4.7 Conclusions

In the design of user interfaces, three elements have to be considered that ultimately dictate how a solution will look like:

- the user
- the process
- the way work is (should be) accomplished

The user, his cognitive capabilities, the process, its structure, the work task and how unexpected situations should be handled are all elements that must be considered in the design of an user interface. There are however few ways to model them. Formal methods and engineering equations can of course be used in the description of the technical process; for the other components the analysis will in most cases be formulated in plain language on paper. The analysis fulfils two purposes: it provides a basis to develop and evaluate the design and it gives the designer of the user interface the opportunity to get insight into the problem and its possible solutions.

It is very important that all components of the system user - interface - process - goal match each other in terms of type and quantity of information transferred, speed, its comprehensibility, etc. Many existing systems are not easily accepted or led to problems in their operations because of a mismatch somewhere in the chain between user and goal.

In this Chapter we have examined several issues that can be considered in the definition of the "external" interface, with special attention to the presentation of information, its coding, and the aspect of commands. Three important general principles that can guide the development of a design are simplicity, visibility, and consistency. In addition, in the design of user interfaces must also be considered "indirect" aspects like technical documentation, necessity for training, support, openness for the users' points of view in the revision or the upgrade of a design, and many more. Ultimately, it is the combination of all these factors that determines the success - and usability - of an interface.

of log files is not too expensive and is quite exact. Notes taken by an external observer are the cheapest method and the one easiest to evaluate, but they are also most prone to errors.

Shneiderman (1987) indicates the following objective criteria for the evaluation of a computer system and of its user interface:

- execution speed. How long does it take to perform specific tasks ?
- rate of errors by users. How many and what kinds of errors are made in carrying out the benchmark set of tasks?
- subjective satisfaction. How much did users like using the system?
- time to learn. How long does it take for the typical user to learn the most important commands?
- retention over time. How well do users maintain their knowledge after an hour, a day or a week?

Of these goals, speed of performance, rate of errors and satisfaction can be evaluated after a comparatively short time and therefore make a good choice of testing benchmarks. The time to learn and retention over time require by their nature a longer time for evaluation. Therefore, they cannot realistically be included among the parameters for direct consideration. They could, however, play an important role in the periodic re-evaluation of products that reach a larger market and are not limited to one target user group only.

It sounds like a paradox, but the best user interfaces are also the easiest to forget. The reason is that if they appear to be natural, the user will not have to make any special effort to memorise them.

So far only few companies have understood the importance of user-conducted software testing before products are released. The prevailing attitude is to let the market do the testing. Many companies are not interested in user satisfaction but rather in getting quick money for their software. What the user does with this software, is not the original company's matter. It is however questionable whether this "strategy" really pays in the longer run.

interact much more, for example in periodical meetings, while in static design designers and users are not required - if not altogether discouraged - to meet.

The advantage of the dynamic design is that the final users have a much higher degree of participation and thus tend to be much more satisfied. Their needs are better approached, and they know what they are going to get. The disadvantages of this method are the longer development time and the higher costs. The outcome of an iterative design process cannot be known in advance, which represents a risk for a company. But in a certain sense, it can be said that the test element in the dynamic design is an abbreviated and more direct form for the tests that later would take place anyhow with the final users. The higher investment supported by the companies for this initial testing can be offset by better user acceptance, lower failure rates and thus less after-purchase complaints.

The dynamic design method can also be considered under a different perspective. The concept of complexity was introduced in Section 3.2, where it was also reported that - lacking a precise metric - the only practical method to estimate it is via user tests. In the dynamic design method the users are involved and rate the system under development by unconsciously following criteria like complexity, task matching, ease of information transfer from other contexts, etc. Testing software acceptance directly with the users combines in a practical way the analysis of different aspects that are difficult to carry out explicitly, and therefore delivers the type of results that are needed in practical design.

Due to the fact that the evaluation of user software is a process that potentially has to be repeated several times, it is important that the evaluation method gives practicable results and is easy and inexpensive to carry out. Among the methods to evaluate software performance and satisfaction by the users are:

- Questionnaire
- Analysis of log-files
- Protocols made by an independent observer
- Video collection of material
- Direct questioning

According to Müller-Holz-auf-der-Heide (1991), the video-recorder gives the best and most exact results, but is also the most expensive method. An offline analysis

In static design the project specifications and a test plan are written on the base of the initial requirements. The system software is then developed and modified until it stands all tests. In this method there is only one feedback loop to check whether the software is in agreement with the detailed design specifications.

Static design does not show whether the requirements themselves were correct or failed to address the task and the goal. The only feedback from the final user reaches the developers usually along a chain consisting in several links and that typically includes marketing people and the field salespeople responsible for customer contact. Changes can be incorporated in the product only with major delays and only if the company decides that they are of interest for several customers. In case of specific projects developed for one customer only, the customer usually binds himself to accept the implementation of the initial requirements while these are still on paper. Any later change would cost him money. Unfortunately, user satisfaction is never included in the requirements.

An alternate design method is the **dynamic design**, described among others in Müller-Holz-auf-der-Heide (1991). The dynamic design includes evaluation and testing phases as part of the design process itself (Figure 4.7).

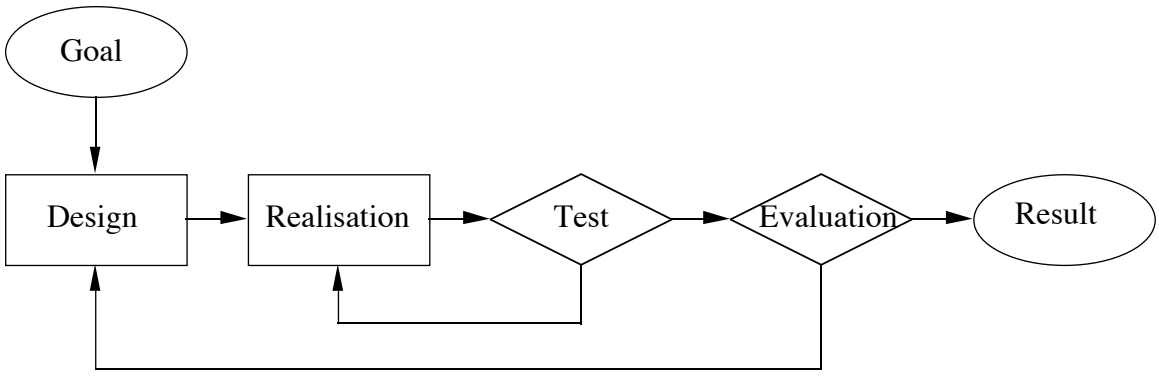


Figure 4.7 The dynamic design method

With this approach, the initial requirements take the form of "set-point values" rather than direct "control signals". Designers, programmers and final users

Ideally, the number of items in a menu is not too large. With too many items on the screen, one may have already forgotten the first ones by the time he is through scanning the list. If a system has a very large number of action paths that can be followed, a tradeoff must be made between the approximate number of choices in every menu and the number of levels in the menu hierarchy.

Similar functions in different menus should be connected to the same keys (*consistency*). A <BREAK> command should always be present, possibly with its own dedicated key. It should be possible at any moment to interrupt the current work and go back to the next higher menu level or even to jump to the highest level (root) menu. It should not be required to go through a series of screens, or even reply to additional questions, just to leave the current menu.

4.6 Prototyping and Evaluation

The development of computer systems and of their user interfaces can basically follow two principles: the static and the dynamic (interactive) design.

The **static design** (Figure 4.6) is the most common software development method. This method is favoured by software development companies because each work step is well defined and can be accounted and paid for as soon as it is completed. The work division helps also divide responsibility among different people and groups, so that nobody is ever responsible for possible dissatisfaction when the work is completed.

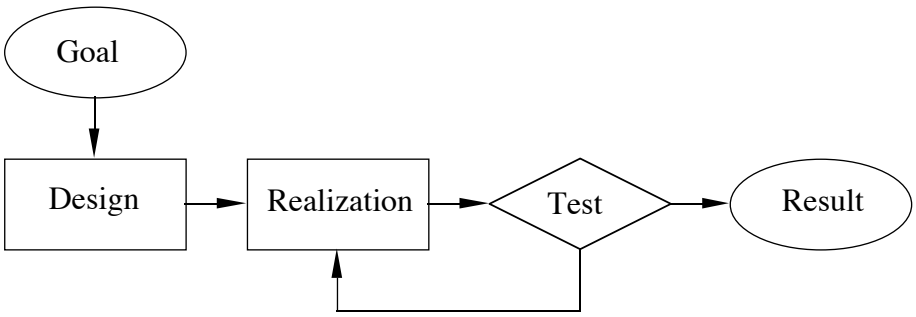


Figure 4.6 The static design method

use. In Section 3.7 was indicated how a process simulation routine would enhance a control system. In such a case, "dangerous" commands could be accepted only after having been run in simulation; only after that the control system would pass them further to the technical process. This strategy may however not be applicable for systems where immediate process manipulation is required.

It is important to have the possibility to stop a computer-controlled actuator or machine immediately in case of an emergency. In such a situation, nobody would have the patience to type in an ordered command sequence on a keyboard. A clearly marked emergency OFF button should be installed within easy reach for the operator. It is common practice to paint the button red on yellow background. "OFF" buttons are usually large enough to be operated while wearing gloves and without need for careful aiming.

It is advisable that help information is available on line. Help should always be called from the same key, which should be distinct and clearly marked. Modern systems offer context-related help, that is, they recognize the current situation (the data or program which is currently active) and offer help related to that context.

4.5 Menus

The principles of visibility and consistency required in screen and command design must of course also be followed in the design of menus. In this respect the following considerations can be made.

To begin with, the menu structure should become quickly clear to the user. Each menu should be identified by a title/headline, possibly using the same text indicated as menu selection item one level above.

The items in a menu should be at the same abstraction level: Functions like "delete character", "print file" and "boot computer system" do not belong to the same menu. The menu items should not be placed randomly, but follow some semantic principle. In case no such principle is evident, alphabetic order will serve as a possible ordering criterion.

each other. In the VMS operating system for VAX computers this method is used for all commands and parameters, where the first four letters are sufficient to uniquely identify any command (commands may also be typed in their entire length).

In fields where an alphanumeric input is requested, only a few combinations usually make sense. "FGS" and "OID" are letter combinations as "ON" and "OFF" are, but they won't be understood by a binary actuator. The possible alternatives to avoid nonsense input data are: (1) to display the correct values as part of the background information; (2) to provide a window menu selection of the possible values; (3) to display a message if the input is not understood by the system.

The alternative (1) is not feasible when the number of possible commands is large; it leads easily to cluttering the screen with too much static information. Alternative (3) may cause delays, depending on how often mistakes are made. The solution (2) may be the optimal choice; this solution is also directly supported under window-based interfaces. A new value can be selected either by typing in it explicitly, or at least some of its characters, or by pointing at it with help of the arrow keys or the mouse. The selection is then confirmed by an <ENTER> command or a click on the mouse. One of the command alternatives - the current, previous, most common or safest one - could also be shown as default selection for a command. The displayed value is then accepted by the system as valid unless explicitly changed by the user.

Typing a command from the keyboard requires some thinking and can lead to errors. It is good to ask for confirmation before execution of sensitive commands, e.g. with a question of the kind "*Do you really want to boot the system [YES/NO]*?" Here might however arise a problem, because once an action is learned, it is carried out automatically at the skill-based (sensomotoric) level and without further thinking. The question alone is no guarantee for the exact intentions of the user. Different strategies might have to be defined.

In some systems potentially dangerous commands are made deliberately difficult to carry out. This is not a good approach. Special commands (that probably have to be used only in particular emergencies) can instead be protected with special passwords. A good control system should be at the same time *safe* and *easy* to

A command defines a reference value for a state; the actual value will later become equal to the reference value only if control system, actuators, sensors and physical process all work correctly. It takes some time before a command reaches the actuators, is executed, and the verification is reported back to the user. Actual and reference values must be presented in a way clearly identifying them so that they cannot be confused with each other.

If a system cannot give immediate response to a command, as verification that the command was accepted and is currently being processed, some kind of partial response should be displayed. This could e.g. be a message of acknowledgement or a different highlight of the input data on the screen. The effect of a command should be immediately evident and there should always be a possibility to reverse it.

For processes with longer time constants, the first reaction could be a message from the process computer of the kind: "*New temperature set-point value is 66˚C. Estimated time to reach set-point is 18 minutes [at 14:28].*" This requires the availability of a simulation routine, as described in Section 3.7.

A hierarchical structure is also possible for commands, in a way similar to the hierarchical structuring of the management levels in a plant (Section 3.3). At the lowest level are the input commands for the direct control of the actuators, while at higher levels complete sequences for the control of complex procedures can be started. These sequences have the aspect of "batch" or "command" files.

Structuring of commands can also take place on the basis of task analysis (Section 3.5). In general, commands should be presented in the same context (or in proximity) of the information that is needed to select the commands themselves.

Consistency is not only important in the process representation, but plays an important role also in command input. The commands should be context-independent and always act in a similar way, irrespective of past activities and system history.

String commands to be typed explicitly on a keyboard should be as short as possible, yet not lose their meaning. A good method is to use the first letters of the command name, provided that different abbreviations are not confused with

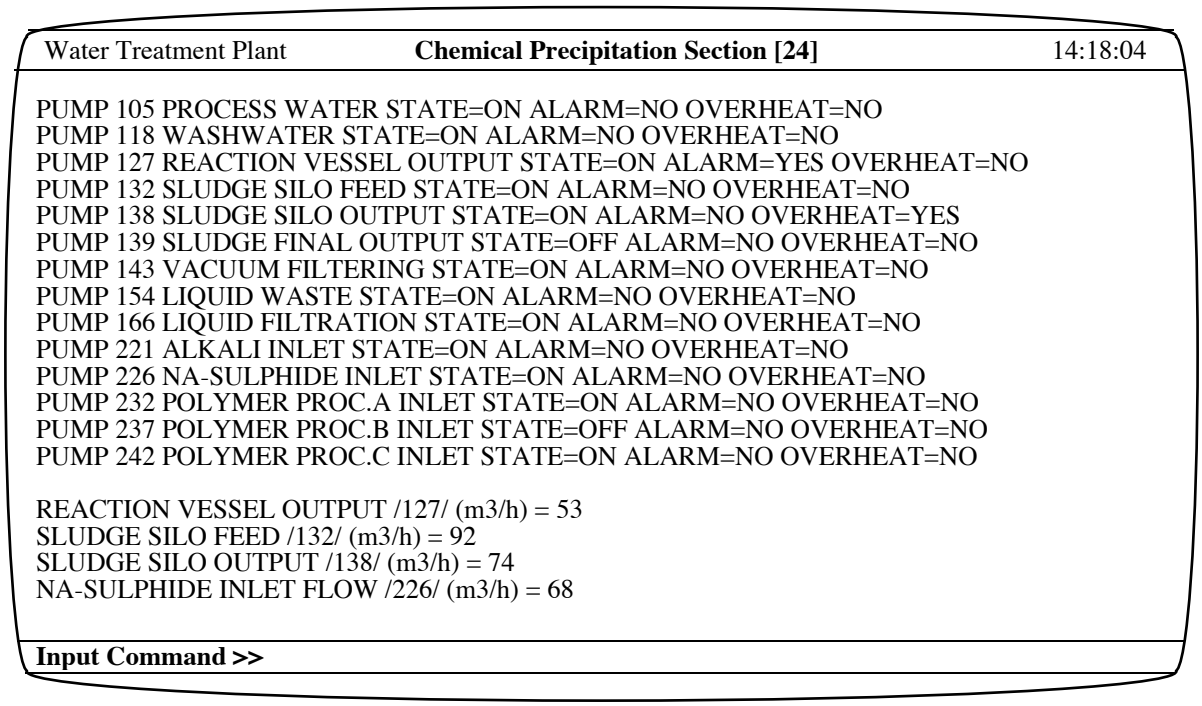
instead implicitly assumed that all information presented on the screen page is equally important. However, a major problem with all these metrics is that they focus only on the symbolic aspect of the display, without considering the *semantic* information to convey. In other words, they do not address the issue of the complexity in relation to what information the user really gets from the display. As an analogy, two different sentences are seldom equally comprehensible, even if they are of the same length and are printed with similar type fonts and point sizes.

In conclusion, no metric for display complexity that can be applied for a large class of problems has been proposed so far. If a detailed task description is available, then Sears' measure of layout appropriateness can be used.

4.4 Operator Commands

The interaction between humans and computers does not only take place from the machine to the user but also from the user to the machine, when control tasks have to be carried out. The user enters data by typing in command sequences on a keyboard, by pushing buttons on a panel or by manipulating a device like a mouse or a joystick. Some considerations follow here regarding the design of communication from human to machine, with special regard to controls typed in via a keyboard.

It is very important for the user to get an immediate "feeling" that a command has been received and accepted, even if the related processing does not start immediately. The feeling starts right from the acoustical "click" when a key is pressed or by seeing a pointer moving on the screen under control of the mouse. Another example of direct feedback are the tones in a touch-tone dialling phone. The feedback is intuitive: nobody recognises the dialled number from the tones, but we get enough information to tell that all the numbers were dialled and whether a finger slip let a number be dialled twice. If nothing seems to happen after having pressed the <RETURN> or <ENTER> key because of long response times, one may wonder whether the system operates correctly.



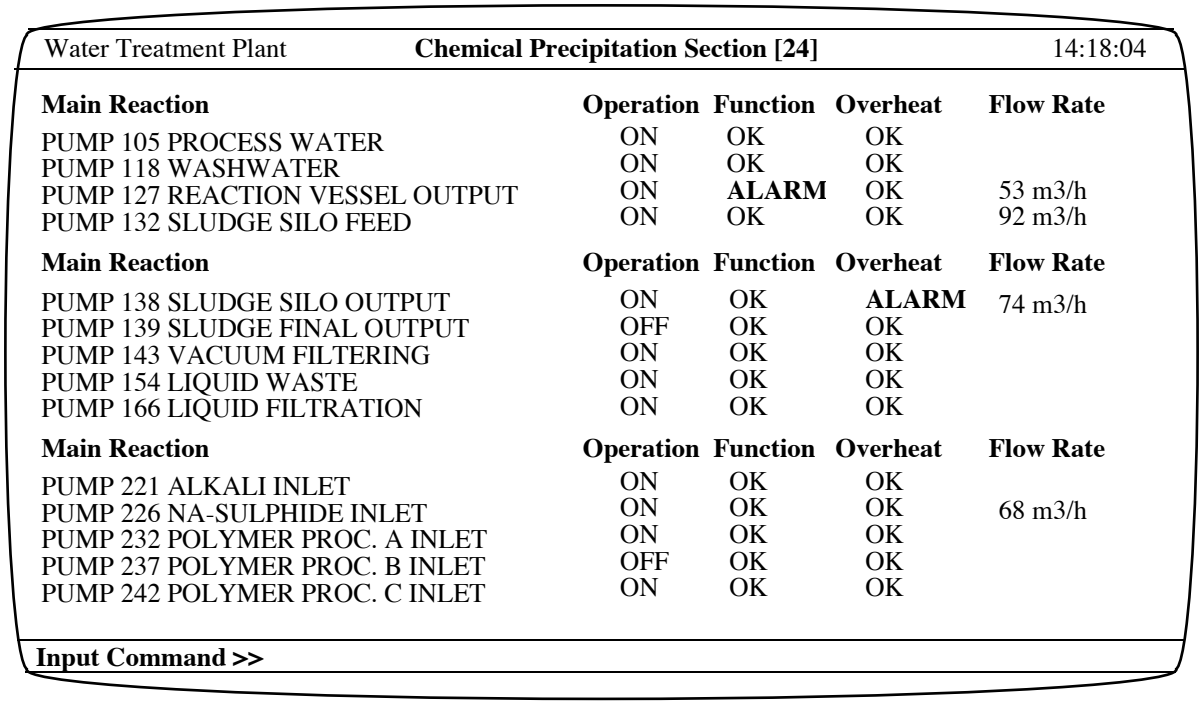
Water Treatment Plant **Chemical Precipitation Section [24]** 14:18:04

PUMP 105 PROCESS WATER STATE=ON ALARM=NO OVERHEAT=NO
PUMP 118 WASHWATER STATE=ON ALARM=NO OVERHEAT=NO
PUMP 127 REACTION VESSEL OUTPUT STATE=ON ALARM=YES OVERHEAT=NO
PUMP 132 SLUDGE SILO FEED STATE=ON ALARM=NO OVERHEAT=NO
PUMP 138 SLUDGE SILO OUTPUT STATE=ON ALARM=NO OVERHEAT=YES
PUMP 139 SLUDGE FINAL OUTPUT STATE=OFF ALARM=NO OVERHEAT=NO
PUMP 143 VACUUM FILTERING STATE=ON ALARM=NO OVERHEAT=NO
PUMP 154 LIQUID WASTE STATE=ON ALARM=NO OVERHEAT=NO
PUMP 166 LIQUID FILTRATION STATE=ON ALARM=NO OVERHEAT=NO
PUMP 221 ALKALI INLET STATE=ON ALARM=NO OVERHEAT=NO
PUMP 226 NA-SULPHIDE INLET STATE=ON ALARM=NO OVERHEAT=NO
PUMP 232 POLYMER PROC.A INLET STATE=ON ALARM=NO OVERHEAT=NO
PUMP 237 POLYMER PROC.B INLET STATE=OFF ALARM=NO OVERHEAT=NO
PUMP 242 POLYMER PROC.C INLET STATE=ON ALARM=NO OVERHEAT=NO

REACTION VESSEL OUTPUT /127/ (m3/h) = 53
SLUDGE SILO FEED /132/ (m3/h) = 92
SLUDGE SILO OUTPUT /138/ (m3/h) = 74
NA-SULPHIDE INLET FLOW /226/ (m3/h) = 68

Input Command >>

Figure 4.4 Example of a poorly structured screen display



Water Treatment Plant **Chemical Precipitation Section [24]** 14:18:04

Main Reaction	**Operation**	**Function**	**Overheat**	**Flow Rate**
PUMP 105 PROCESS WATER	ON	OK	OK	
PUMP 118 WASHWATER	ON	OK	OK	
PUMP 127 REACTION VESSEL OUTPUT	ON	**ALARM**	OK	53 m3/h
PUMP 132 SLUDGE SILO FEED	ON	OK	OK	92 m3/h
Main Reaction	**Operation**	**Function**	**Overheat**	**Flow Rate**
PUMP 138 SLUDGE SILO OUTPUT	ON	OK	**ALARM**	74 m3/h
PUMP 139 SLUDGE FINAL OUTPUT	OFF	OK	OK	
PUMP 143 VACUUM FILTERING	ON	OK	OK	
PUMP 154 LIQUID WASTE	ON	OK	OK	
PUMP 166 LIQUID FILTRATION	ON	OK	OK	
Main Reaction	**Operation**	**Function**	**Overheat**	**Flow Rate**
PUMP 221 ALKALI INLET	ON	OK	OK	
PUMP 226 NA-SULPHIDE INLET	ON	OK	OK	68 m3/h
PUMP 232 POLYMER PROC. A INLET	ON	OK	OK	
PUMP 237 POLYMER PROC. B INLET	OFF	OK	OK	
PUMP 242 POLYMER PROC. C INLET	ON	OK	OK	

Input Command >>

Figure 4.5 The same information as in Figure 4.4, in a structured display

$$C = -N \cdot \sum_{n=1}^{m} p_n \cdot \log_2(p_n)$$

where:

C = layout complexity , expressed in bits

N = number of events (i.e., widths or heights)

m = number of event classes (i.e., number of unique widths or heigths)

p_n = probability of occurrence of the nth event class (based on the frequency of events within that class)

It might not be clear what is meant by number of events and number of event classes, and how to recognize them unequivocally on the screen. The Bonsiepe/Tullis metric applied to the screens of Figure 4.4 (N=64, m=18, p_i=1/18) and Figure 4.5 (N=79, m=5, p_i=1/5) gives 266 and 183 bits respectively. Structuring the data has reduced the number of unique widths and heights, which contributed to a reduction of the equivalent complexity content of the second screen.

The proposed metric is not exempt from flaws. Its major drawbacks are that it does not cover graphic displays, the use of colour and other current techniques. Sears (1992) proposes a different method for layout evaluation, called **layout appropriateness**. This metric incorporates simple task descriptions that can assist designers in organising widgets (small items) in the user interface. The layout appropriateness metric requires a description of the sequences of widget-level actions users perform and how frequently each sequence is used. The appropriateness of a given layout is then computed by weighing the cost of each sequence of actions by how frequently the sequence is performed.

In other words, Sears' method makes use of a detailed task analysis (or task description) and a set of widgets to organise, so that an optimal layout in the sense of practical use (layout appropriateness) can be defined. Existing or proposed designs may be compared and evaluated with the optimal layout design for a given task.

The inclusion of the task and thus the consideration of what information is most relevant is an important aspect in the approach by Sears; in Tullis' analysis it is

A different kind of representation for complex processes was proposed by Lind (1990) and is known as **Multilevel Flow Modeling** (**MFM**). In this representation are primarily shown the material, energy and information flows of a plant. In MFM every system or part of system consists of a source, a sink and transport components for each of the flows; there are connections between the different flow functions. Because of the physical equilibrium laws for materials and energy, the flows must remain constant through the process or may change only in a predetermined, known way (like e.g. when fuel is burned to produce heat). An interruption of one of the flows is an indication of a possible disturbance in the process; further, the disturbance is conducted to the point where the equilibrium state is no longer verified. MFM is a very new type of representation and there is so far no indication about its acceptance in real process control applications.

4.3.4 Representation complexity

The principles of simplicity, visibility and consistency applied to the design of screen pages are probably the most successful method for the reduction of the complexity of a presentation. All changes in a design that improve one of these aspects without negatively influencing the others should be implemented.

Several methods have been proposed to measure the **complexity** of a representation (this is not the same as the system complexity treated in Section 3.2), but so far there is no widely accepted metric. It is therefore not possible to prepare alternative designs and test them *a priori* to select the less complex, given a certain - fixed - amount of displayed information. The evaluation of the preference of a screen layout (where complexity would be one of the implicit factors that are tested) has still to be carried out in practice by conducting series of tests with people.

Tullis (1983, 1988) compare different metrics for the complexity content of screen displays. One such metric is a relative measure of how many characters are used up on the screen (i.e. measure of the fill factor).

A different metric is derived from the Shannon definition of information; it was first developed by G.A. Bonsiepe and is also described in Tullis (1983):

red always means "alarm", when he perceives the colour his reaction can be immediate. If a thinking effort is needed, like to weigh the stimulus "red" with information about the current display screen and what red means in that specific context, the effort will be greater and the reaction slower.

In other words, consistency means that coding is not context-dependent (i.e. when a coding feature has different meaning on different screen pages). In addition, coding must be natural, be processed as far as possible unconsciously and not require explicit interpretation. Coding "cold" by red and "hot" by blue is technically easy, but would require an effort to be understood because it is unnatural.

There might be conflicts in the right choice of symbols. If a process is in alarm state because its temperature is too low, should it be represented in red or in blue? It depends on what has highest priority, whether to give an immediate feeling about the temperature or an indication of the functional state, where the natural colour for alarm is red. In any case, consistency should hold through all representations. It does not matter what a code represents, as long as it always represents the same thing.

4.3.3 Screen representation and layouting

For the definition of a graphical display page, the picture can be designed according to different principles. In the **physical** or **technical** (**spatial**) representation a plan of the plant or subunit is designed with its specific symbols (most technical symbols are standardised). In the layout development one could strictly follow the available technical drawings (these could be very complex and difficult to understand) or represent the evolution of the physical process linearly, without consideration for the actual spatial device placement. In the latter case, a straight disposition from left to right is usually preferred.

In the **task-oriented representation** is shown the information that is necessary to carry out a specific task. This representation is more oriented to operations, while the technical representation is more apt to support conceptual thinking, e.g. to identify the source of a problem.

For instance, if a switch is used to connect alternatively two devices (or a production line branches into two cells), the switch position itself can be shown, or also which device (or cell) is connected, identifying it with a different feature like an empty or framed symbol (Figure 4.3). The symbolic representation does not require an explicit interpretation of the picture.

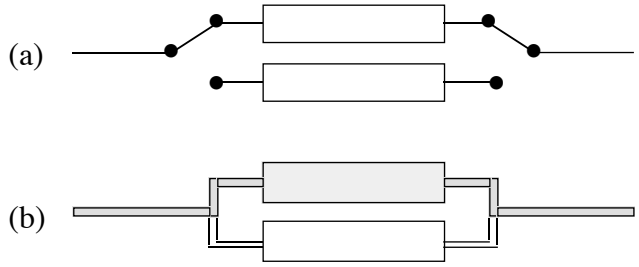


Figure 4.3 (a) Symbolic representation; (b) functional representation

Fisher and Tan (1989) report about an "highlighting paradox" in visual displays. The problem they address is whether highlighting important items on a display screen makes them easier or not to recognize. The result is that ease of recognition depends on the highlighting mode. Blinking and reverse video lead to delays, but not colour. The reaction times depend also on user practice. Highlighting can be of benefit to novice users, but as subjects become more practised, they generally know the location and the identity of the option they are searching and highlighting becomes therefore more like a hinder. Finally, a text should never be let blink, because this makes its reading more difficult. It is sufficient to let a small symbol blink near it.

Consistency means that a coding clue (colour, shape, etc.) maintains its meaning across the whole system. If red indicates an "alarm" state in a screen picture, it should not indicate "hot" in a different picture or "inflow" in a third one.

It might be argued that with training the operators will be able to tell when red means "alarm" and when it means "inflow". This is true, but then an important advantage of cognitive knowledge is missed. If we consider the action model (Section 2.1), we see that the most efficient reactions are those at skill-based (sensomotoric) level and at rule-based level. If an operator learns by training that

changing different properties in parallel and thereby conveying more information at the same time. Common visual codes for items shown on a terminal screen are colour, shape, dimension.

The effects of ease of perception for different types of coding have been object of several experiments. Jubis (1990) tested perception facility for symbols that were coded in different ways: shape, colour and shape together (redundant colour coding), and colour alone. According to this study, coding by colour and shape combined and by colour alone leads to faster reaction times in the human observers than shape alone; colours result then to be the most important coding attribute.

Colours can provide clues when they display functional states. Green is generally perceived as indication of security, permission or correctness (it may for instance indicate that an unit is in proper working order). Red is related to states of alarm, danger and prohibition. Yellow is understood as a warning and can indicate the presence of some minor problem.

Simplicity suggests that the number of colours, shapes, highlights should be kept to a minimum. Moreover, a few different colours or shapes can be recognised alone, but from a few values, these make sense only in comparison with others. Colours should be used sparingly; 4-5 colours are understood with no major effort; there are indications that 7 different colours is an absolute maximum to never exceed. The limits on the different colours or clues that can be identified simultaneously are related to the capacity of short-term memory (Section 2.2).

It is important not to rely only on colours as a means of showing important information. A large number of people are blind to some colours and are therefore incapable of recognising them (Section 2.2). Environmental factors like illumination and shadows may make difficult the perception of some colours on a terminal screen. The information to be shown should therefore present some kind of redundancy, for instance with help of labels, texts or other graphical symbols, in order to ensure that the conveyed meaning is understood.

A symbol may be displayed filled, empty or rastered. Alone, it is possible to tell with certainty only 3-4 different raster patterns, others have to be observed in relation to a reference frame.

In general, a good layout organization has the following characteristics:

- it is adequate for the purpose; it does not present more or less information than necessary (*simplicity*);
- it is, as far as possible, self-explaining (*visibility*);
- it is consistent at more levels. The same coding (in symbols and colours) carries the same meaning on different screen pages and the user knows what to expect in different situations (*consistency*).

The purpose of a display representation can be to induce the user to perform a certain action, by providing informative support to a work task. In this case, the information on whose base the user must act should be highlighted on the screen. If instructions for the control of a machine are given, these must be clearly represented. If several alternatives depend on the displayed data, the required action must be represented in a direct way. The style "*If the temperature is over 200 °C, then the action x should be performed*" is not straightforward. On a screen page, clarity and conciseness are imperative: "*Temperature = 226°C. Perform action x.*".

The memory of the user should not be overloaded, especially in consideration that one of the problems the computer *doesn't* have is to effectively store and recall information. The user should not have to remember information from a display page to use it then in another screen page; we have seen in Section 2.2 that there are definite and restricted limits to the amount of information that can be retained in short-term (working) memory. The completion of any given task should be completed in few steps and with as few commands as possible.

4.3.2 Coding

The human-computer interface must be able to draw the user's attention to important facts and to allow prompt and correct reaction on the basis of the given information. In this task, coding plays a crucial role. Coding is also important for chunking (Section 2.2), to reduce a large amount of information to a few chunks that are directly manageable in working memory.

Coding is the change of some property of a communication channel. A **code** relates the type and amount of change of the channel property with the information that has to be carried. Several codes may act concurrently, i.e.

on the same screen. The screen dedicated to a work cell will present only few basic data for each device, like whether it is operating correctly and the current production or processing rate. At an higher abstraction level, the screen layout for a production line will present basic data for all the cells, whose detailed states do not have to be displayed. Although the main concept (the plant or production line) is complex, the functional idea on each screen can remain simple: does the machine, cell or plant operate correctly or not ?

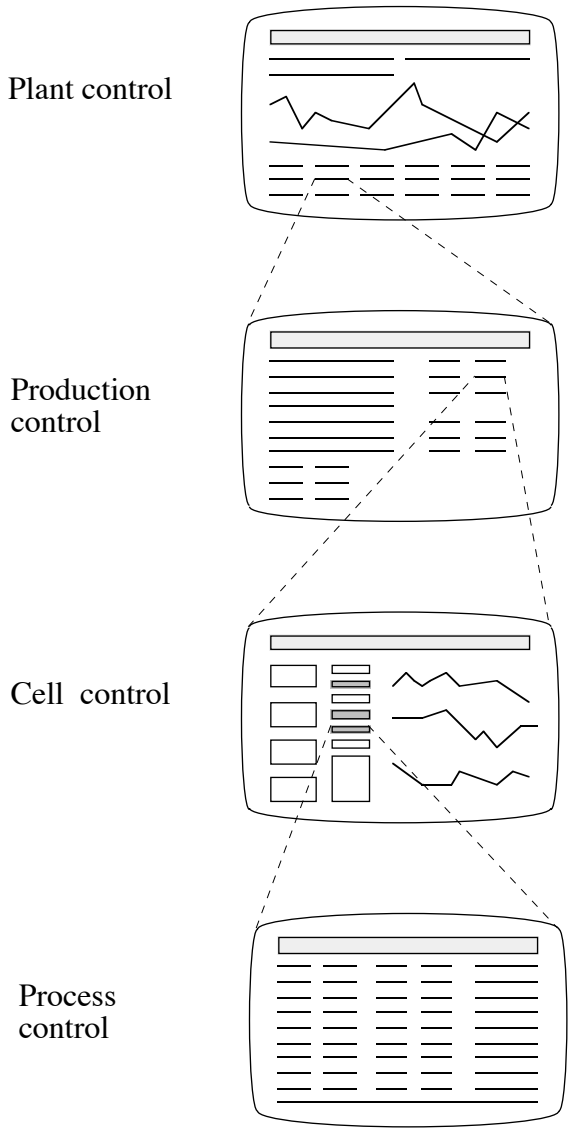


Figure 4.2 Process representation organized according to the plant management levels (compare with Table 3.1)

The information showed on a display may be considered from a direct point of view (to convey information) of from a pragmatic point of view (to lead the operator to perform the intentions of the system designer). The purpose of process control is (1) to direct the process according to a defined production schedule and (2) to recognize and identify alarm states and take appropriate measures to guarantee the safety of a process. As indicated in Section 3.5, the information on the display must then support different types of task:

- normal operations: start-up, shut-down, normal process control, parameter identification, process optimisation
- identification of alarm states
- indication when appropriate action has taken place in abnormal situations, otherwise support in the search for a correction strategy.

The information content should be adapted to the user: the expert is not interested to "simple" information that instead can be very important for a beginner. Yet novices share often the same equipment with experts, so that compromises must be made. In addition, the typical computer user tends to give importance to everything that appears on a screen. Therefore screen displays should be simple and not contain useless information, otherwise there is a risk that unimportant items draw unwarranted attention.

The problem of the screen representation consists on the one hand of *what* should be shown and on the other hand, *how* it should be represented. The first question can be answered with an analysis of the structure of the technical plant and the goal of the representation (i.e. information transfer or completion of a work task). If the data have general information character, then possibly even a simple tabular representation might fulfil the goal. In this case it should only be ensured that consistent units, lead texts, state representations, etc. are used.

The simplest method for structuring the data from a complex plant follows the hierarchical structuring of a plant as described in Section 3.3 (Figure 4.2). On each display page should be shown only one basic **concept** in the simplest possible way. Emphasis should then be given to the most important information concerning a specific object or work task. For example, in a plant overview, the functionality of the plant has the highest importance. All parameters related to the same device or to different devices, but in the same task context, will be shown

A clearer distinction between static and dynamic information is obtained on screens when the lead texts are written with normal intensity and the dynamic variables in high intensity. If the dynamic variables are also used as input, the variable currently selected may for example be shown in reverse video (that is, where foreground and background colours are exchanged). Dynamic variables showing parameters in alarm state can be highlighted using a different colour or reverse video. The contrast between normal and highlighted representation should however not be too high, otherwise readability might be impaired.

The use of words with negative connotation ("ALARM", "WARNING") should be avoided in lead texts unless they are clearly unrelated to the actual state of the controlled system. The texts should motivate and not unnecessarily alarm or irritate the user.

Quite often, it is important to abbreviate some words and expressions. Different principles for abbreviation have been described, like e.g. in Shneiderman (1987). The principles of simplicity, visibility, and most of all consistency should be used in the choice of abbreviations.

Concerning the selection of fixed, pre-programmed messages, a "nice" machine does not blame faults on the user ("*wrong input*!") but on itself ("*The command cannot be carried out as indicated. Input these additional data.*"). The second message indicates also what kind of measure is required and provides therefore practical support. Concerning style and pregnancy, the operational instructions for fire extinguishers are a good reference also for computer texts.

4.3 Screen Layout Design

4.3.1 Process representation

The representation of process-related information on computer screen displays is one of the most important aspects of the human-computer interface for monitoring and control applications. This subject is also one of the best treated in the literature on user interfaces.

In most cases, the process or system development engineer is responsible for the selection of appropriate text dialogues that are then used by other people, the process operators. The operators will have to take decisions and perform actions on the base of the presented information. Computer language should then not be a poor copy of natural language, but be defined on its own to achieve the intended communication purpose. A few indications about this use of language will follow here.

A short sentence used to indicate only a limited number of states can be divided in a fixed text (**lead text**) to indicate the type of selection together with a variable for the actual state (**dynamic variable**). The lead text alone shall not give complete information in grammatical/ syntactical sense; it should rather create a small "tension" to be released only in conjunction with the dynamic information. In this case it is avoided that the lead text alone is misunderstood as to be itself the actual state information. Only the combination with dynamic information should make complete sense. In most cases it is sufficient to use a different form for a noun or verb, like

instead of: **device A11 powered: YES/NO**

use: **device A11 power: ON/OFF**

In general, lead texts that require an answer like YES or NO should be avoided. I have once seen a system with messages like "NOT ALARM STATE=YES" (in plain language, "everything's OK"). This kind of output might be correct in Boolean sense, but is not easily understood by humans and only brings confusion.

The lead text should not be too generic and should contain hints on what the dynamical part is going to be. Compare the following examples:

device A12 status: **ON/OFF ?**
OK/ALARM ?
ACTIVE/STANDBY ?

write instead:

device A12 power: **ON/OFF**
device A12 operation: **OK/ALARM**
device A12 connection: **ACTIVE/STANDBY**

overcome such a situation, two methods can be used. One is that each consistency aspect is related to one visual clue only (size, colour, form, etc.), so that different conditions may be indicated at the same time (e.g. via size and colour). A second solution is to establish a hierarchy, where a type of information overrides others.

An important support for consistency is offered by the use of standardised interfaces. The initial anarchy of many different products has evolved to a situation where specific user interfaces (Microsoft Windows, X-Windows) dictate the basic operational mode of the interface. Some of the most important guideline documents and standards for the design of user interfaces are reported in Appendix A1.

4.2 Use of Language in the User Interface

Information exchange with computers takes place in different forms: by setting switches and reading lamp indicators, by entering analogue values via a continuous manipulator (mouse or joystick) and by looking at symbols and data on a screen. The most common way to transfer information is via natural-language messages displayed on a screen or typed via a keyboard. In this interaction the language is used differently than for normal people-to-people communication.

Typical computer outputs are either sampled information about analog values ("TEMPERATURE=66.2˚C"), clear text messages with a few possible alternatives ("DEVICE CONNECTED/DISCONNECTED") or prerecorded messages ("The computer will be shut down in 5 min"). The language used for human-computer interaction lacks most of the features that belong to natural languages, like unclear expressions, redundancies, use of rhetorical figures etc. The computer use of language is fully predictable ("TEMPERATURE=27.9˚C"), contrary to natural language ("it must be almost thirty degrees, I'll have a cold beer"). A person telling the day temperature reporting decimals in colloquial speech (and without intending to be ironic) would draw almost as much attention as a computer asking for a beer.

contributes to confusion. General consistency with known and trained rules and established standards is therefore more important than the use of metaphors.

Consistency

Consistency means that the same representation is used for similar or analogue components in a system. This means that to apply consistency in the description or visualisation of a system, it is first necessary to create a structure for it.

Consistency can also be considered as **visibility by analogy**. Where visibility is necessary to understand a concept the first time, consistency helps transfer existing knowledge to new contexts. We have seen that there are experimental indications that the transfer of operational (syntactic) knowledge is much easier than the transfer of conceptual (semantic) knowledge (Section 2.7). Ideally, consistency should hold at the operational as well as at the conceptual level, which is not an easy task to accomplish.

Consistency can also be considered the other way around. Different clues (colour, style, size, etc.) must reflect differences in the real world. In this case, consistency goes along with simplicity and visibility. A picture should not show more clues than necessary (simplicity) and each clue will indicate a real state of things (visibility).

Consistency is probably the most difficult feature of all to achieve in a user interface. For this, it is necessary to form classes of similarities and differences, and then apply the same rules (language, abbreviation, colours...) to qualify the related information. The classes requested by consistency should be kept to a minimum.

Consistency is more difficult to achieve when different people participate in the same development project. One programmer might like to write warning messages all in capitals, another in lowercase. There is always a certain number of issues that remains unaddressed in team projects, and the style of warning messages might be one of those. But when they will be apparent for the operator, he will be led to draw conclusions that in reality are unfounded.

A difficulty in achieving consistency is when several conflicting rules apply for a certain representation and it is not obvious what rule holds over the others. To

An example of simple and "visible" display for process control applications is shown in Figure 4.1. It is not necessary to read a numeric value or check an handbook to find out whether the displayed value is within the allowed range.

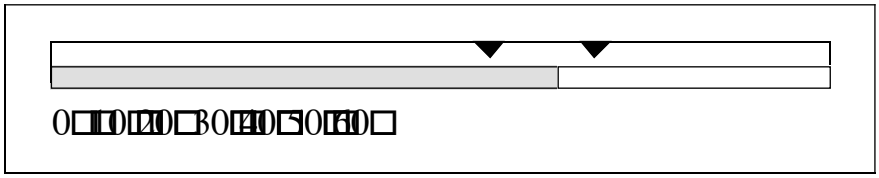


Figure 4.1 Example of intuitive display output

The aspect of technical display instruments is the result of a development that took several decades. Pointer instruments provide immediate information about the relative magnitude of values and their trend; digital instruments show information with higher precision, but are more difficult and take longer time to be understood. In addition, many digital instruments do not give clues about the precision of the displayed data. In some cases are displayed even five-six figures where only three make sense. How can the user know it?

In some computer systems, visibility is obtained by using pictorial **metaphors** related to everyday experience. On the computer screen is shown an **icon** that represents an operation of the machine by means of known symbols. For example a pen can indicate something to write with and a magnifying glass a zooming feature to look at data more in detail. The symbols are not real, they relate the operation of the known objects with similar operations for the computer. In this way the symbols give visual clues to their function and operation.

Metaphors have nowadays become something of a cult object, at least for part of the computer user community. It is however important not to overestimate their importance, especially in industrial process control applications. The comparatively unskilled office user might find it pleasant and easy when everyday symbols indicate system functions. For process operations, it is probably better to indicate things for what they are and not resort to different representations. When metaphors from different sources are mixed, this may add complexity that

untrained personnel and programmers with no formal instruction about human-computer interaction principles.

On the other hand, the design of screen pages alone does not solve more complex problems in human-computer interaction. Therefore the layout designer, rather than following recipes, should have a wider margin of freedom while respecting a few basic principles.

Norman (1988) has identified three basic principles valid for all kinds of applied and functional design and consequently also for user interfaces. These principles are simplicity, visibility and consistency. They have to be understood as framework and not as an immediate design guide for specific details.

Simplicity

Simplicity is the most important rule for all kinds of design. In process-control applications, every display page shows a model of the physical process and its operation. Simplicity means that no useless or irrelevant - excessive - information is presented together with the important data. On the other hand, simplicity should not mean impoverishment in the representation. Due to the fact that simplicity as such does not have an objective metric, it can only be taken as a general principle to be put in the context of other design and evaluation methods.

Visibility

Visibility is the degree of transparency of a system in describing its own operations. Ideally, the user should have the feeling to interact directly with the technical process and not with the computer system (Section 1.2). The operation of the process computer must always be clear as well; the user must all the time know whether he is acting with the technical process or with the monitoring and control computer.

A basic principle of visibility is that some clues, mainly visual (colour, form, shape etc.) indicate the purpose and function of a device. Visibility should provide the link between the physical process *T*, its operations, and the mental model the user has about the process, *M(T)* (Section 2.5).

4 Guidelines for the Practical Realisation of the User Interface

The principles of human-computer interaction described in Chapter 2 and Chapter 3 must lead to the practical realisation of the user interface. Hardware and software components have to be properly selected, designed, constructed and put together. Today, computers and workstations offer advanced capabilities for little money. The question is then not *whether* to use advanced graphical representation, but rather *how* to use it effectively.

In the following, we will assume a hardware user interface based on a graphical terminal, a keyboard with control keys and a pointing device like a mouse. This kind of hardware is readily available, is quite cheap and more than adequate to build a good user interface. This chapter deals mainly with the presentation aspects of the user interface, with emphasis on the coding of messages and commands. Some of the considerations also hold for the design of control panels with pushbuttons or other types of interfaces.

Section 4.1 introduces general design principles; these principles are then reformulated in Sections 4.2 and 4.3 as simple and practical rules and guidelines. Section 4.4 is a brief introduction to the command interface, Section 4.5 deals with menus and Section 4.6 introduces briefly prototyping and evaluation methods for the design of user interfaces.

4.1 General Design Principles

The representation of data on a screen page has drawn a lot of attention in the literature about the design of human-computer interfaces, where practical design suggestions are often given in the form of "cookbook-recipe" rules. This approach might seem unscientific; many rules cannot be applied directly or just do not make sense outside specific contexts. However, the "cookbook-recipe" approach has at least two important advantages: It makes people conscious of different ways to formulate problems and is immediately understandable by