of materials. Plots of variables can be shown, highlighting important relationships.

The graphical capabilities of computer displays provides exciting potential for improving on traditional instrumentation, but the designs need to be based on psychological principles and not just on what appeals to the designer, who may never have operated a complex process. As Lees has suggested, the starting point should be consideration of the operator's tasks and problems; the display should evolve as a solution to these [110].

Operator inputs to the design process as well as extensive simulation and testing will assist in designing usable computer displays. Remember that the overall goal is to reduce the mental workload of the human in updating their process models and to reduce human error in interpreting feedback.

9.5 Summary

A process for safety-guided design using STPA and some basic principles for safe design have been described in this chapter. The topic is an important one and more still needs to be learned, particularly with respect to safe system design for human controllers. Including skilled and experienced operators in the design process from the beginning will help as will performing sophisticated human task analyses rather than relying primarily on operators interacting with computer simulations.

The next chapter describes how to integrate the disparate information and techniques provided so far in part III into a system-engineering process that integrates safety into the design process from the beginning, as suggested in chapter 6.