support the construction of tools for failure-analysis or hierarchical alarm structures.

Most of the mentioned routines are already available today and are used in industrial settings. What we still lack is a *methodical integration* of these tools. It is still quite difficult to have different programs run together, although the situation is becoming better because of the standardised user interfaces and operating system environments. The combination of windows-based user interfaces, object-oriented programming concepts, process databases, standard notations for automated functions and special applications like process simulators and routines for signal analysis probably indicates the direction for the developments of the next years.

3.8 Conclusions

Today's computer systems offer advanced data-processing capabilities, that are mainly directed to fulfil technical requirements. Human-related issues come to a large extent still on a second plane. There are user interfaces that take in consideration the cognitive capabilities of the process operators with use of advanced and standardised display interfaces (Microsoft Windows, X-Windows), but in general the cognitive needs of the user in relation to the process are not getting the attention they deserve.

To design a process interface that is oriented to the user, the cognitive capabilities of the user must be considered as the starting point. The computer should not be a substitute for the tasks that are already performed well by humans, but instead enhance those tasks where humans are less capable.

In this chapter it was presented the hypothesis of the process computer as *complexity interface* and as a *tool for the exploration of process data*. If the complexity for the control of a technical process is higher than what human capabilities allow, the computer has to *reduce* this complexity to a level where it becomes manageable and matches human cognitive capacity as well as the task to solve. The computer can also act as a powerful tool to explore the data from the process and therefore not only support process-control tasks but also help in understanding and learning about the process itself.

outcome is as wished, then a real command could be given to execute the action in reality.

A simulation routine - even a very good one - cannot completely match the real evolution of a complex process. A complementary routine could therefore be used to analyse and compare the simulated and the real process evolution. In this way one would at the same time gain better insight in the technical process and use this knowledge to improve the simulation routine itself.

Another aspect where humans have problems is finding causality relations among different parameters. The "exploration" tool should then be so designed to allow the analysis of interactions among process variables. Because of the intrinsic time delays in many processes, the tool should allow correlation of data from different sources an for different time intervals and time/phase shifts.

An essential part of a comprehensive, human-oriented "exploration" tool would be the support of different types of visual presentations on the computer screen. Many relations that cannot be found by formal analysis become apparent if they are visualised in the proper form. With such a visualisation tool, the construction of an entropy/temperature diagram as mentioned in an earlier example should be a straightforward operation. Shneiderman (1991) reports how visual interfaces with real-time access to databases and immediate presentation on screen can be a powerful exploration and learning tool. However, this technique is still very young and there is not equivalent example for process control.

A final consideration about a generic monitoring and supervisory tool that has been raised by many users is the importance that the operation of the tool itself is known. This means that the users should have access not only to the process data, but also to the algorithms on the base of which the process data itself is processed and displayed.

Modern control languages (e.g. Grafcet, IEC 848, IEC 1131-3) allow the definition of logic connections on a computer screen and then carry out the related automated operations. A similar representation tool could be used to define the interconnections among the variables and further the conversion of some variable types into other types for display as process data. The use of standardised and known logical and procedural notation could also be useful to

At this point, the aspect of the interface must depend on the scope of responsibility of the process operators. This is defined via the task analysis, which must also indicate the type of knowledge the operators must have to perform their tasks. If only *operational* knowledge is required, then the user interface has to be oriented to support the necessary system operations. If instead *conceptual* knowledge is required, then the operators have access to process data and to the related controls in order to (1) build and refine their mental models and (2) to support reasoning in unusual (e.g. emergency) situations.

The very different problem-solving attitudes of people and machines must be considered. Computers are good at doing mathematics and in precise control; their use in NC-Machines is therefore entirely warranted. But computers cannot draw conclusions from elusive clues, unless they are specifically programmed to do just that. And even if computers can be programmed to solve specific problems (e.g. character recognition), they are still very far from solving cognitive tasks at large. For that kind of problem, humans should be in charge. The technical system should be kept as simple as possible and designed from the beginning to *support*, not to *replace* humans.

On the side of the theoretical knowledge provided via courses and handbooks, also the computer can be a very powerful tool for learning by allowing the direct exploration and manipulation of the system. The user interface must support the analysis of process data and trends, as well as the search for relations among different variables. The interface could be designed to support answering to questions of "what-if" type, in a way not much dissimilar to how spreadsheet programs work. Thanks to spreadsheets, tasks like budgeting that earlier appeared to be boring and difficult have turned into a much easier and more playful issue. The reason lies probably in the intrinsic *simplicity* of the tool, even if it is used to manipulate complex data structures.

Real-time process data and the internal couplings in processes are intrinsically more complicated than static budgeting equations, so that the analogy with the spreadsheet might have its limits. Yet, the ideal tool would have to contain in some form simulation routines to study the dynamical evolution of a system, and thus support the problem of predicting its evolution. The simulation routine would take the real process data as input at the start of each simulation. The operator might test beforehand the outcome of a command on the process. If the

on the screen layout in relation to their actual spatial location. The representation would be on a one-to-one basis: temperature values, valve open/closed, etc. An alternative way to look at the process state would be to display a Rankine cycle on a entropy/temperature diagram (Beltracchi, 1987). With a current monitoring and supervisory package, it is quite straightforward to define a screen page to display the hardware and the data from the sensors, but the Rankine diagram would probably represent a major undertaking. The user might have to write major portions of code in C language, so that the additional effort could turn out to be economically unfeasible.

Suggested developments

One of the problems in the definition of the user interface results from the fact that humans and computers are considered to be equivalent and therefore implicitly interchangeable. It is heard often that "computers replace people". This formulation is however not correct. What computers do is to carry out functions similar to what some people did before, but in a different way.

The very question has to be stated in a different way from the beginning:

- what do humans best
- what do computers best
- how can computers support humans in what they have difficulties with.

The "ideal" process monitoring and control computer would then basically enhance the qualities that humans either do not possess or possess in limited capacity. Some of these qualities - those that are of relevance in process control - are the following (see also Section 2.6):

- most people tend only to react to contingent data, without planning for future actions
- people have difficulties with exponential developments; when they extrapolate trends, they usually do it in a linear fashion
- people usually do not recognise delays in the process they control and that such delays may lead to instabilities
- people tend to think in causal series, not in causal nets
- people tend to reduce the complexity of the problem they deal with to fewer and fewer causes.

programs is more difficult and usually takes longer time, but allows a much higher degree of freedom.

In Section 3.3 the hierarchic structuring was described as a powerful organization method, especially suitable for process data. The computation of the general aggregated system variables takes place in the monitoring and supervisory system, that has to be programmed in this respect. In order to implement the desired strategy for data structuring, the monitoring and control system must provide the right kind of support.

Current supervisory systems (as advertised at beginning of 1993) basically still support the "one sensor - one indicator" principle. With few exceptions, the definition of internal variables (**derived variables**) whose value is function of other variables or process values is difficult. Standard packages provide data structures for measurement information and allow the definition of ranges, e.g. with a minimum and a maximum value, and the control of the colour of a symbol on a display screen. Many features to support programming and operations are very well conceived and designed. Yet, with these packages the computer does little more than replace panel instruments. The functionality of the user interface is due to the concentration of the data, not to a new way to look at them.

A dozen supervisory and control systems presented at the Hanover Industry Fair 1993 may represent the state-of-the-art in current technology. Their capabilities reflect what the market is supposedly currently asking: remote control of PLC (*programmable logic controllers*), integration in hierarchical networks or fieldbuses, plotting, alarm logging, PID-control, etc. For most of these products the user interface is bases on "Microsoft Windows". Programming - even at applications level (objects, communication with sensors, etc.) - is done with OOP (*object-oriented programming*) concepts. For those cases where special solutions are required, "C"-interfaces are available on many of the systems. These packages are mainly oriented toward the solution of problems from the point of view of data collection and processing.

We can exemplify this concept with the following example. A user interface for the control of a nuclear power reactor could be designed to represent the hardware directly. All the data collected by the sensors would then be displayed

system. It is interesting to note how many process operators already carry out this task on their own, to make up for the boredom of supervisory control (Olsson and Brehmer, 1990). Overconfidence in the power and capabilities of technology is to be avoided with all means.

3.7 What Should We Expect from Process Control Computers ?

Current technology

The considerations in relation to user psychology, task definition and technical process must at some time be translated into the design of the user interface. The way the work is organised and conducted depends largely on the tools at disposal for the realisation of the monitoring and control system.

In general, for the design of process computers two ways can be followed. In one solution, the process computer is programmed from scratch, using a language like Pascal or C and with help of software tools like screen editors, mask generators and similar ones. The other solution is to make use of a standard monitoring and control package. With standard packages, the designer is concerned only with the description of the technical process, its sensors and actuators, but not with procedural steps for the actual data collection in real-time. In addition, it must be considered that, especially in large plants some system or new construction always take place; the monitoring and control system must also be adapted and reprogrammed to follow such changes. There is no "final" state, but rather different degrees of functionality (databases for process control are described in detail in Olsson and Piani, 1992 and 1993).

There is a tradeoff in the approach of developing an *ad hoc* system compared with the use of a general monitoring and control package, and that tradeoff is between cost/time and flexibility. General-purpose packages are tailored to the intended application with help of parameters, so that the user does not have to concern himself with step-by-step program sequences. They are therefore usually easier to program, but with them it is usually more difficult to do anything that lies outside the initial scope of the solution. On the other hand, writing procedural

lamps. Some pilots complain today that with the new CRT-displays they miss the immediate information about the whole aircraft that they had with the conventional panels (parallel information). An additional criticism is that with the CRT-display, the controls are no longer located in proximity of the respective indicators. The pilot must know where in the cockpit they are placed and reach for them, looking at the CRT-screen in a direction and acting on the controls in another.

The comments by the pilots let arise a question. Control panels have evolved during time, and task analysis aspects must have been taken in consideration more or less explicitly. The decision to place together the indicators and the controls for each subsystem is ergonomical and must not have happened by chance. In the case of the CRT-system displays, it is pertinent to ask whether a task analysis has been carried out before their introduction in the cockpit, or if it was tacitly assumed that a control computer would represent a better solution anyhow.

It is difficult to provide a clear-cut answer to how far one should rely on computers in control of complex systems. It is probably the very initial questions that have to be stated in a different way:

- what are the requested tasks for regular operation
- what range of freedom is left to the operator to act in case of emergencies and unexpected situation (notice that the question is not posed as *what tasks are required* in emergency operations. If unexpected situations could be planned, they would not represent unexpected situations longer, but rather one type of operation).

These questions are closely related to those addressed in task analysis. There is therefore no definite and general answer, but the answer has to be found for every case depending on the user and the type of task to accomplish. An explicit choice about the desired level of automation has to be made, instead of supporting automation and supervisory control at all costs. If the control system is designed to keep the technical process stable, then the process operators must know that they have to compensate for what is saved in time and efficiency (compared to manual control) by critically evaluating all information from the

On aircraft almost all systems and subsystems are directly powered from the engines. Probably as a consequence of the engine failure, the screens for the Captain were blank; he had to fly the aircraft with emergency artificial horizon and speed indicator. In the MD-82 there is a system display that shows problems according to their priorities. Because of the multiple subsystem failures and the steady electrical switching, the screen changed the display data all the time; the effect was similar to looking at data scrolling quickly on a computer terminal. To this respect, Capt. Rasmussen himself came to word in an interview (reported in VC Info, 1992):

> "Q. Stefan, this crash was not your first emergency. As you told me before, you have already been confronted with engine failure in a DC3 during an Atlantic flight, with 'contaminated oxygen' on a T38 and again with engine failure on a F27. What was the most stressful situation in all these emergencies, including the Arlanda crash?
>
> A. The uncertainty! The confusion and the loss of faith that were caused because of the nature of the electronic instruments that in this situation could not provide any relevant information. The system panel turned into a crazy, senseless colour play."

The main issue in this situation was to identify the problem source as quickly as possible. Paradoxically, the computer system designed for this task turned into an hindrance for the pilot to determine that the engines had failed; he would probably have deducted this information better from an older-type control panel. The failure to identify the problem source in time meant that no action could be taken either.

The system display computer might be enhanced to include a priority scheme in the selection of the data pages. With such a solution, engine failures would get higher priority than e.g. cabin air conditioning. But how long should one screen page be displayed when all subsystems are alarmed and new alarms are generated all the time? And in case of major failures, should only the power subsystem be displayed? And what about if a different subsystem is the cause for failure?

The aircraft system CRT-display is an example of task mismatch. On conventional control panels, all units in each subsystem are grouped together; switches and fuses are located close to the related instruments and warning

stable picture: Some pilots now believe more this indication than the older instruments and do not look critically at the computer-generated data.

The map display is an example of a system that - if it were reliable all the time - would simplify the pilots' task. At present, many pilots perceive the map display as a too complex device that does not really support the task at hand. If a pilot really wants to cross-check the navigational display with help of the other aids, his workload becomes higher than by navigating with the older instruments alone. Another element of uncertainty is that it is not openly documented how the system works. Although it is unlikely that knowledge of the internal algorithms would help the pilots to better identify map-shifts, it would probably provide clues about the function - and thus the intrinsic limits - of the Flight Management System.

System failure on board

In modern "glass-cockpit" aircraft the state of the board subsystems (electric, hydraulic, engine, fuel, air-conditioning,...) are shown on a small CRT-display. This display replaces a control panel with a large number of indicators, lamps and buttons. With the CRT-display, the pilot can select the system / subsystem of his choice, and the related data is shown in graphical form on the screen. Moreover, if some subsystem indicates a malfunction, the appropriate screen page is automatically selected for immediate presentation. The system also selects the display pages automatically depending on the flight situation. For example, immediately after take-off the indication of the landing gear is shown until its retraction is completed.

On December 27th, 1991, a MD-82 airliner started from Arlanda airport in Stockholm with destination Copenhagen. Four minutes after take-off the plane crashed on the ground. It was mainly due to the skills of Capt. Stefan Rasmussen that nobody was seriously hurt or injured and all passengers survived the crash. The cause of this accident was ice formation on the wings; some ice was also blown into the engines, that were damaged and malfunctioned. This aircraft was equipped with electronic cockpit instrumentation, including a CRT-display for on-board systems.

on a CRT-screen in the so-called "glass cockpit". For the pilot is it very comfortable to use this map display, which shows all NDBs, VORs, airports and airways relative to the aircraft's own position and is constantly updated, compared with navigating the aircraft by conventional instruments with their needles and pointers, and that require a more difficult interpretation.

The accuracy of the map naturally depends on the accuracy of the position computed by the Flight Management System on the base of data from the INS. For example, after a longer flight over water and out of range of land-based navigational aids, the position is no longer updated and the map picture must be expected to drift away ("map shifting"). This effect can also take place over land and during the continuous update with navigational aids because of changes and disturbances in the radio signals.

The FMS computer is at the present unable to handle all these possible errors. The only way to detect a map-shift is that the pilot cross-checks the displayed output with conventional navigational information. An undetected map-shift may lead to navigational errors with all their consequences, including the flight into obstacles. Pilots have reported several dangerous cases where map-shifts almost led to accidents.

The pilots know that they have to cross-check the information from the Flight Management System with help of conventional navigational aids. However, performing this check all the time means an higher workload compared with conventional navigation. High navigational accuracy is essential mainly when flying at low altitudes during approach or departure, and during these flight phases the cockpit workload is high anyhow.

The comfortably looking map display suggests a high level of navigational reliability, which leads to a high preference for this instrument. Yet, even after having detected a map-shift by cross-checking with conventional navigational aids, many pilots experience a "psychological barrier" to believe that the computer-generated display may be wrong. With the old instruments, shifts and oscillations in the pointers or uncertainties in digital displays provided an immediate psychological clue for the pilot that the data was approximated; the pilots were thus unconsciously warned to check the data all the time. The electronic display was designed to be as comfortable as possible and provide a

the display information on travelling teenagers. What about a display of the type "*This train is at the moment developing a power equivalent to that of X Ferrari race cars*", with *X* updated in real time. The ICE Bord display is an example of mismatch in the human interface, where the displayed information and the *interest* of the user are not the same. This display was designed without taking in consideration real-life human factors.

Aircraft navigation

Aircraft navigation is a complex skill that requires the integration of different types of information. Conventional aircraft navigation is accomplished by using compass indications and radio signals from land-based navigational aids with known geographical coordinates: NDB (Non-Directional radio Beacon), VOR (Very high frequency Omnidirectional Range), ILS (Instrument Landing System), DME (Distance-Measuring Equipment), etc. The information from these navigational aids, compass headings, VOR-radials, DME distances, and so on are displayed with dedicated cockpit instruments like the compass rose, needles, pointers and digital counters (for DME distances). From these indications the pilots derive position, course and headings and determine the input signals for the autopilot.

More modern aircraft are equipped with Inertial Navigation Systems (INS). The main components of these systems are high-precision accelerometers. Starting from the known initial coordinates at the airport terminal building, the speed and the spatial coordinates of the moving aircraft are computed by integrating the acceleration values in course of time. But because of the mechanical inaccuracy of the system hardware (accelerometers, gyros, cardan-gimbals), the position indicated by the INS may "drift away" and thus become more and more inaccurate. This inaccurate position can be corrected and updated with help of conventional navigational aids.

Newest aircraft are equipped with a so-called Flight Management System (FMS), a computer that basically uses the navigational information of the INS system. In addition, the FMS uses the information of conventional radio navigational aids (when they are available) to automatically update the position information indicated by the INS. By combining the current position (as computed by the FMS) and a navigational data base, a map picture can be created and displayed

effect of reliance on the machine, that makes the human user less attentive to discover on his own discrepancies with normal states. When a failure occurs, it might be too late.

To exemplify how even well-designed control computers can hinder manual operations in emergency cases, we will refer to two actual experiences from civil aviation. These examples show how difficult it is to foresee all possible problems in advance. Civil aviation is also a good starting point because it is one field where safety and testing standards are very high.

On-board display on trains

A good example of poor complexity- and goal-matching in an human-computer interface is the information display installed in the new German ICE (*Inter-City Express*) trains. The problem with these displays is not related to the screen colour or resolution or to the menu keypad (they are actually very well designed and pleasant to operate), but rather to the fact that *dynamic* displays are used for the representation of *static* data. The typical traveller is most of all interested in the actual speed (especially when the train goes fast) or context-related schedule information like connection trains, whether they are delayed, and so on. Unfortunately, none of these data can be recalled on the onboard information display. Instead, information such the current number of passenger and freight cars of the *Deutsche Bundesbahn* or the lexicon-definition of what a regional train is can be called up, all information that would have better found its place in the *Bord Journal*. Sometimes the current train speed is shown on the display and the data is fixed for some tens of seconds, without update. However, the selection of this display page is not under the control of the user.

Several times I could observe passengers standing in the aisle and trying to select the display page for the actual train speed (which, as said, cannot be selected explicitly). The real-time display of current information about speed, engine power, current drawn from the power line, distance to the next station, and so on would be of higher interest and would probably have a psychological impact, especially on young people. It is interesting to notice that at the same time, the railway administration advertises for job openings as train conductor, trying to present it as a *dynamic* profession, but using *static* posters ("Berufswahl Bahn"). They did not consider the subtle, implicit (and probably more effective) impact of

Some of the problems related to the parallel presentation of information on the same screen page can be solved by using windowing systems. However, the windowing system is not a solution itself, it is a *tool* to support a solution. Even with a windows-interface it must still be defined - on the basis of the process and task analysis - what data has to be presented, what parameters can be moved from one window to another, in what form, etc. The identification of this data and how to manage it is not automatically done by the tool, but can only be performed by the designer and by the user on the basis of motivated choices.

The task analysis provides an additional way to look at a problem under a different angle. The space on a control panel, or CRT-screen, is limited. With knowledge of the task it is easier to select the information required to carry it out and present it in a compact form. The task analysis gives therefore indications about **views** (subset of states that are selected for presentation) oriented to the process operations.

In conclusion, task analysis provides part of the background that is necessary to design the human-computer interface according to the user's needs. The task analysis will be oriented to the knowledge of how much operational and how much conceptual knowledge the users of a system are supposed to have.

3.6 Examples of User Interface Mismatch

We have hence presented a simple conceptual approach to the evaluation of the matching between technical process - user interface and operator. It is now instructive to look at examples that have some kind of mismatch. The mismatch is at times not immediately evident, and even designs that were intentionally good turned out to have unexpected problems. We do not want here to classify problems in a comprehensive fashion, but rather to show how many problems have dimensions that elude the initial scope of the user interfaces: problems can be smarter than their solutions!

In several occasions it has been pointed out how complex supervisory systems simplify routine work but can make complex tasks even more complex (Bainbridge, 1982, and Norman, 1990). To this is added the subtle psychological

The analysis of the technical process together with the task analysis provide the foundation for the organization of the user interface to the process (this interface doesn't necessarily have to be a CRT-display, it might as well be a control panel with lamps and switches).

In the "classical" representation of a process on a CRT-display, subunits are displayed separately, each on a dedicated screen page. If the task analysis indicates the necessity for working with certain data in parallel, these data must then be showed together on the screen. Otherwise, when the operator has to intervene on several subunits at the same time, he could not visualise them concurrently and would have to jump from one display page to the other. Cases have been reported where operators had to write down data from a screen page to input it again in a different one. A very peculiar situation, considering that one of the things computers do well is to transfer data internally fast and efficiently[1].

Part of the criticism related to the introduction of computer-based user interfaces at the place of traditional control panels may be due to a **task mismatch**. For example, operators in some chemical plants have complained that with old panels all information they needed was immediately available; with the computer screens, instead, they have to jump across pages in order to find it. This is the result when information is organised only following the hardware structure and not to the operations to be performed.

---

[1] I have experienced this situation several times - and in different contexts - the last time in March, 1993. I had to book a flight to the US, so I called two (very well known) airlines to define the route. All data about flights, airports, times, seat preference, etc. were stored and the bookings confirmed. When I went to the travel agent to collect the tickets, they were able to recall all booking information on their screen, but not to make any change to it, neither to issue the tickets. A safety measure, they said, to hinder unallowed changes. They had to manually write down the data from the screen to paper, call the airlines, let them cancel the reservations and then type in these data once again. The whole procedure took one hour and a half. I was at the same time amused and annoyed to look at the terminal screens in the office, the modems, the printers, the telephones, etc. and consider how all this advanced technical equipment did not help to make the work easier.

3.5 Task Analysis

The acting entities in a technical system are the technical process (tool), the goals to be reached and the human user (Section 1.2). In order to design of the appropriate user interface one more issue has to be considered, namely the way in which a task is solved. Task analysis must address both how work is performed without computers and how tasks are expected to be solved with the computer.

The ways in which work is carried out are several, and it would be unrealistic to propose here a general approach to their analysis. In more realistic terms, the designer of a user interface should not only concentrate his attention on the process to be controlled, but also how the operator carries out the task.

A formal task description could take the form of general indications or also reach the detail level of the recipe for a batch chemical processing. A task description would contain all operations in the control of each step of the technical process in the same form as it could be used to automate the task ("open valve A, load 15 l of component B,..."). In addition, the task description would focus on aspects like where the valve is located, what information is needed before and after the action, how does the user know that the action was successful (e.g. by monitoring a flow meter), etc.

The purpose of task analysis is dual. On the one hand, it provides a more or less formal and deep description of what has to be done from the point of view of the user. The user interface can then be designed on the basis of the task analysis document. On the other hand, the task analysis helps the designer of the user interface (who almost never is the same person as the user) formulate and understand better the needs of the user.

The task analysis will have to identify the **typical** and the **critical** work tasks. For the first, the analysis is quite straightforward. For critical operations instead the problem is the difficulty to know beforehand how critical they are. And unplanned situations cannot be planned by definition. Critical tasks that can be identified should then be analysed beforehand. For all other tasks that cannot be foreseen and identified in advance, the margins of freedom and the type of information that an operator must have should be defined.

keyboard dialogues are superfluous. Why typing "*SET DEVICE#2=ON*" when a simple switch fulfils the same function? If the process is more complex and includes several parameters, then the user of a keyboard is warranted ("*SET DEVICE#2=ON, POWER=MID, SETPOINT=3224*").

A mouse, a trackball and a joystick are cheap and simple input devices that can be used for the fast control of processes with an immediate feedback: mouse and trackball for pointing at objects on a computer screen, the joystick for the remote control of a mechanical actuator (e.g. a robot arm). All these devices are not much precise, but allow fast corrections. The human acts as "feedback correction" either from the visual input from the screen or by watching the position of the mechanical actuator. The use of a joystick to control the set-point temperature in a chemical reactor would be a mismatch, because the joystick without feedback is not precise and because the slow dynamics of a chemical process does not require the use of a fast input device. A potentiometer or, for even better precision, a digital input device (keypad) would represent a much better choice.

The last element in the chain between goal and user is the user himself. Factors that influence the aspect of the interface are previous experiences, the use of other interfaces (which means that there could be a transfer of operational knowledge, Section 2.7), the mental models about the technical process, knowledgeability and disposition in the use of the computer, as well as many others. Very important in this respect is the user's decision scope: How much must process operators follow predefined guidelines and how much should they take their own decisions; where are the limits? The interface should therefore be oriented to the required user competence (at rule-based or knowledge-based level) and the type of work. A complicated interface must be evaluated against the necessity for training. Only when all components in the chain user-machine-goal fit one another, then the goal can be reached with the optimal approach of user and machine.

control signal of a regulator might exceed the value it normally takes and could therefore be an indication for malfunctioning, but this would not appear on the general indication for the device, that shows "green" as long as set-point and actual values are similar. In addition, the process operators might have to compare low-level data with the higher-level representations to check that the supervisory system is working adequately, which would also add to the workload.

On the one hand, complexity related to routine operations has become smaller; on the other hand the scope of the work is now wider. The user is probably relieved from a large part of the control tasks at plant level, but must now dedicate increased attention to monitoring the supervisory system. In addition, the distance from the actual devices makes it more difficult for the user to create his own mental models from sensomotoric data.

Complexity matching

The second main consideration together with complexity reduction in the human-computer interface is **complexity matching**. This concept is analogue to the impedance matching in the connection of electrical devices in cascade: the optimal transfer of energy takes place when the output impedance of a circuit is equal to the input impedance of the cascaded circuit. Already at the hardware level, the user interface must match the amount and the precision of the data that have to be transferred. Amount and type of data and user's skills determine ultimately the type of interface.

If a process generates only a few events per hour and the number of the input and output parameters is small, there is no reason why a small printing terminal or control panel can't be used for monitoring and control. The installation of a complex process computer for the control of a *simple* process would not contribute to complexity reduction but rather to an increase, because the complexity of the control system is now added to that of the technical process itself. If the technical process does not generate enough data, a screen display would act boring and uninteresting.

The same principle of complexity matching holds for the control interface from the user to the machine. If the command input consists only of few bits, then

Both solutions have in common that there is a mapping between the controls and the circuitry in the refrigerator. In other words, a functional model of the refrigerator is needed somewhere along the chain user - technical system - goal (see Figure 1.3). Somewhere an inverse mapping (the predictive mental model) must take place. If it doesn't take place in the technical system, the user must do it. In conclusion, if we want to avoid that people build their own mental models, the relevant mappings must be realised, in some form, in the machine.

Complexity reduction in large processes

The number of sensors and actuators to monitor and control a process cannot be defined arbitrarily, but depends on the nature of the process and on the control goals (the formal description of this concept is due to Kalman and is well known in control theory; practical issues are discussed in Olsson and Piani, 1992).

The first step in complexity reduction is the analysis and structuring of the process data on whose base the user must act. We have seen earlier (Section 3.3) the example of a chemical reactor and that its complexity depends on the level of detail at which it is considered. The value of every process state could be displayed on a panel, but the user would probably have to write off these values manually and then use them in some computation. This represents - useless to say - a certain workload for the user. What the user is actually interested in (the goal) is whether all current values are close to their respective set-points, whatever these might be. The comparison of hundreds of monitored data with the related set-point values can be easily done by a computer several times per second. A Boolean connection could then deliver a general indication, whether the reactor as a whole operates correctly or not. The user is relieved of manual work, yet he is free to check the actual data at the desired definition level, if he wants to. A single operator might then supervise several reactors at the same time. The computer can also take into account special operational situations. For instance, the differences between actual values and their set-points during transients are expected, so that they do not need to be considered as alarms. In a computer takes all these factors into consideration, it contributes to complexity reduction.

The transfer of work from the reactor to a control room does not entirely relieve the operators from their earlier workload. The hierarchical structure simplifies the supervision of a large process, but hides some data from view. For example, the

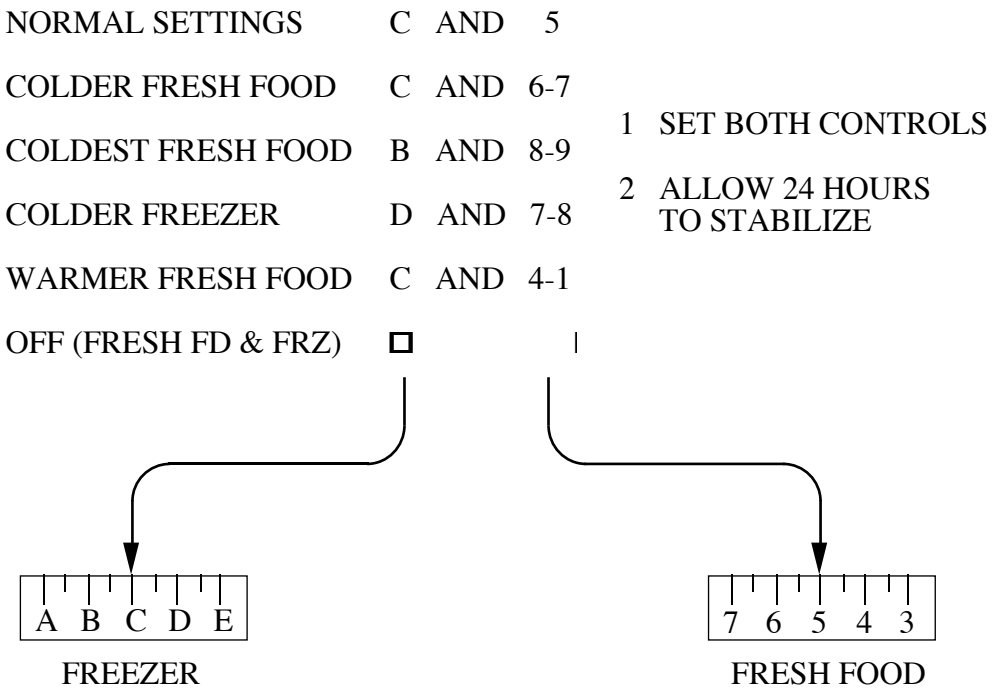
In this specific case we have two internal, or state, variables (the temperatures in the two compartments that depend on the position of two valves), two external variables (the scale indicators) and a time constant (24 hours). The internal variables are the same as the goal (the desired temperatures). Why is this problem so complicated then? We don't know the relation between the control settings and the internal states and because the time constant is so long. If manipulating the controls led to an immediate change in the temperature, we could just "fine-tune" the refrigerator. We would also get an immediate and spontaneous "mental model" of what setting does what, probably even the instruction were superfluous. But who remembers yesterday's setting? This is a good example of how the internal connections may add a lot of complexity to a system that in principle is quite simple.

On the other hand, with knowledge of how the refrigerator works we could figure out how to set the controls to get the desired temperature. This means, however, that we should have conceptual device knowledge as opposed to the simple operational knowledge that would make sense for an application like this.

This problem can be solved with an appropriate user interface, for which two basic designs are possible. In the first solution, two separate thermostat scales marked in degrees (°C/°F) are connected to the valve controls in the refrigerator and map the preset temperatures (the goal) to the required position of the valves. A label would warn the user that the selected temperature will be reached latest after 24 hours.

A different solution is to use a microprogrammed circuit to simulate the operation of the refrigerator and display the temperature outcome of the current setting after 24 hours. The display reading would then mean something like "*if you use that setpoint, in 24 hours the temperature is going to take this value*". In this case, the user could "fine-tune" the refrigerator because the response is immediate.

In both cases the problem of the user's mental model of the actual circuitry in the refrigerator is eluded and thus avoided altogether. The user does not need to know *how* the refrigerator works, but just what temperature he wants to have.



My house has an ordinary, two-compartment refrigerator - nothing very fancy about it. The problem is that I can't set the temperature properly. There are only two things to do: adjust the temperature of the freezer compartment and adjust the temperature of the fresh food compartment. And there are two controls, one labelled "freezer," the other "fresh food." What's the problem?

You try it. Figure 3.1 shows the instruction plate from inside the refrigerator. Now, suppose the freezer is too cold, the fresh food section just right. You want to make the freezer warmer, keeping the fresh food constant. Go on, read the instructions, figure them out.

Figure 3.1 Norman's Refrigerator. Two compartments - fresh food and freezer - and two controls (in the fresh food unit). The illustration shows the controls and instructions. Your task: Suppose the freezer is too cold, the fresh food section just right. How would you adjust the controls so as to make the freezer warmer and keep the fresh food the same? (from Norman, 1986)

adequate for the control task. The user can control the technical process only as long its complexity does not exceed his cognitive capabilities. The computer - an information processor - can be used as interface component for complexity reduction.

Complexity reduction

The first goal of the process control system is the **complexity reduction** of the technical process. In other words, the technical process "seen" through the computer interface must be simpler that the technical process seen through the conventional instrumentation. The computer should not add complexity on its own and servicing it must not represent an overload of the cognitive capacities of the process operators.

At present, monitor and control computers are used to replace process instrumentation, but in most cases they still follow the "one sensor - one display" approach. The main advantage in their introduction lies therefore in the more comfortable organization of the workplace and in the automatic logging of process data. What is lost on the part of the operator - if compared with conventional instrumentation - is the direct, "tactile" perception of the equipment. In the real world the difference between grams and tons, kilowatt and megawatt, millilitres and hectolitres is immediately evident; on a CRT-screen, instead, all devices and machines have the same size: large differences may be reduced to the position of decimal points.

Example: Norman's refrigerator

Norman (1986) describes a simple problem that can be used as a basis to examine many of the issues involved in complexity reduction and user interface design (the same example is also reported in Norman, 1988). Norman describes the interface of the thermostat control in his home refrigerator and declares it as a bad design, because "it provides a false conceptual model" (Figure 3.1).

In the discussion about complexity (Section 3.2) we have seen that there is no objective measure of complexity for a system, but that the number of observed system variables, the internal system states, their interconnections and the time constants provide a reasonable indication of the complexity of a problem.

Looking at the table of hierarchical management levels, it also appears where the human role is best and where the use of machines is warranted. At process control and field level, the time constants are of the order of milliseconds; no human can carry out any tasks at this speed. At cell control, time scales of seconds suggest that humans might carry out operations, but this would remind of Charlie Chaplin in *Modern Times*. (Driving a car also involves response times in the order of tenths of seconds or seconds, but it can be performed because many actions take place at the skill-based (sensomotoric) level and the physical effort is limited.) Only at production control and at management level humans can perform the related tasks, being able to cope with the slower timescales. At management level there are large quantities of data to process, but usually this data is reduced with help of statistical methods - or aggregate parameters - like histograms, pie diagrams, etc. The data is brought to a level matching the cognitive capabilities of the humans working with it.

In conclusion, a conscious human effort is necessary to reduce the intrinsic complexity of a system, an effort that if it is not carried out beforehand by the designers, is left later to the process operators. Due to the fact that it is difficult to define complexity in an objective way, it is also not straightforward to define how to reduce it. A generally feasible and often used approach is the identification of hierarchical levels to group subsystems, systems and functions; other ways may of course be possible (further considerations in this sense are contained in Rasmussen, 1985). Here is where human intelligence and intuition play a role that no "artificial intelligence" or predefined method can approach. Only knowledge of a task and the work method can indicate how the problem can be solved.

3.4 Computer Control Systems as a Tool to Deal with Complexity

To study the application of computers in process interface and control we will work on the following hypothesis: Every human-computer interaction problem can be considered as a cognitive problem in the operation of an unknown, often complex system. The user wants to solve a problem or reach a goal with help of the technical process. The user follows a mental model that more or less reflects the real technical process and that, with sufficient experience and knowledge, is

Management level	Data quantity	Response time	Frequency
Management	MByte	days	days
Production control	kByte	seconds	hours, minutes
Cell control	Byte	seconds, 100 ms	seconds
Process control	bit	ms	ms
Field level: sensors, actors	bit	ms	ms

Table 3.1 The hierarchical management levels for a company and the related information requirements

point and actual temperatures, flows, concentrations, etc., are of interest. If the same reactor is considered in the context of a wider processing unit, then only the influent and effluent rates are relevant, together with the knowledge of whether everything is OK or not. Only when a failure is noticed at higher levels, it makes sense to look at more detailed data about the reactor to find the reason.

In the chemical reactor, the lowest level is the heat regulation loop where it is decided whether additional heat is necessary or not in order to keep the reactor temperature constant. The "decision" is made by a regulator on the base of the actual and of the reference temperature. At a higher level it is defined at which temperature the process must operate. At a still higher level is selected the chemical process to be run, etc. Obviously, it does not make any sense to have a very stable and optimal temperature for process A, when process B is run. The hierarchical model is decentralised: the decisions influence each other, but each level is independent in the choice of how to carry out these decisions. The control loops would then be carried out manually or automatically, with a higher trend to automation at the lower levels.

to another, but in general humans cannot cope with complexity above some levels.

The amount of information about a technical process has therefore to be reduced and brought below the cognitive limits of the people who have to deal with those data. The intrinsic system complexity can be reduced with an intelligent human effort, that can be made by the plant designer, the system programmer, or the process operator. If plant designers and system programmers do not take care of complexity reduction, this task will be left to the operators.

There is no obvious and immediate way to reduce the complexity required to describe a system. In general, one wants to identify patterns and structures, so that many details in the system can be described by a few aggregate parameters. The identification of the right type of structure and systematic description is a task that only humans may perform. This task is also primarily the responsibility of the designers of the technical process and of the user interface.

The most direct type of structure is hierarchic. In many complex systems it is possible to define a hierarchy where the system's components are organised together and are described by common parameters. The hierarchical levels correspond more or less to the type of decisions that are needed for the control of a process. In general, all entities located at the same level have intensive mutual data exchange; the data exchange between levels is usually reduced and not time-critical.

An example of hierarchical model is that of a company; the hierarchical levels for a manufacturing company or process plant are reported in Table 3.1. How many the decisional and operational levels are, how much these levels depend on each other and how much autonomy is left to the single entities varies from case to case. In the table are compared the typical data quantities, the response times and the frequencies for new actions at the different hierarchical levels of a plant. We observe today a wide use of computing equipment at every layer and a tendency to vertical integration among layers.

The hierarchical classification is useful also in the analysis and structuring of process control. In a chemical plant a reactor can deliver dozens of monitoring data. When the attention is focused on the reactor alone, then the values of set-

of complexity of a system. An expression like "200 monitoring variables, 80 control parameters, 25 system states that must be inferred, 12 quality-related parameters, 8 internal variables with time constants of hours, all others with time constants of a few minutes" is immediately understandable. This indication is by far not *correct* in a strict sense, but - lacking a better alternative - it fulfils a purpose.

For practical human-computer interaction, this complexity definition must be further qualified. Complexity is at the end the result of both the structure of the system and the way the human operator understands it. Systems with the same number of internal variables may show very different degrees of complexity in relation to the human user. The only workable definitions are therefore behavioural.

The definition of complexity as behavioural has an important consequence: complexity cannot be estimated beforehand, but can only be evaluated with practical tests. In this way, an indication of complexity could be the *time* or the *number of steps* that the users take to perform a task; all these parameters can easily be recorded on the computer where the simulation tests are run. The problem with this procedure is its cost and the time it requires to organise and carry out the tests. On the other hand, this method has two advantages. It eludes the problem of an objective measurement of complexity and includes the human component directly. The result is therefore a more direct indication of the performance of the users interacting with a system and can be used for the evaluation of design alternatives for systems and their user interfaces.

3.3 Coping with Complexity

As we have seen in Section 3.2, there is no general objective and quantifiable definition of complexity. Complexity is in part a property of a system and in part a subjective experience by the user.

From psychological experiments, we know that there are limits to the cognitive capabilities of people (Section 2.2 and 2.6). The exact limits vary from one person

Kieras and Polson propose a procedural notation to be used as a metric in order to quantify the amount and complexity of the knowledge required and the cognitive processing load involved in using a system. The notation they propose is based on production rules of the IF-THEN type and on transition graphs; however, this approach still does not lead to a *quantitative* metric. In their work, Kieras and Polson oversee that the formal description of a system in terms of production rules and graphs is complex in itself. General and understandable measures can only be obtained via computer simulations of the target system and the user of this system, where parameters like the number of rules to learn, the rules effectively used, cycles to compete a task, etc. can be quantified. The method might therefore be valuable for very simple systems, but it is questionable whether its validity can be extended to complex systems like those found in process control applications.

In conclusion, despite the importance of the concept of complexity, there is no universal definition and a related metric. Complexity understood as the quantity of information that is necessary to describe a system depends on the selected physical representation and understanding of the system ("looking" at a gas, do we see molecules or aggregate variables like pressure and temperature?). This question is therefore closely related to what mental model an operator has of the target system; for a person some relations might be obvious, while another might fail to see them.

A practical approach to complexity definition

The complexity definitions advanced by system scientists are still at a very theoretical level and are therefore of little practical use in applications of human-computer interaction. In addition, all complexity definitions we have examined consider the target system as static and do not take its time evolution explicitly into consideration. Yet we have seen that the evaluation of the temporal evolution of dynamic systems is a task many people have difficulties with (Section 2.6), so that in some way time has to be included in the definition of behavioural complexity.

The number of the monitored variables of a system and that of the internal variables that must be inferred or set in mutual relation, considered in addition with the time constants, gives an immediate and practical indication of the order

A metric for complexity

Henneman and Rouse (1986) are concerned with finding a metrics for system complexity. A further differentiation they make is between **structural complexity** and **strategic complexity**. To get additional data, they used an experimental setup similar to the one previously described (the US phone system). Their initial observation is that, when operators decide which path through a system is most likely to lead to finding a failure, they must make a tradeoff between their uncertainty concerning the state of a subsystem represented on a display page and their expectations of finding a failure in that subsystem. From this observation, Henneman and Rouse suggest that an appropriate measure of strategic complexity that reflects the trade-off between state uncertainty and probability of failure is the multiplication of these two metrics.

The problem here is that the metrics obtained are in form of required time and probability of success in carrying out typical tasks. These data cannot be deducted by information about the system, but can only be collected experimentally and with settings that in many cases only in part can represent the real system.

Another direct measure of complexity is the **mean time until failure diagnosis**. Yet this measure alone is difficult to characterise and quantify, because this time depends obviously on the number and the location of failures. Moreover, if - hopefully - there are only few failures in a system, the time would be based on a sample that is not statistically representative.

A different approach to the definition of complexity is taken by Kieras and Polson (1985). They relate the complexity of a device to:

(1) The complexity of the user's task representation, and the learning, memory, and processing capacity demands implied by this task representation;
(2) The number of device-dependent functions, which are not part of the user's initial task representation, and the difficulty of learning them;
(3) The ease with which a user can acquire device how-it-works knowledge.

and switching station to evaluate how to route the calls most economically and efficiently. The system is highly automated, but nevertheless human monitoring and intervention is still a necessity. During overload situations, or in the case of major equipment failures, network performance can degrade rapidly. Human network controllers must intervene when the automatic solutions are excessively expensive or in case of problems that require human judgement. To deal with these situations, the human operator can cancel alternate routing, reroute calls, issue line load controls, and play recorded announcements.

Henneman and Rouse ran simulations of the phone dispatching system, where the test subjects served as the network controllers. The goal is to keep the network in operation in presence of random equipment failures and under varying load conditions. In the experiment were investigated the influence of clustering and the number of hierarchical levels (i.e. the number of subnodes within a major node) the operators deal with. The results indicated that increasing the number of hierarchical levels tends to decrease the quality of performance. With many hierarchical levels, it took more time for the effects of lower level failures to become obvious at the higher monitoring levels, so that their effects tended to be more serious than in the systems with less levels. This lengthened diagnosis time tended to degrade practically all other dimensions of performance. An additional - unexpected - result of these experimental simulations was that users could adopt strategies to compensate for some situations (e.g. increased failure rate), but apparently not for others (e.g. small clusters with constrained resources).

Henneman and Rouse suggest that the results might be flawed because of the experimental conditions. The simulated network is small and each node in a small network represents a larger fraction of the total size of the system than does one node in a large network. In other words, when a node fails in a small network, a relatively large portion of the overall system fails and leaves less resources for handling the remaining customers. The fewer resources to absorb the effect of failures lead therefore to faster propagation of failures in smaller systems compared with large-scale, real networks.

safer against failures. Thus the level of interconnectedness in a system affects then in different ways the level of two types of complexity: problem-solving complexity and system-control complexity. These complexities, however, are in part oriented to the human role and should then be considered under the perspective of behavioural complexity.

Behavioural complexity

Behavioural complexity is related to the human role in the control of technical systems; it can be considered under two aspects: as perceptual complexity and as problem-solving complexity. **Perceptual complexity** deals with "the human's ability to recognise, rotate, reverse, etc. displayed patterns as a function of various attributes of the pattern, including number of line segments, symmetry, etc." (Henneman and Rouse, 1986). This type of complexity can be studied with simple experiments. On a computer screen, the number of displayed components can be used as an immediate measure of perceptual complexity.

Perceptual complexity is a simple attribute, but it has little practical importance. Experimental results in fact indicate that perceptual complexity is no good predictor of fault diagnosis performance. For this reason the concept of **problem-solving complexity** was introduced; for its very nature, however, no formal definition of this type of complexity can be given. This type of complexity can only be estimated with help of experiments with test subjects.

Task complexity

Complexity can be considered not only as an intrinsic system feature but also in relation to a task to perform. Henneman and Rouse (1984) investigate **task complexity** with help of an experiment. Their approach is to analyse how some of the basic factors in the representation on CRT-screens (like the number of items per display page, number of levels in the system and component failure rate at the different levels) influence the complexity of a system in respect to how it is perceived and handled by the user. The system they tested was the simulation of the US nationwide telephone system.

The US phone system is designed as a five-level hierarchical network composed of more than 170 million telephones (as in 1984) and more than 22000 switching centres. The network consists of two basic elements, the actual transmission paths

inherent to a system and **behavioural complexity** includes the human component, i.e. it indicates how complexity is perceived by the human.

Nonbehavioral complexity

Under a nonbehavioral perspective three types of complexity can be distinguished: computational complexity, software complexity and descriptive complexity.

Computational, **combinatorial** or **algorithmic** complexity is oriented to the application of computers. These complexities are defined as the length of time or the memory allocation required to compute a certain function or algorithm on a particular type of machine. Closely related to the computational complexity is the **software complexity**, that can be expressed with variables like programming time, program structure and program length.

The **descriptive complexity** (complexity of physical systems) is related to the degrees of freedom or the number of independent variables that are necessary to describe a physical system. The type of description and the level of detail are here crucial. The complexity of an ideal gas is quite different when each molecule is described with its dynamic variables or the gas mass is considered as a whole in terms of the thermodynamical variables P, V, t, and taking statistical properties into consideration.

Physical systems can also exhibit a varying degree of **organization** that contributes to a reduction of complexity. A crystal contains an high number of molecules, but due to its organization, the structure can be described by a small number of parameters. This would not be the case if the single atoms were not organised and each one had to be described on its own.

In the view of Henneman and Rouse (op.cit.), the degree of **redundancy** and **internal connections** in a technical system can have two different effects on its complexity; both effects are of relevance in the practical operations. On the one hand, the internal connections may make the system more difficult to understand because they add new information. Internal connections would make the conceptual model of a system more complicated and thus possibly negatively influence tasks at the knowledge-based level, like the search for the cause of a failure. On the other hand, internal paths and redundancies may make a system

operators. A first approach could be to select operators on the base of their "thinking ahead" capabilities, maybe with help of experimental runs of the type that was described in Section 2.6. A different approach is to use the electronic medium to *enhance* (not to *replace*) the human capabilities where they are weakest, while not interfering with skills that are acceptable the way they are. The exposition in this chapter is based on the second assumption.

To investigate the role of computers in process control there are several red threads that can be followed. Our "red thread" is to start from the technical process as a complex system and consider the interface as a *tool* to reduce and cope with this complexity.

The advantage with this choice is that it does not have to refer explicitly to the operators' mental models of the process, *M(T)*. The user interface acts namely as the interface between the process *T* and the operators' mental models *M(T)*. Unfortunately, knowing so little about mental models, the risk to end up with unfounded results is very high. Moreover, different operators will have different mental models $M_1(T)$, $M_2(T)$,..., so how can one be sure to cover all of them? With the **complexity** approach we have the advantage that the operator can learn and use a particular interface, knowledge of the details of just *his* model are not essential. What matters is that the operator can perform his job properly. This approach is therefore more apt for the design and the evaluation of real interfaces.

3.2 The Concept of Complexity

In order to use complexity as main reference for the description of a system or of a task, we need to have a workable definition and, possibly, a metric for it.

A comprehensive comparison of different definitions and metrics for complexity was done by Henneman and Rouse (1986). Their classification of the different theories about complexity will be reported here with extensions and qualifications.

In general, and for the purpose of human-computer interaction, complexity can be divided in nonbehavioral and behavioural. **Nonbehavioral complexity** is

must not only control the technical processes alone, but rather the processes as they are interpreted by the designers of the automation systems. Process requirements and operators' skills are in general not weighed against each other in the design of the user interface; the emphasis is usually put instead on the computer control hardware. On the other hand, it can be said on behalf of system designers and automation engineers that they have initially acted with positive expectations about the role of automation and they applied what they learnt. The problem is then rather a cultural and educational one: the role of operators is seldom considered in engineering courses.

Ideally, monitoring, supervisory and control systems should *simplify* the work of the operators in control of technical processes. There is a direct relation between process complexity and difficulty in carrying out operations; a reduction in the complexity of a system or a task means that it will be easier to handle. Here a difficult balance must be reached. The user interface must help to reduce system complexity until it reaches a manageable level, below the intrinsic limits set by human cognitive capacity. On the other hand, an interface that is "too simple" is no good either. As Bainbridge (op.cit.) indicates, such an interface might fail to provide the operators the opportunity to train and keep up with the skill needed to handle unforeseen situations. In addition, with complex computer systems the operators must *monitor the monitoring system* and notice whether it is operating correctly. This adds complexity to their work.

A dynamic system is said to be complex to the extent that the human can observe it in different ways, at several abstraction levels, all of which are pertinent to the system operation. The major difficulty in the human control of a dynamic system is due to the fact that its output changes without explicit human intervention. This inevitably forces the human controller or supervisor to "keep ahead" of the system in order to successfully complete any realistic task. The requirement for keeping ahead of the system leads to anticipatory, as opposed to purely reactive, behaviour. Anticipatory behaviour, in turn, implies the ability of the human to predict future system output of the basis of present system state and present and future input. But we have seen that there is experimental evidence for the difficulties that people have to act anticipatorily (Section 2.6).

In the control of complex processes where particular cognitive skills are requested two ways can be followed on the definition of the role of process

3 The Digital Computer as a Tool to Deal with Complexity

This chapter presents monitoring and control systems as a *tool* to reduce and deal with the intrinsic complexity of technical processes. Complexity is here considered in relation to the data exchange between the human user and the process through the user interface. Section 3.1 introduces process control under the aspect of a complexity problem. The concept of complexity for a generic system is treated in more detail in Section 3.2. Section 3.3 is dedicated to how to cope with complexity by structuring and Section 3.4 to how computers can be used as interfaces to reduce complexity.

Section 3.5 deals with the issue of task analysis as a basis for the development of user interfaces. In Section 3.6 are presented three case studies about the mismatch in the interaction user-machine and how this leads to an increase of general complexity (or to missing the intended goal). Section 3.7 indicates how user interfaces and monitoring and control systems can be developed and used in order to act as complexity-reducing tools between technical processes and human users.

3.1 Process Control as a Complexity Problem

The role of process operators has greatly changed over the last years. Due to the widespread use of automation and process supervision equipment, operators have gradually moved away from the processes they run and into control rooms, from where they monitor and supervise the automated operations of the processes.

This effect is particularly important in supervisory systems (those that combine process monitoring with automated functions). These systems have not necessarily brought a simplification in the work of process operators. Instead, as e.g. Bainbridge (1982) points out, the concept of automated support has backfired, because "designers who try to eliminate operators still leave operators to do the tasks which the designer cannot think how to automate". Operators