INTERFACE DESIGN

It is very difficult to conceive of an engineering system that does not require interaction with people somewhere along the way. In some cases, people manually control an engineered system, as is the case of a driver of an automobile. In others, people supervise the operation of an engineered system that is usually being controlled by automation, as is the case of human operators of some nuclear power plants. In still other cases, people are required to maintain or repair an engineered system that normally runs autonomously without human intervention, as is the case of technicians trouble shooting electronic hardware. Finally, in some cases people share responsibility with an engineered system in a shared mode of control, as with pilots interacting with the autopilot of a commercial aircraft. In all of these cases, and many others as well, the interaction between people and technology is an unavoidable fact of life.

As computer technology is being introduced into more and more sectors of contemporary society, this interaction between people and technology is mediated by a computer interface. For example, intelligent vehicle highway systems, such as electronic maps, are being introduced into automobiles to help drivers find their way and avoid high-traffic areas. Computer-based displays are also being introduced into advanced control rooms for nuclear power plants to help operators perform their job more effectively and efficiently, thereby replacing analog, hard-wired instrumentation (e.g., analog gauges). Also, increasingly sophisticated computer software and hardware tools have been developed to facilitate the trouble-shoot-

J. Webster (ed.), *Wiley Encyclopedia of Electrical and Electronics Engineering*. Copyright © 1999 John Wiley & Sons, Inc.

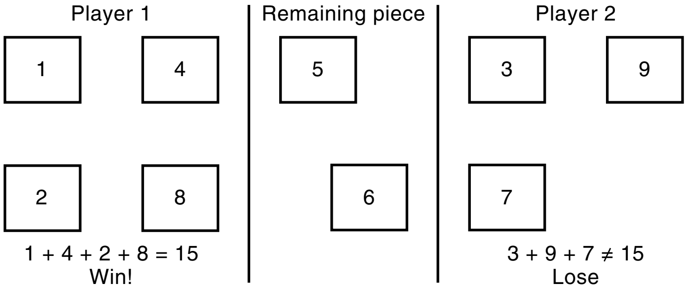


Figure 1. One “interface” for playing the two-person game. Because information is presented in the form of abstract symbols, people have to play the game by relying on reasoning.

ing performance of electronics technicians. Finally, increasingly sophisticated computer-based flight management systems are being introduced into the “glass cockpits” of modern commercial aircraft to help pilots during the various phases of flight. As a result, the human–computer interface is becoming a dominant mode of interaction between people and technology. Engineers are increasingly being required to design a human–computer interface so that people can interact with the engineered system.

The importance of this aspect of design cannot be overemphasized. Depending on the type of decisions that are made in creating the human–computer interface, the resulting system can either be successful or completely unusable. This is true even if the technological core of the engineered system remains unaltered. In other words, the adequacy of the human–computer interface can either make or break the system.

This crucial observation can be illustrated by a very simple example (1). Consider the following two-person game. There are nine cardboard pieces available to each player. Each piece has drawn on it one of the integers from 1 to 9. The pieces are face up so that both players can see all of the numbers. The players take turns drawing one piece from the remaining set. The first player to hold three pieces in his hand whose integers sum up to exactly 15 wins the game. If all nine pieces are drawn and neither player has a winning combination, then the game is tied. One interface for playing this game is shown in Fig. 1. This is an obvious way to represent the problem, given the rules just described. The reader is encouraged to envision playing this game in order to get an appreciation for its demands.

An alternative interface for playing the very same game is shown in Fig. 2. There is a 3 × 3 matrix of blank squares. Players alternate marking a square, one player with an X and the other with an O. The first player to get a vertical, horizontal, or diagonal sequence of three symbols (Xs or Os) wins the game. If all of the squares are taken and no player has such a sequence, then the game is tied. Readers will recognize this as the well-known game of tic-tac-toe.

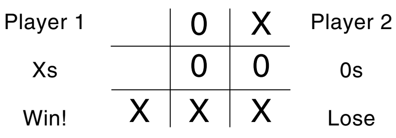


Figure 2. A second “interface” for playing the same game. Because information is presented in the form of concrete patterns, people can play the game by relying on perceptual skills.

It should be obvious to the reader that playing the game with the first interface is considerably more difficult than playing the game with the second interface. In fact, the difference is so strong that the reader may think that a different game is being played with the second interface. However, an examination of the formal properties of these two games unambiguously reveals that the logics of the two games are actually isomorphic to each other (1). At their core, the two versions of the game are exactly the same. Nevertheless, we correctly perceive that one version is much easier than the other. The reason for this is that the way in which the problem is represented (i.e., the interface) has such a big impact on human performance. The interface in Fig. 1 is based on abstract symbols (the integers from 1 to 9), and so it requires people to engage in analytical problem solving (e.g., mental arithmetic) to perform the task. It is well known that problem solving activities are comparatively slow, effortful, and error-prone (2,3). This accounts for why this version of the game seems so much more difficult to play.

In contrast, the interface in Fig. 2 is based on concrete patterns (Xs and Os). Because information is presented in this more concrete form, people are able to rely on their powerful pattern recognition capabilities that have been honed through experience with everyday tasks. It is well known that pattern recognition capabilities are comparatively fast, efficient, and less prone to error, given the proper level of experience (3). As a result, this version of the game seems much easier to play.

The point of this simple example can be generalized to the design of human–computer interfaces. The ease and reliability with which people interact with a particular engineered system is strongly affected by the type of interface that is provided. Given the very same technical system, human performance can either be made to be very difficult and cumbersome if the human–computer interface is designed one way, or it can be fluid and efficient if the interface is designed in a different way. As a result, the interface designer is actually responsible for shaping human behavior. If there is a good fit between human capabilities and limitations and the demands being placed by the interface, then the result will be effective and reliable performance. On the other hand, if there is a poor fit between human capabilities and limitations and the demands being placed by the interface, then the result will be human error. Therefore, human errors can frequently be traced back to inadequate interface designs (2).

Now that the importance of human–computer interface design has been illustrated, the design remedy seems relatively obvious—design the interface to take advantage of people’s skills and the result should be enhanced human performance. Unfortunately, most engineers are not very well prepared to deal with the design challenges imposed by human–computer interaction. Traditionally, engineering education has focused almost exclusively on the technical component of the system, to the detriment of the human, social, and environmental considerations. As a result, it is not surprising to find that there have been, and continue to be, many computer systems that are poorly designed from the perspective of the people who have to use them (4).

DESIGNING SMART INTERFACES

Fortunately, a fair amount is known about how to design effective human–computer interfaces that lead to efficient and reliable system performance (3,5,6). Some of this knowledge, which comes from the discipline known as *cognitive engineering* (7,8), will be described next.

Rote and Smart Devices

One useful way to classify interfaces is according to the distinction between rote and smart devices first put forward by Runeson (9) in an exceptionally insightful and entertaining paper. Traditionally, human–computer interfaces have been designed to be rote devices, but as we shall see, there are very strong reasons for moving toward a smart approach to interface design (10).

Rote Devices. According to Runeson (9), a *rote* device performs a rather simple task, namely measuring a basic context-free (i.e., primitive) property of the environment. An example is a ruler, which measures a fundamental dimension, length. The advantage of rote devices is that they can be used to derive a variety of different properties. For example, a ruler can be used to measure various lengths, from which one can compute area and volume. The disadvantage of rote devices is this need for computation—that is, to derive more complex properties, the person must know the rules (or algorithm) for combining the elemental measurements. For instance, to derive the area of a triangle with a ruler, the person must know the appropriate formula. In general, the person must also engage in calculations to derive the higher order properties of interest. These calculations, in turn, require some time and effort to carry out.

Runeson (9) originally argued that the metaphor of rote devices is not a very appropriate one for human perception. For a goal-directed organism, the most important properties of the environment are likely to be the higher-order functional properties that are relevant to its immediate goals, not the context-free primitive properties measured by rote devices. For example, within the domain of perception, it is usually more important for a person to know whether a certain object affords sitting than to know the exact dimensions of the object. The suggestion is that if human perception were to function like a rote device, it would be very inefficient, if not intractable. Returning to the sitting example, it would be very difficult for observers using primitives based on rote devices (e.g., length) to determine whether a given object was indeed sit-onable or not. In addition to an extensive knowledge of geometry, anthropometry, and biomechanics, a very large number of calculations would also be necessary, requiring substantial effort and time, for what appears to be a very basic task.

Smart Devices. The alternative is what Runeson (9) refers to as *smart* devices. These are specialized on a particular type of task in a particular type of situation. Their disadvantage is that, unlike rote devices, they cannot be used for a large, arbitrary set of purposes. Their great advantage, however, is that they "capitalize on the peculiarities of the situation and the task" (9, p. 174). In other words, smart devices are special-purpose devices that are designed to exploit goal-relevant constraints pertinent in a given setting. As a result, smart devices can *directly register* higher-order properties in the environment, rather than having to calculate them from sensed primitive variables using computational rules and effort.

The ingenious example that Runeson gives of a smart device is a polar planimeter, a mechanical device consisting of two rigid bars connected by an articulated joint. The end of one bar is used as a fixed anchor, whereas the end of the other bar is used to trace along the perimeter of a flat surface. There is also an analog meter, which displays the value of the measurement made by the polar planimeter. Remarkably, a polar planimeter directly measures the area of *any* two-dimensional figure, regardless of shape. This is accomplished by anchoring one end of the planimeter on the flat surface that the figure is laying on, and then using the measuring end to trace continuously around the perimeter of the figure, until the measuring end reaches the point along the perimeter at which the measurement was initiated. At this point, the analog meter will indicate the area of the figure.

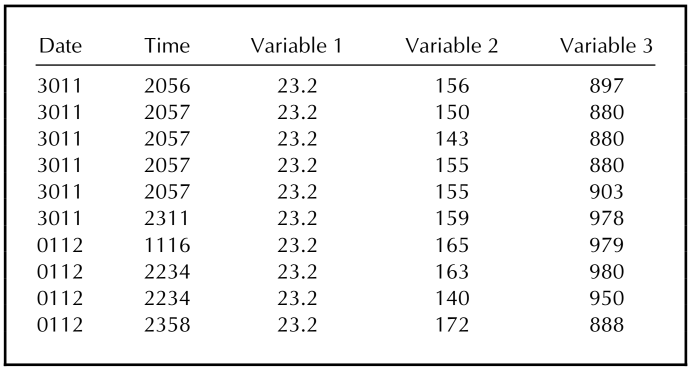
A polar planimeter has several important properties. First, it does not use length to arrive at its measurement of area. In fact, it cannot be used to measure length at all, or any other property for that matter. It is a special-purpose instrument; it can only measure area. Second, there is no meaningful sense in which one can say that the polar planimeter is calculating anything. There are no primitive inputs that need to be integrated in any way, because there are no primitive inputs to begin with. There are no intermediate calculations either, because if one stops the process of measuring the area of a figure prematurely, the readout on the polar planimeter has no meaningful interpretation (e.g., stopping midway around the perimeter does not usually lead to a reading that corresponds to half of the area). Third, no rules or knowledge are represented in the device. The polar planimeter is simply a mechanical measuring instrument, and does not possess any internal representation, or model, which it uses for reasoning. It does not reason; it measures.

This set of properties derives from the fact that the polar planimeter is a physical embodiment of the constraints that are relevant to the task at hand. Although it can be described in very analytical and computational terms (i.e., surface integrals), such a description has no causal, explanatory value in describing its operation. Rather, such a rationalized description can only explain the constraints to which the design of the device had to conform to be a reliable measuring instrument. But once these constraints are embedded in the physical device, the analytical account merely represents the design history, not the real-time operation, of the device. In other words, the polar planimeter does not have a symbolic model of the goal-relevant constraints in the environment, although it could be said to be a mechanical adaptation to those constraints, and thus can be described in symbolic terms.

Although it may not be apparent from the description so far, the distinction between rote and smart devices has a great deal of relevance to human–computer interface design.

Rote and Smart Interfaces

Rote Interfaces. Traditionally, human–computer interfaces have been designed as rote devices. The philosophy has been referred to as the *single-sensor single-indicator* (SSSI) design approach (11). Basically, it consists of displaying all of the



Date	Time	Variable 1	Variable 2	Variable 3
3011	2056	23.2	156	897
3011	2057	23.2	150	880
3011	2057	23.2	143	880
3011	2057	23.2	155	880
3011	2057	23.2	155	903
3011	2311	23.2	159	978
0112	1116	23.2	165	979
0112	2234	23.2	163	980
0112	2234	23.2	140	950
0112	2358	23.2	172	888

Figure 3. An example of a rote interface. Only raw sensor data are shown, so people have to derive higher-order information.

elemental data that are directly available from sensors. Anything and everything that can be directly measured has a single display element associated with it. A hypothetical example is shown in Fig. 3, which is actually very similar to some human–computer interfaces currently being used by people to interact with complex engineering systems. There are many disadvantages to such an approach (5,11), some of which are identical to those associated with rote instruments.

The most important drawback is related to *controllability*. To deal consistently with the entire range of domain demands (particularly fault situations), people need comprehensive information regarding the system state. But with the SSSI approach, only data that are directly obtained from sensors tend to be displayed. Thus, higher-order state information, which cannot be directly measured, but which is nevertheless needed to cope with many fault situations, may not be made available to users. In fault situations, it is generally not possible to derive the higher-order properties from the elemental data, and so people may not have all of the information that is required to consistently control the system under these circumstances.

The disadvantages of the SSSI approach are not limited to fault scenarios. Even under normal operating conditions, SSSI interfaces put an excessive burden on operators. In these situations, it is, in principle at least, possible to recover the higher order, goal-relevant domain properties from the elemental data represented in the interface. The problem, of course, is that considerable effort and knowledge are required of the operator to carry out this derivation. Not only are the higher order properties not directly displayed, but in addition, the relationships between the various elemental display elements also are not usually represented in the interface (see Fig. 3).

Finally, there is also the issue of information pickup. Just because the information is in the interface does not mean that the operator can find it easily (5). In the SSSI approach, the form in which information is usually presented (e.g., similar looking analog meters or digital numerical displays) is not very compatible with the capabilities of the human perceptual system, thereby hindering the process of information extraction. Each instrument tends to be presented individually, and there is virtually no integration or nesting of display elements. This makes it difficult for people to perceive the state of the system, even if all of the requisite information is in the interface.

In summary, the rote interface approach makes people's jobs more difficult than they really need to be by requiring them to engage in computations, store information in memory, and then retrieve that information at the right time. All of this takes time and effort. Clearly, an alternative approach is required.

Smart Interfaces. The advantages of smart devices suggest that the approach may be a useful one for human–computer interface design. The goal of a smart approach to interface design would be to provide the information needed for controllability in a form that exploits the power of perception. The first step in achieving this goal is to identify all of the information that is relevant to the context of interest to the person who will be using the interface. The second step requires identifying the various relationships between these variables of interest. As we will see, relationships play a critical role in smart interfaces. The third and final step is to embed these variables and relationships into the interface in a form that makes it easy for people to pick up information accurately and efficiently.

The advantages of adopting a smart approach to interface design are very similar to those identified earlier for smart devices. Rather than requiring people to remember and retrieve the relationships between variables of interest, these are directly shown in the interface, thereby reducing the demand on memory. Similarly, rather than requiring people to derive higher-order information by engaging in computations, this information is also directly shown in the interface, thereby reducing the demand for analytical problem solving. As a result, human performance should be more accurate and more efficient as well.

However, just as with the polar planimeter example, these advantages can only be obtained if the designers perform the required systems analysis up front and build the results of this analysis into the interface. That is, the power of smart interfaces is that they offload much of the analysis to the designer (who has more time and better tools to deal with these issues off-line and *a priori*), and thus relieve the burden on the person using the interface (who is busy performing many other tasks in real time). The demands imposed by the task at hand cannot be displaced if effective performance is to be achieved. The only choice is who is better capable to deal with those demands, system designers or people operating the interface. These abstract points can be illustrated by the following concrete example of a smart interface.

Application Example

One activity that people are usually responsible for when interacting with an engineered system is assessing the overall status of the system. Is the system in a normal or abnormal state? For a complex system with many variables, this can be a very challenging task, involving a number of different steps (5). First, the person has to know and remember which variables are the most important ones in determining overall system status. Typically, only a small subset of the hundreds or thousands of available variables will be needed. Second, the person must collect together the status of these relevant variables. This activity requires more knowledge, because the person must know where to look to find the variables of interest. This activity also requires time and effort, because the variables generally will not be found all in one place. For instance, in a human–computer interface, the person may have to search through a hierarchical menu of windows to find the window that contains one of the relevant variables. This procedure would then have to be repeated for each of the relevant variables. Third, the person may also have to integrate together these variables to obtain the higher order information of interest. This activity may be required because sometimes the variable of interest is not something that can be measured directly by a sensor, but rather something that can only be derived from several of the variables that can be directly sensed. This derivation process requires knowledge of the correct integration formula, and mental effort to compute the derived variable from the lower-level sensed data. Fourth, the person will also have to know and remember the normal range for each of the variables that are relevant to determining overall system status. After all, the value of a particular variable only takes on some meaning when it is compared to its nominal or limit values.

All of the activities just listed must be performed if the task of overall system assessment is to be done accurately and reliably. However, as the discussion of rote and smart devices indicates, there are different ways in which these demands can be allocated between the designer and person operating the system. With a rote interface, only raw sensor data are presented. As a result, a great burden is put on the person to perform the variable identification, collection, integration, and normalization activities listed above. Rather than getting some help from the interface, the person must perform these activities unaided, which as already mentioned, requires a fair amount of knowledge, puts a substantial load on memory, and demands a fair amount of time and effort. Given the properties of smart interfaces, it should be possible to do much better by off-loading at least some of this burden to the designer. Rather than forcing the user to deal with all of these demands, it should be possible for the designer to build these constraints into a smart interface.

Figure 4 shows an example of a smart interface developed by Coekin (12) that can help people perform the task of assessing the overall status of a complex system. This display consists of eight spokes arranged in a symmetrical fashion. Each spoke displays the status of one of the variables that are relevant to determining the overall status of the system. Note that these individual variables can be either raw sensor values or higher order information that must be computed by integrating together a number of variables that are sensed individually. The current states of individual variables are connected together by a line joining adjacent spokes. Another important feature of this display is that each of the variables displayed has been normalized according to its nominal value. If each variable is at its nominal value, then it will be displayed at a fixed distance from the center of the polygon. If all eight variables are in the normal range, then a symmetrical figure will be obtained.

As a result, the task of determining whether the system is in a normal state or not is dead-simple. If the figure is symmetrical and in its normal diameter, as it is in Fig. 4(a), then the system is in a normal state. On the other hand, if this symmetry is broken, as it is in Fig. 4(b), then the system is in an abnormal state. Moreover, the way in which the octagon deforms may give some information about the nature of the abnormality. For example, if the left side of the polygon caves in toward the center, this may signify one type of failure, whereas a deformation of the right side towards the center may indicate a very different type of failure. This can be accomplished by grouping variables that are functionally or physically related in proximate spokes.

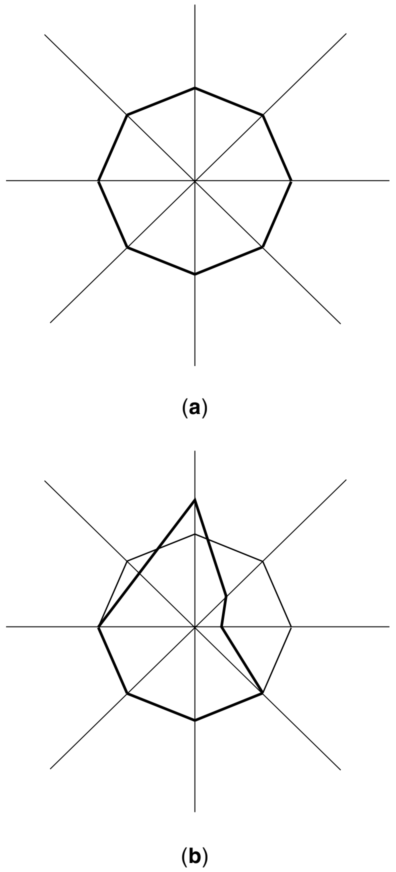


Figure 4. An example of a smart interface. Adapted from Coekin (12). (a) Normal state specified by a symmetrical octagon. (b) Abnormal state specified by a deformed, assymetrical polygon. Concrete patterns are shown, so people can directly perceive higher-order information.

The demands that this smart interface places on people are trivial compared to the demands imposed by a rote interface. As mentioned earlier, this is because the interface designer has taken on much of the responsibility of dealing with the relevant constraints. The reason why this smart interface required much less knowledge, memory, effort, and time to use is because the designer has identified the relevant variables, brought them together into one place, performed any necessary integration, and normalized the variables with respect to their nominal values. Because the designer has done this work, much less work is left for the person who is controlling the system in real time.

The smart interface concept illustrated in Fig. 4 has been adopted to design system status displays for a number of different application domains, including aviation (to monitor the status of engineering systems), medicine (to monitor the life signs of a patient), and nuclear power plants (to monitor the state of the plant). It can surely find use in many other situations as well. Nevertheless, it is important to point out that this octagon interface is merely one example of a smart interface. The important point to take away is not so much the details of this particular exemplar, but rather the process by which it was designed. If designers can identify the goal-relevant constraints and build them into the interface in a form that makes it easy for people to pick up information, then very different interfaces can be developed for other applications but with the same advantages as this smart interface.

ADVANCED APPLICATIONS

There are a number of promising directions for the advanced application of smart interfaces for complex engineered systems. Two of these are creating smart interfaces from visualizations of engineering models described in textbooks, and the application of smart interface design principles to make automation more visible.

Visualization of Engineering Models

One powerful and virtually untapped source of ideas for smart interfaces is the set of visualizations developed by engineers over the years to teach basic principles and models in textbooks. A prototypical example is the temperature–entropy (T–s) diagram that has been used in thermodynamics textbooks for years to represent the saturation properties of water. More specifically, the T–s diagram has been used as a frame of reference for representing the various phases of different thermodynamic heat engine cycles (e.g., the Rankine cycle). This is a fortunate choice from the viewpoint of cognitive engineering because different areas of the T–s diagram represent important thermodynamic distinctions in a form that is very easy for people to discriminate (e.g., the area under the saturation curve for water indicates a two-phase state). Similarly, vertical and horizontal lines in this diagram represent meaningful thermodynamic constraints on the various phases of the heat engine cycle in a form that is also easy to perceive (e.g., isentropic and isothermal transformations, respectively).

Beltracchi (13) has taken advantage of these properties by creating a smart interface that represents the state of a water-based power plant in the form of a Rankine cycle overlaid on the T–s diagram. This interface brings to life the Rankine cycle representation in the T–s diagram found in textbooks by animating it with live sensor data and by coding information with perceptually salient properties such as color. An initial evaluation of this smart interface suggests that it has important advantages over more traditional, rote interfaces that have been (and continue to be) used in nuclear power plant control rooms (14). This suggests that it may be possible to develop other smart interfaces from the myriad of visualizations that can be found in engineering textbooks. Some obvious examples include: pressure–volume diagrams, phase diagrams, and nomograms.

Making Automation More Visible

So far, this article has concentrated on techniques for building constraints that govern the behavior of engineered systems into easily perceivable forms in a human–computer interface. However, the very same logic could be applied, not just to make process constraints visible, but also to make automation constraints visible as well. This seems to be a very fertile area of application, because a number of studies have indicated that one of the problems with contemporary automation (e.g., on the flight decks of "glass cockpits," or in the control rooms of petrochemical refineries) is that the behavior of the automated systems is not very clearly displayed (15).

This creates a number of difficulties for the people who are responsible for monitoring these systems. First, it is difficult for people to monitor the actions of the automation to track how those systems are reconfiguring the process in response to disturbances or changes in demands. If people cannot keep track of the automated systems' actions, then people's understanding of the current configuration of the process may not correspond to the actual configuration. Thus, their subsequent actions may not be appropriate, leading to unintended consequences. Second, it is also difficult for people to monitor the state of the process to anticipate problems before they jeopardize system goals (e.g., by propagating to other parts of the process). Research has repeatedly shown that it is essential for people to be able to effectively anticipate the future state of the process if they are to function as effective controllers. If people are operating in a reactive mode, then they will always be one step behind the course of events, and given the lags in complex engineered systems, they will not be able to respond to problems until after they occur. Third, it is also difficult for people to monitor the state of the automation to quickly detect and diagnose any faults in the automation. In highly automated systems, the primary reason why there are people in the system is to supervise the automation and to detect fault situations in which the automation is not working properly so that they can improvise a solution to the problem. If people do not have rich feedback from the human–computer interface that makes it clear whether the automation is working properly or not, then they will not be able to perform this role effectively and reliably.

If, on the other hand, designers think of human–computer interfaces for automation from the perspective of smart devices, then many of these problems can potentially be overcome. Just as it is possible to build process constraints into an interface in a form that is easy to perceive, it should also be possible to do the same for automation constraints. In fact, one can envision human–computer interfaces that provide integrated representations of process and automation status. These smart interfaces should allow people to independently track the status and configuration of both the process and the automation, thereby providing them with the feedback that they need to effectively understand the interaction between the controller and the process being controlled.

THE FUTURE

Perhaps ironically, as technology evolves in sophistication and availability, there will be an increasing need for effective human–computer interface design. The reason for this is that there will be more and more engineered systems that require an interface with the people who will interact with those systems. The perspective of smart devices described in this article should enable designers to develop human–computer interfaces that provide a good fit between the characteristics of people and the demands imposed by the systems with which they interact. The result should be safer, more productive, and more reliable system performance. Only by designing for people will these goals be achieved. Or in other words, if technology does not work for people, then it does not work (16).

ACKNOWLEDGMENTS

The writing of this article was made possible by research and equipment grants from the Natural Sciences and Engineering Research Council of Canada.

BIBLIOGRAPHY

1. A. Newell and H. A. Simon, *Human Problem Solving,* Englewood Cliffs, NJ: Prentice-Hall, 1972.
2. D. A. Norman, *The Psychology of Everyday Things,* New York: Basic Books, 1988.
3. K. J. Vicente and J. Rasmussen, Ecological interface design: theoretical foundations, *IEEE Trans. Syst. Man Cybern.,* **SMC-22**: 589–606, 1992.
4. N. G. Leveson, *Safeware: System Safety and Computers,* Reading, MA: Addison-Wesley, 1995.
5. D. D. Woods, The cognitive engineering of problem representations. In J. Alty and G. Weir (eds.), *Human-Computer Interaction in Complex Systems,* London: Academic Press, 1991, pp. 169–188.
6. K. B. Bennett and J. M. Flach, Graphical displays: implications for divided attention, focused attention, and problem solving, *Human Factors,* **34**: 513–533, 1992.
7. D. D. Woods and E. M. Roth, Cognitive engineering: Human problem solving with tools, *Human Factors,* **30**: 415–430, 1988.
8. J. Rasmussen, A. M. Pejtersen, and L. P. Goodstein, *Cognitive Systems Engineering,* New York: Wiley, 1994.
9. S. Runeson, On the possibility of "smart" perceptual mechanisms, *Scand. J. Psychol.,* **18**: 172–179, 1977.
10. K. J. Vicente and J. Rasmussen, The ecology of human-machine systems II: mediating "direct perception" in complex work domains. *Ecol. Psychol.,* **2**: 207–250, 1990.
11. L. P. Goodstein, Discriminative display support for process operators. In J. Rasmussen and W. B. Rouse (eds.), *Human Detection and Diagnosis of System Failures,* New York: Plenum, 1981, pp. 433–449.
12. J. A. Coekin, A versatile presentation of parameters for rapid recognition of total state. In *Proceedings of the IEE International Symposium on Man-Machine Systems,* Cambridge: IEE, 1969.
13. L. Beltracchi, A direct manipulation interface for water-based rankine cycle heat engines. *IEEE Trans. Syst. Man Cybern.,* **SMC-17:** 478–487, 1987.
14. K. J. Vicente, N. Moray, J. D. Lee, J. Rasmussen, B. G. Jones, R. Brock, and T. Djemil, Evaluation of a Rankine cycle display for nuclear power plant monitoring and diagnosis. *Human Factors,* **38**: 506–521, 1996.
15. D. A. Norman, The "problem" of automation: inappropriate feedback and interaction, not "over-automation." *Phil. Trans. Roy. Soc. Lond.* , **B 327**: 585–593, 1990.
16. A. M. Lund, Advertising human factors. *Ergonomics Design,* **4** (4): 5–11, 1996.

Kim J. Vicente
University of Toronto

INTERFACES, SEMICONDUCTOR-ELECTROLYTE. See Semiconductor-electrolyte interfaces.

INTERFACES, SEMICONDUCTOR-INSULATOR. See Semiconductor-insulator boundaries.

INTERFACE STATES. See Surface states.

INTERFACE TRAPS. See Surface states.

INTERFACING, MICROCOMPUTERS. See Microcomputer applications.

INTERFERENCE, COCHANNEL. See Cochannel interference.

INTERFERENCE IN WIRELESS COMMUNICATION SYSTEMS. See Cochannel interference.

INTERFERENCE, SIGNAL. See Symbol interference.

INTERFERENCE, TELEPHONE. See Telephone interference.