Perceptual control and layered protocols in interface design: I. Fundamental concepts

ARTICLE *in* INTERNATIONAL JOURNAL OF HUMAN-COMPUTER STUDIES · JUNE 1999

Impact Factor: 1.29 · DOI: 10.1006/ijhc.1998.0259 · Source: DBLP

4 AUTHORS, INCLUDING:

Justin G Hollands

Defence Research and Dev…

57 PUBLICATIONS **1,717** CITATIONS

SEE PROFILE

Int. J. Human-Computer Studies (1999) **50,** 489–520
Article No. ijhc.1998.0259
Available online at http://www.idealibrary.com on IDEAL®

Perceptual control and layered protocols in interface design: I. Fundamental concepts

P. S. E. Farrell and J. G. Hollands
Defence and Civil Institute of Environmental Medicine, Box 2000, Toronto, Ontario, Canada M3M 3B9. email: philip.farrell@dciem.dnd.ca; justin.hollands@dciem.dnd.ca

M. M. Taylor
Martin Taylor Consulting, 369 Castlefield Avenue, Toronto, Ont., Canada M3M 3B9. email: mmt@interlog.com

H. D. Gamble
ICE Inc., 489 Queen St. E., Toronto, Ont., Canada M5A 1V1. email: hugh_gamble@iceinc.com

Perceptual Control Theory (PCT) is a general psychological theory based on the tenet, "All behaviour is the control of perception". Layered Protocol Theory (LPT) can be seen as PCT applied to the special case of communication between cooperating partners, both controlling their own perceptions. PCT and LPT can be applied to the design and analysis of human–machine interfaces, although LPT may be more tractable in many cases. LPT is discussed in the context of the analysis and redesign of the interaction between a pilot and a Control Display Unit (CDU) in an operational helicopter.

1. Introduction

Consider the following interaction (shown with comments in Figure 1). A pilot is flying an advanced helicopter that accepts voice commands to which it responds by voice. The pilot is attempting to set a waypoint, a place along the path to which, some time later, the helicopter will direct the flight.

- *Pilot*: Set a waypoint at Beaverton marker.
- *Helicopter*: Do you mean Badgerton marker?
- *Pilot*: Yes, Badgerton.
- *Helicopter*: In place of the waypoint already set at Westville?
- *Pilot*: Yes.
- *Helicopter*: Waypoint at Badgerton marker has been substituted for previously set waypoint at Westville, between waypoints already set at Stellarton and Dunville.

What has happened here? The pilot has a task. He wants to be able to fly between Stellarton and Dunville by way of Badgerton. The hypothetical helicopter is so designed

1071-5819/99/060489 + 32 $30.00/0

that it is able to use a geographic database to produce optimized sequences of waypoints or flights between end points specified by the pilot. The pilot can choose to accept the waypoints the helicopter chooses autonomously, or can review them, depending on his trust that the helicopter will do a good job. Sometimes, as in the example, the pilot may have a reason for wanting to fly by some specific route rather than a route that might be better according to the helicopter's internal criterion. The helicopter can discover this only if the pilot acts in a way that the helicopter can detect and interpret as the pilot's wish that the route be changed.

The pilot wants to see that the helicopter is in a state that will direct him along the desired route. This means that he must perceive the helicopter to have set a waypoint at Badgerton in place of the one at Westville (see Figure 1 for a sketch map of the relative locations of these waypoints). The helicopter not being able to read the pilot's mind, the pilot must act in some way if the helicopter is to do what is necessary for him to achieve the desired perception. In this example, his action is to speak, though many other different actions might plausibly have achieved the desired result—perceiving the helicopter to be in a state in which the flight path will be what the pilot wants.

After the pilot's initial utterance, the helicopter perceives the pilot as saying that a waypoint should be set at a place that is implausible according to the helicopter's model of route criteria. Perhaps there was a recognition error by the helicopter, or perhaps the pilot misspoke. Either way, the helicopter identifies an alternative word that is a plausible substitute and proposes it.

The pilot having acknowledged the word-level correction as being proper, the helicopter next examines the resulting route, which would be either Stellarton–Westville–Badgerton–Dunville or Stellarton–Badgerton–Westville–Dunville, both of which involve an unusual jog in the route. But a plausible route would be created if the new Badgerton waypoint was substituted for the Westville waypoint. The helicopter proposes this possibility. The pilot might not have anticipated this substitution, having intended only to avoid a potentially dangerous leg between Westville and Dunville by flying Westville–Badgerton–Dunville, but the proposed rerouting would accomplish the same purpose more readily. His perception of the new route then would satisfy his highest level goal, better than the way he originally conceived.

In this example, three levels of abstraction are involved in the dialogue. From the pilot's viewpoint, he wanted to perceive the helicopter as having set a route that went from Stellarton to Dunville, but avoided the direct path between Westville and Dunville. For this dialogue, that is the top-level reference perception, goal, or—using terminology we shall define later—Primal Message.

At the next level, the pilot has goals to see waypoints set in the order Stellarton–Westville–Badgerton–Dunville. This goal is not achieved in the dialogue, because it is only a means whereby the pilot can achieve his higher-level goal, and the helicopter proposes a better way of achieving the same higher-level goal. Goals change in the course of many dialogues, if the purpose of the dialogue is accomplished in other ways.

The third level in the dialogue is the level of words. What from the pilot's point of view should have been perceived by the helicopter as "Badgerton" was perceived as "Beaverton", but since Beaverton did not fit the helicopter's model of how routes should be set, it

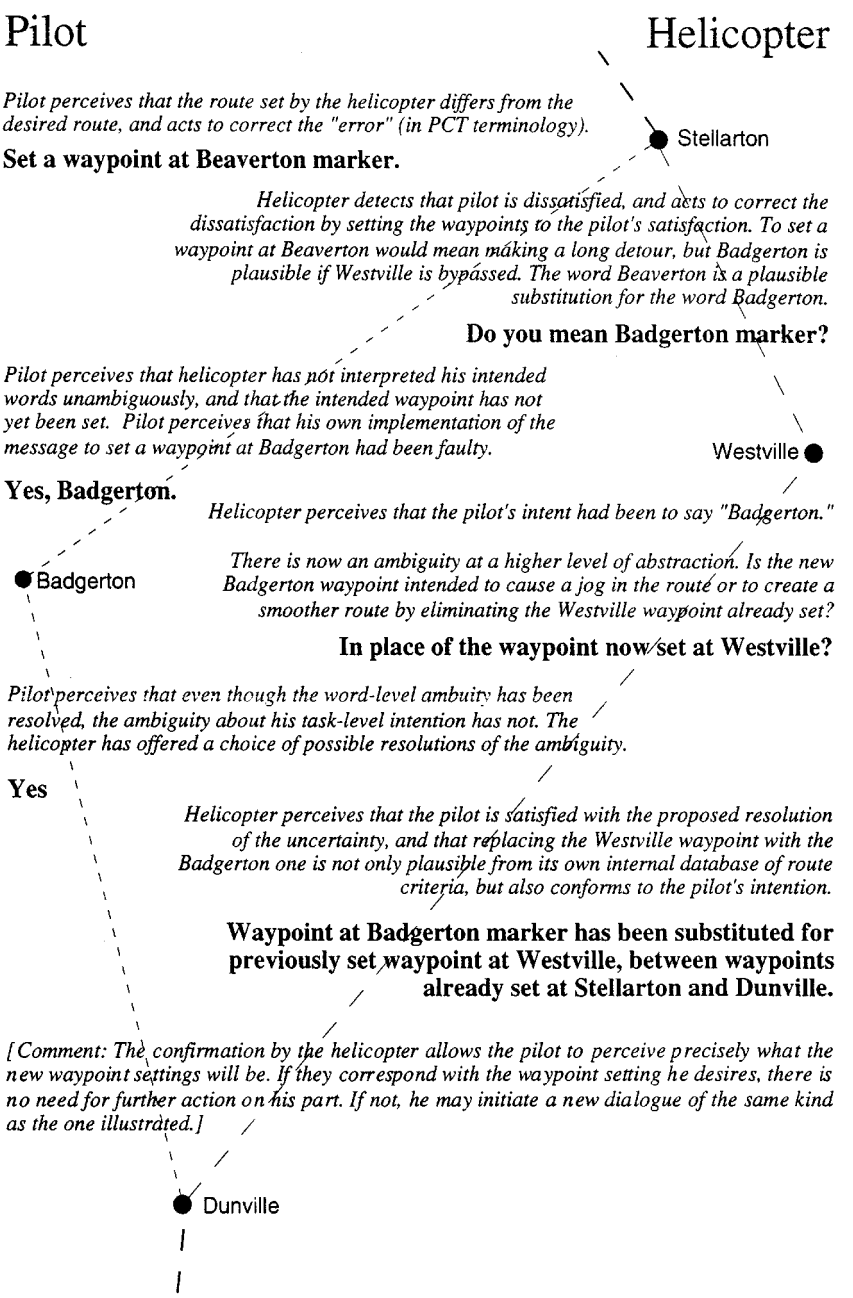

FIGURE 1. Commentary on the dialogue between the Pilot and the Helicopter, superimposed on a sketch map of the locations of the waypoints mentioned.

was rejected as probably not what the pilot wanted, even though it was what he said he wanted. If the pilot had answered the helicopter's "Do you mean Badgerton marker?" with "No, I do mean Beaverton" then the new waypoint, however implausible, would have been accepted, possibly after further dialogue about waypoints intermediate between the existing ones and the distant Beaverton marker.

To recapitulate, the three levels are (1) task—get safely from start to end; (2) route detail—set a route that efficiently achieves the task goal; and (3) words—communicate words that effectively support the achievements of the route details. At each level, the pilot is acting on his world (the helicopter) to bring his perception of its state to a desired condition, which is called the *reference state*. The pilot is not changing the helicopter's state directly, but so long as the state he perceives the helicopter to be in is not its reference state, he acts. The helicopter sets and resets its own state according to its internal criteria as well as according to the pilot's actions. It has a reference state for the pilot to be satisfied with its settings, so it will continue to modify its setting so long as the pilot continues to show dissatisfaction by prolonging the dialogue.

What we are illustrating in this small example is *perceptual control*. Both the pilot and the helicopter are acting so as to affect a perception of their world. The pilot's world contains much more than just the helicopter's waypoint settings, and the helicopter's world contains more than just the pilot's state of satisfaction with the waypoint settings. But the waypoint settings are part of what the pilot can see and affect, and the pilot's satisfaction is part of what the helicopter can sense and affect. The theory that deals with this kind of interaction is Perceptual Control Theory.

Perceptual Control Theory (PCT; Powers, McFarland & Clark, 1960; Powers, 1973, 1978; Robertson & Powers, 1990) applies to purposeful interactions between living organisms and the environment in which they live. In particular, it describes the psychology of those interactions—what people do, and why. Other papers in this issue are concerned with the theory itself and its applications in fields as disparate as simple tracking, the maintenance of one's self-concept and the structures that are observed in the movements of crowds. The present paper and its companion (Taylor, Farrell & Hollands 1999, this issue) deal with the use of PCT in the analysis and design of user interfaces.

The position taken in these papers is that a user's interaction with a machine constitutes a *dialogue*. Even though the machine is neither human nor truly purposeful, we argue that the analysis of human-computer interaction can be treated as if the interaction were between two humans, each having perceptions and goals that the other cannot know precisely. PCT can be applied to each dialogue partner separately, the other partner being treated as part of the external environment. Between two partners in a dialogue, however, the interactions are more complex than the interactions between a person and a rock in the external environment that the person wants to move. The other person may, for instance, not *want* to cooperate, while a rock has no choice. Although PCT can, in principle, accommodate this kind of complexity, we simplify matters by using a specialization of PCT that we call Layered Protocol Theory (LPT), in the same way as chemistry is simplified by coalescing the quantum-mechanical details into parameters such as reaction rates and valences.

LPT was originally developed independently of PCT (Taylor, McCann & Tuori, 1984; Taylor, 1988a, b, 1989). Only later was it seen as a special form of PCT (e.g. Taylor, 1993),

a discovery that greatly clarified some aspects of LPT. In this and a companion paper (Taylor et al., 1999, this issue) we develop LPT and its relationship with PCT. In the present paper, we illustrate the use of LPT in the context of a unit that sets navigation and communication parameters for helicopter operations, while in the companion paper we examine a central component of LPT that we call the "General Protocol Grammar" (GPG).

Before discussing LPT further, we contrast PCT with two other framework views of psychology.

2. Perceptual control theory

2.1. CLASSES OF PSYCHOLOGICAL THEORY

PCT has a core tenet ("All behaviour is the control of perception"). At first glance, this seems to challenge both traditional behaviourism, for which the core tenet might be "All behaviour is caused by stimuli", and cognitive psychology, for which the tenet might be "All behaviour is determined by planning". And it does challenge these approaches. Let us examine briefly how.

In this issue, Powers and Bourbon strip the question to its bare bones in the context of a very simple experiment. They argue that there are three possibilities for the way someone can interact with the world, each possibility leading to a different style of psychology . The first style might be called Stimulus–Response (S–R). In an S–R style of psychology, people are seen as transformers, taking in stimuli that may well be complex, and emitting responses that are functions of the incoming stimuli. The second style might be called "cognitive" or "outflow". In a cognitive style of psychology, people are seen as planners, deciding the actions needed to accomplish a desired result and then executing those actions. The style of PCT is "control". In a control style of psychology, people act to reduce the difference between a perceived current state and a desired (reference) condition for that state, changing how they act if the perceived state is not converging to the desired state.

The idea behind S–R types of psychology is closely linked with the idea of learning through classical conditioning [Figure 2(a)], in which a particular action is tied to a particular stimulus pattern. The idea behind "outflow" or "cognitive" psychologies is equally closely linked to learning through operant conditioning [Figure 2(b)] in which a particular action is tied to a particular desired result. Both of these approaches lead to an emphasis on the *action* that is fixed by learning, whereas perceptual control leads to an emphasis on the *perception* to be stabilized. According to PCT, neither classical nor operant conditioning accurately describes learning, but we ignore this issue for the purposes of the present paper [see, for example, Powers (1992) for a discussion of the PCT approach to learning].

Driving a car provides an analogy that illustrates the critical difference between fixing action and stabilizing perception. On average, when one drives down a straight road, the steering wheel is centred. It would be natural, then, to assume that the best way to drive would be to centre the wheel. But if one chooses to fix the steering wheel at the centre, one will soon find one's car in the ditch. If, on the other hand, one varies the setting of the steering wheel so as to perceive the car to be always in the middle of its lane, one can

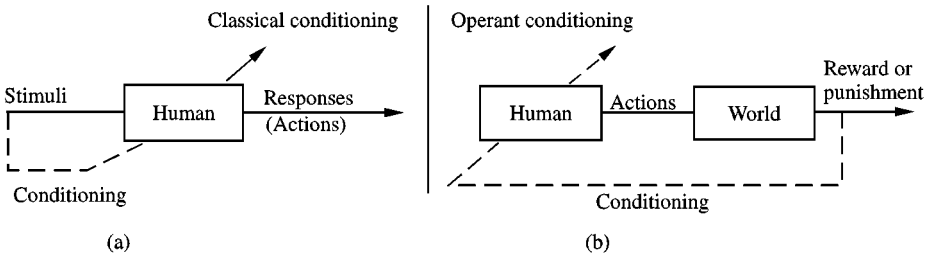

FIGURE 2. Schematic representations of classical and operant conditioning. The dotted line denotes a signal that is used to modify a transformation function.

drive happily for as long as one wants, while the steering wheel is, on average, centred. A corollary to "All behaviour is the control of perception" is "Constant results are achieved by varying action, varying results by constant action".

The contrasts among the three views may be exemplified by caricaturing their approaches to the problem of picking up a glass and drinking from it. In an S–R view, the person has seen a glass in this position before, and has been conditioned to perform certain muscle tensionings in a sequence that leads to the glass being applied to the mouth. In a cognitive view, the person computes "inverse kinematics" to plan a sequence of muscle tensionings that will result in the glass being applied to the mouth. In a control view, the person has a reference perception of the glass being applied to the mouth, which requires a reference perception of the hand being applied to the glass; since the actual perception of the hand is that it is not being applied to the glass, the hand is moved until it is perceived as being applied to the glass, and then hand and glass together are moved until the glass is perceived as being applied to the mouth. The muscle tensioning sequences may be observed as they happen, but they are neither retrieved from memory (as in the S–R view) nor precomputed (as in the cognitive view).

Actions, in PCT, are ordinarily a *by-product* of controlling perceptions, not the result of some intention to act in a particular way. This principle is at the heart of the Layered Protocol approach to interface design.

2.2. A CANONICAL CONTROL LOOP

A controller not only must be able to perceive some state in the outer world, but also must be able to act so as to affect that state. The only action available to a simple organism such as a bacterium might be to move when the perceived environment is too hot, too cold, too salty or otherwise inappropriate. It continues to move until the perceived environment becomes more suitable—nearer its reference value. A complex organism such as a human has far more opportunities for action, and can affect a far more complex variety of perceptions, such as the level of pleasure experienced by a friend, or the intricate structure of the software being designed to support a proposed user interface.

Birth endowment and personal learning have allowed a person to establish reference values for perceptions of certain states of the world, values that together allow the person

at least to stabilize his or her internal chemistry. In everyday language, these reference values are *goals*, though the word is a bit misleading; in everyday speech a goal that has been achieved is no longer a goal, whereas a reference value is a reference value whether or not it matches the corresponding current perception. Nevertheless, we will use the term "goal" interchangeably with "reference value" in the following.

If the perceived state of some aspect of the world is different from its goal state, then the person is likely to act so as to bring it closer to the goal. When energy supplies are low, the person may perceive a state we call "hunger", for which the person has a reference value near zero. The perceived state being different from its reference state, the person is likely to act to reduce the hunger perception. To do this the person will set reference values (goals) for some "lower level" perceptions, such as seeing food, seeing money pass to the owner of food, feeling the food in the mouth and so forth. If, on the other hand, the person is stuffed, no action is required to bring the perception to its reference of "zero hunger", the output of the "hunger control" system therefore sets the reference value for feeling food in the mouth to zero, and the person does not act to get food even if it is available. Goals change over time, as a person's internal needs change.

The above discussion leads to a canonical description of a control loop. We call the internal part of such a loop an *Elementary Control Unit* (ECU), as shown in Figure 3(a). The canonical control loop of PCT has two sources of input: a reference signal, which comes from the outputs of one or more higher-level control systems and specifies what the ECU tries to perceive, and a disturbance from the outer world that affects what it does perceive. Likewise there are two outputs from the ECU: the side-effects that are distributed over aspects of the external world not contributing to the perceptual signal, and the value of the perceptual signal itself, which is available as a possible component of the input to the perceptual input functions of higher-level ECUs.

PCT is usually conceived as a hierarchy of ECUs like that of Figure 3(a). These ECUs are connected so that the output signal of an ECU at level n contributes to the reference inputs of several ECUs at level $n-1$, and the perceptual signal formed by an ECU at level $n-1$ serves as an input to the perceptual input functions of several ECUs at level n.

Except when it comes to the muscles acting on the outer world, all "actions" in PCT consist of setting reference values for lower-level perceptions. This fact is important when it comes to approaching the design of interfaces. We will ask always what controlled perceptions are implied by a design choice, not what actions are to be facilitated. The latter usually fall out naturally from consideration of the former. In Figure 3(b), the distribution of the single output to, and the reception of inputs from, intermediate levels between the depicted ECU and the environment is shown in the form of a multiplicity of connections to and from the outer world, affecting a variety of observable physical variables.

The controlled variable of any control loop is the perceptual signal, which is created by a Perceptual Input Function that has inputs from possibly many sources, including possibly disparate sensory systems. The form of the Perceptual Input Function of any ECU determines (or defines) a Complex Environmental Variable (CEV) that is a function of physical observables in the outer world (and possibly also of variables internal to the organism).

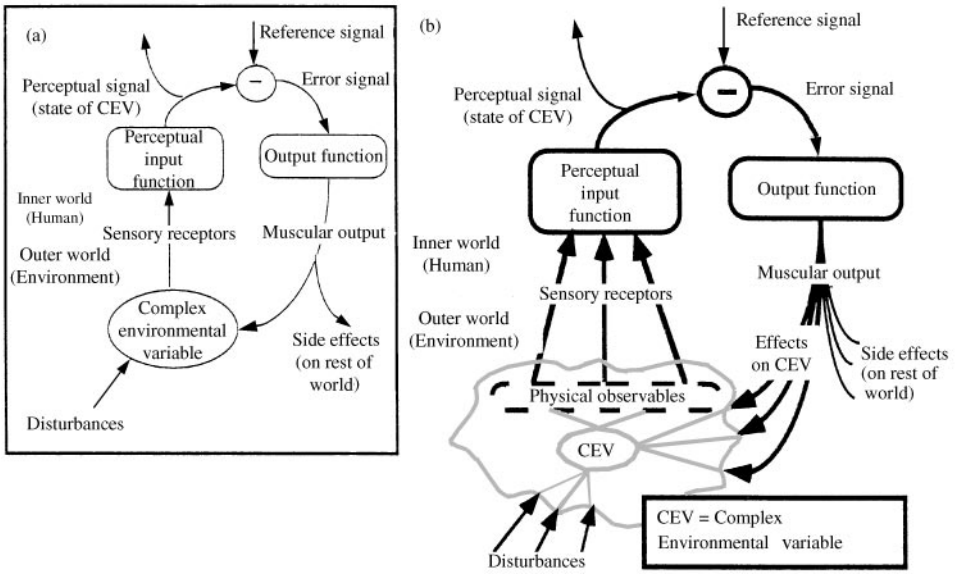

FIGURE 3. A canonical control loop in PCT. The loop has two inputs (the reference signal and the disturbance from the environment) and two outputs (the perceptual signal and the side-effects on the environment). (a) The simple loop, in which all signal paths are shown as simple scalar values. (b) Showing generic situation, in which the control system acts on and senses many different physical aspects of the world through intermediate levels of control. Intermediate levels of control are subsumed into the fan-in and fan-out of the connections to the environment.

The CEV of an ECU is not an identifiable physical entity, but is defined by the Perceptual Input Function of the ECU in question. The same physical observables may be used by quite different ECUs for control of completely different perceptions. For example, in a car, the driver can hear noises from many sources, can see a variety of lights and gauges, and can feel different vibrations and air movements. From these sensory possibilities, a perception related to a tape player (e.g. Haakma, 1999, this issue) might be that "the tape is playing". The corresponding CEV is that the tape is in a playing state. From the same set of physical observables, another perception might be that the car engine is running smoothly, and yet another that the passenger is talking. Although a CEV is *composed* of physical observables that an external observer could detect and influence, it is *defined* by the perception that is being controlled.

Disturbances from the external environment affect the physical observables of which the CEV is composed, thereby influencing the perceptual signal. In the ECU, the perceptual signal is compared with its reference signal, the difference constituting an "error" signal. The output function works on this error signal, producing output that is distributed (usually through other, lower-level, control systems) to various physical effects on the environment, some of which influence the physical variables that contribute to the CEV. Those effects alter the value of the perceptual signal, in the direction of the value of the reference signal if control is effective, reducing the error. The control loop is complete.

A disturbance is not independently observable by the ECU whose CEV it disturbs, though other control systems may, of course, observe all or part of that disturbance as inputs to their own Perceptual Input Functions. Likewise, the side-effects of the action output of the ECU are, by definition, not observable by the ECU.

Like all control loops, the canonical control loop of PCT is a *negative feedback* loop. In engineering, "negative feedback" means feedback that decreases the error and stabilizes some variable. "Positive feedback", on the other hand, always causes an affected variable to run away to a limit, possibly even to the point of physical explosion. To paraphrase George Orwell, "Positive feedback, bad; negative feedback, good". This should be contrasted to the usage sometimes found in dialogue studies, where "positive feedback" may mean something that encourages or agrees with the partner, almost the opposite of the engineering usage adopted in PCT.

In LPT, each partner acts (not always successfully) so as to perceive the other partner as coming to the desired state. A minimal dialogue, therefore, consists of the interaction of one negative feedback loop controlling some perception in the initiating partner with another negative feedback loop controlling some perception in the other partner. In a successful dialogue, the interactions of these and other loops results in negative feedback overall. But not all dialogues are successful; if the overall feedback in a dialogue turns out by some mischance to be positive for any extended period, the dialogue partners might even come to blows, or if one partner is a computer, the other might pound on its keyboard!

3. Layered protocol theory

Although PCT can, in principle, be used to model any interaction between a person and the outer world, including communication with another person, computer, or a machine, its specialization in the form of LPT (Taylor, 1988*a*, *b*) is simpler for communication purposes. The basic tenet of LPT is: "All communication is the control of belief".

A "belief" in LPT means a group of simple perceptions. The structure of the layered protocol hierarchy is the same as that of the PCT control hierarchy, but the controlled variable in each loop is more complex than the simple scalar value that is the perceptual signal of an elementary control structure. By combining several perceptual signals into one "belief" we can deal with some issues more readily than we could by considering all the interactions among elementary perceptions. Accordingly, several ECUs are subsumed into one more complex entity we call a "Protocol Node". How this is done is sketched below. The approach is like that of a chemist who finds it easier to deal with known properties of molecules than with all the interactions of the atoms that constitute them, even though dealing with the atoms would eventually yield the same, or perhaps a more accurate, answer.

The interaction between two humans, or between a human and a machine, is driven by the differences between the many reference beliefs of both partners and their corresponding current beliefs. In LPT, each partner senses (among other things) the actions of the other partner. The other partner's actions are interpreted at different levels of the hierarchy of Protocol Nodes, and affect the beliefs at all levels (or, in PCT terms, the other partner's actions affect the complexes of perceptions throughout the hierarchy).

3.1. ELEMENTS OF LPT

From the originator's point of view, a dialogue proceeds as follows: Some reference perception differs from the corresponding perceived state of the world, and the perception might be influenced by some action the recipient might do. The originator acts (through a hierarchy of ever lower-level control units) to affect the recipient in such a way that the recipient acts on the originator's CEV in such a way as to move the originator's controlled perception closer to its reference value. That is all there is to it, though a great deal of complexity may be hidden in the phrase "through a hierarchy of ever lower-level control units".

Colloquially, a "message" is "what the originator wants to get across". What the originator wants to get across is the state at which he wants the recipient to arrive, at least insofar as it differs from the state in which he believes her to be. That state may result in some overt action on the part of the recipient, by inducing her to change some reference levels for her own controlled perceptions, or it may change some uncontrolled perceptions that represent her knowledge of the world. In the one case, success for the originator is having the recipient act overtly, and in the other it is to perceive that she has come to know what he wants her to know.

From the recipient's point of view, the dialogue is a little more complex, at least in a cooperative dialogue. In a cooperative dialogue, the recipient's main reference perception is to see the originator as being satisfied. The action of the originator in initiating the dialogue is a disturbance to that perception—the recipient perceives that the originator is not satisfied, because the originator acted. The recipient therefore acts in such a way as to alter the perception of the originator's state back to "satisfied". This ordinarily entails trying to understand in what way the originator is unsatisfied, which involves various beliefs about states in the originator. The "way the originator is unsatisfied" is exactly what is usually called "the message" in a dialogue. Colloquially, to say "He gets the message" is the same as saying "He is now doing what I want", or "He now knows what I want him to know".

When the recipient has understood what the originator wants, and has acted so that the originator believes this, then the originator's reference belief matches the originator's belief. The originator is satisfied. What remains is for the originator to act so that the recipient can see that this is so, thereby allowing the recipient's perception to match its reference of "the originator is satisfied".

In LPT, we give the name of "message" to those actions that affect the partner's beliefs, whether the effects are intended or inadvertent. Messages are not ordinarily formally coded and decoded, any more than actions are fixed in simple perceptual control; like the actions in steering a car, messages evolve over the course of an interaction until both parties can bring their beliefs to their reference states. This approach to the changing nature of messages with a fixed objective was foreshadowed by the design-interpret approach to communication discussed by Thomas (1978), who was drawing from game theory, not from PCT.

3.2. MESSAGE TYPES

Figure 4 illustrates in simplified form the two main kinds of control loops involved in the dialogue. The Originator wants the Recipient to come to some state that we call the

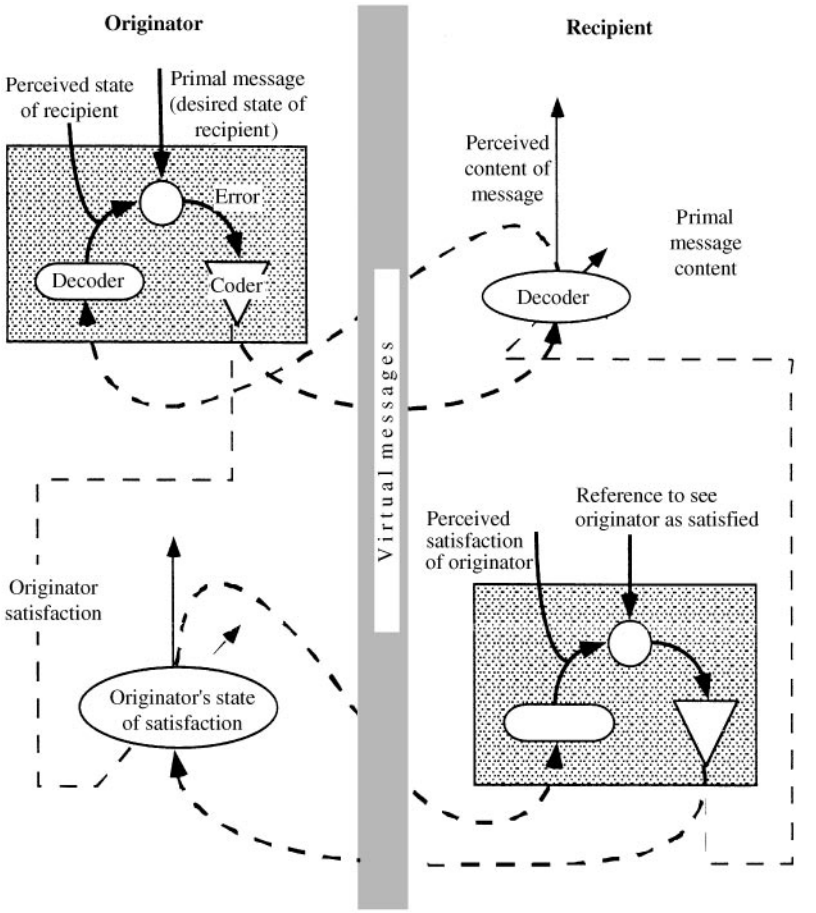

FIGURE 4. The Originator controls a perception of the Recipient's state, sending virtual messages intended to bring it to a reference condition. The Recipient passively interprets the Originator's virtual messages to discover what the content (the desired state) may be.When the Recipient has come to the intended state, the Primal Message has been understood. In a separate control loop, the Recipient tries to bring her perception of the Originator's satisfaction to its reference level ("fully satisfied" if the dialogue is cooperative). The output actions of one control loop may affect the passive element of the other loop. For example, one of the effects of the Recipient perceiving the Originator not to be satisfied with her interpretation of the Primal Message may be that she changes her interpretation, and one of the effects of the Originator perceiving the Recipient not to be coming to the desired state is that he changes his level of satisfaction in a way perceptible to the Recipient.

"Primal Message". Any difference between the actual and the desired state of the recipient constitutes an error signal that is likely to induce output by the Originator. We call such output a "Virtual Message". Virtual Messages affect the Recipient's understanding of the primal message; but they do it passively, since the Recipient is not trying to influence the content of the Primal Message. The content of the Primal Message is, for the Recipient, an uncontrolled perception.

Virtual Messages from the Originator allow the Recipient to understand the Primal message; other kinds of Virtual Message from the Originator allow the Recipient to

perceive how satisfied the Originator is with the Recipient's current understanding of the Primal Message. The perception—the Originator's degree of satisfaction—is one the Recipient *is* trying to control, and the only way to do that is to get the originator to believe that the Primal Message has been properly understood. The output of the Recipient's "originator satisfaction" control system therefore sends Virtual Messages to the Originator, and at the same time affects the way the Recipient interprets the Originator's Virtual Messages.

There are, of course, other perceptual control loops active in a truly cooperative dialogue. The Originator wants to see, for example, that the Recipient is satisfied that the message has been correctly interpreted, and the Recipient wants to see that the Originator is satisfied that the Recipient is satisfied. Virtual Messages from either partner to the other are the mechanism through which these perceptions are controlled. All of these loops affect each other in various ways. In total, we identify in the companion paper (Taylor *et al*., 1999, this issue), nine perceptual control loops in each partner that relate to the other partner's perception of how the communication is progressing, and a tenth in the Originator, which the Primal Message content is controlled.

3.3. PROTOCOL NODES

In LPT, we compact the nine or 10 separate but interacting perceptual control units shown and implied in Figure 4 into a single construct we call a Protocol Node, which takes the place of the ECU of PCT. Figure 5 shows two stages in the process of compacting the interacting control loops of Figure 4 into a pair of Protocol Nodes, one in each partner.

The first stage in compacting the several perceptual control loops into a single Protocol Node recognizes that both the Primal Message content and the various states of satisfaction are communicated through virtual messages across the physical medium that separates the two partners. In this first stage of compaction, therefore, we add complexity to the output function, call it a "Coder", and allow it to serve all of the different action outputs of the Protocol Node [Figure 5(a)].

In the second stage of compaction, we take advantage of the same fact that there is but one interface supporting all the different kinds of effects each partner has on the other, and combine the nine or 10 perceptual input functions into a single, more complex, "Decoder". The output of each Decoder is a belief structure that contains both elements of the content of the Primal Message (the state the Originator wants for the Recipient) and elements of the perceived satisfaction of both parties about the progress of message transmission. Two of these components are shown in Figure 5(b) as outputs from each of the two Decoders. The usual Protocol Node diagram used later in this paper and elsewhere shows only one belief output, which incorporates all the content and satisfaction aspects of the message transmission.

The "Primal Message" is the Originator's reference belief—the desired state of, or action by, the Recipient. The difference between the Primal Message and the currently perceived state of the recipient is the error. The error is determined in the Model component of the Protocol Node, which takes the place of the simple comparator of an Elementary Control Unit. In addition to the Primal Message component, the Model

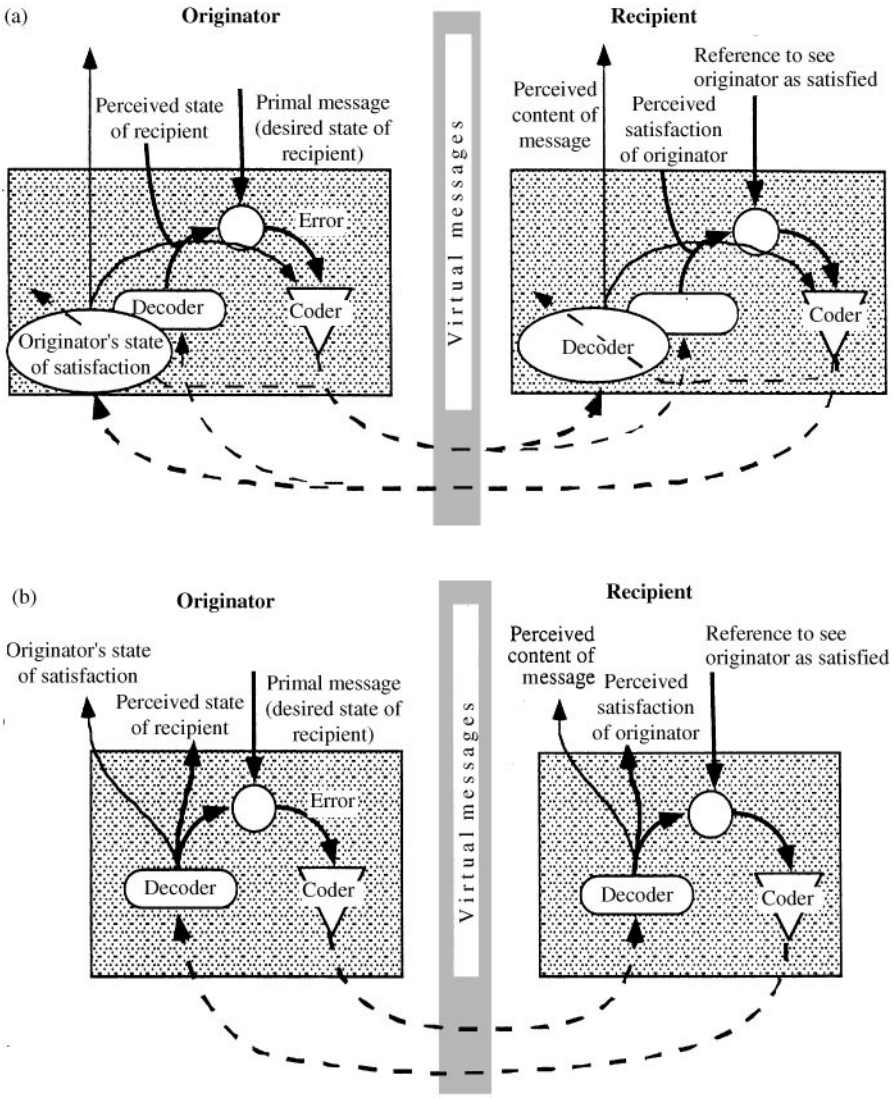

FIGURE 5. Two stages in compacting the interacting control loops into a pair of Protocol Nodes, one in each partner: (a) The passive beliefs usually use the same supporting virtual message channels as the controlled beliefs, so the paths across the interface are shown combined; (b) the input interpretive functions that correspond to the Perceptual Input Function of the control loop are combined with the passive perceptual functions to form one complex "Decoder". The final stage of combination is to combine the passive perceptions with the controlled perceptions into one complex "belief" as is done in most depictions of Protocol Nodes in discussions of LPT (e.g. Figure 5).

must also deal with error in the various controlled perceptions of satisfaction. The Model therefore compares a complex reference belief state to an equally complex perceived state of the world to produce a complex "belief error".

Compacted in this way, a Protocol Node contains three main elements: a *Decoder* (corresponding to the Perceptual Input Function), a *Model* (similar to the comparator but more complex because of the greater complexity of "beliefs" as opposed to "perceptions"), and a *Coder* (corresponding to the Output Function). The perceptual signal of the ECU corresponds in a Protocol Node to a belief about some state of the partner, the nature of which is defined by the characteristics of the Decoder.

3.4. VIRTUAL MESSAGE TYPES

The error output from the Model is the input to the Coder, just as the difference between the reference and perceptual signals is the input to the Output Function in the PCT Elementary Control Unit. The Coder produces output in the form of a "Virtual Message" that affects the recipient. Virtual Messages are "virtual" because they exist as implementations only when executed by lower-level units. An invitation to a party, for example, could be a virtual message referring to a whole complex of understandings by the recipient, such as where and when the party is to be held, what kind of party it might be, who will attend, and so forth, all of which could, under some circumstances, be implemented by the pair of words: "Coming tonight?"

Virtual messages are classified into two groups. Some virtual messages assist the recipient to interpret the Primal Message, while others help one partner to assess the beliefs of the other about the progress of the recipient's interpretation. We call the former "Content messages", the latter "Feedback messages" or "Protocol messages". The different message types are discussed in depth in the companion paper about the GPG (Taylor *et al*., 1999, this issue). Here it suffices to note that each virtual message is implemented by a lower-level protocol node in which it serves as a reference state for something its originator wants to be able to believe about the partner. That is, virtual messages become *primal messages* for protocols at lower levels, as shown in Figure 6. Each virtual message, therefore, initiates and is implemented by an entire sub-dialogue involving virtual messages at the new, lower, level. Sometimes, of course, the originator already does believe the reference belief about the partner, in which case the virtual message has a null implementation.

Virtual messages are reimplemented at successive levels of the hierarchical structure of PNs within a single partner. Once the messages reach the level of the physical observables (or the interface) then they cross over to the partner's hierarchy of protocol nodes. The dashed lines in Figure 6 labelled "virtual message" represent the transmission of messages between the two hierarchies through the interface.

The characteristics of virtual message at any one protocol level are different from the characteristics of the virtual messages that implement them at a lower level. The message that lunch is ready is not the same as the sequence of letters L U N C H [space] I S [space] R E A D Y. The same *call to lunch* might be implemented by striking a gong, yelling, or making eating gestures, rather than by the written word. The success of the "ready-lunch" message transmission may be shown by the appearance of the person at the lunch table, no matter how the message is implemented.

It is not only the Originator of the top-level Primal message who sends virtual messages. So long as the Recipient does not perceive the originator to be satisfied with the communication, the recipient is likely also to send virtual messages. These are actions

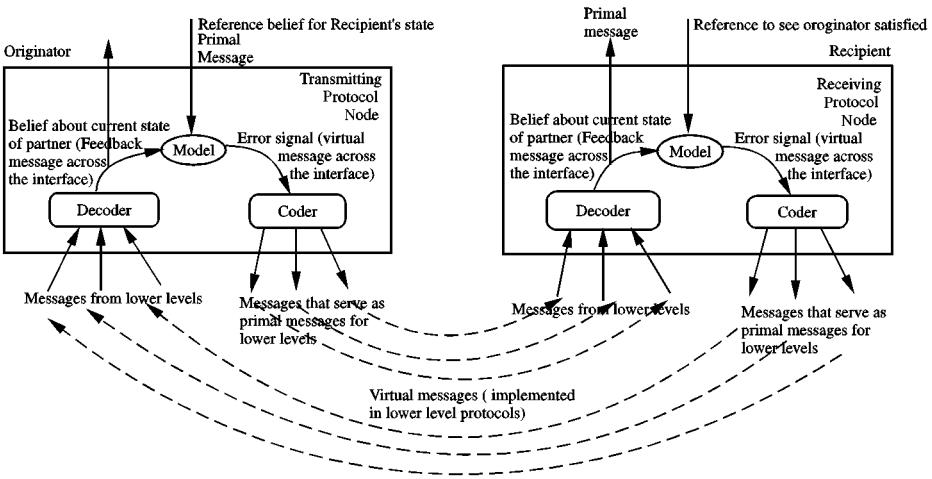

FIGURE 6. The relationship between a Transmitting Protocol Node and a corresponding Receiving Protocol Node. Each uses its decoder to interpret messages emitted by the other's encoder.

intended to assist the originator to act to help the recipient to interpret the Primal Message. Virtual Messages sent by the recipient of the overall message also serve as Primal Messages in Protocol Nodes at lower levels, nodes in which the Recipient at the higher level is now the originator. Each partner is therefore the originator of some of the virtual messages, or, to put it another way, is "in control" of some aspects of any dialogue.

3.5. WHO IS IN CONTROL?

We are concerned in this paper mainly with the design of interactive interfaces that allow a user to take advantage of a machine's capabilities. The designer plans the appropriate displays and devices so that the design and interpretation of virtual messages can be carried out between communicating, cooperating, partners. Both the human and the machine act as originators of messages at different levels of the dialogue between them, and both act as recipients. The human, however, is the one with the task, to do that which involves the assistance of the machine, and it is the human, therefore, who originates the highest-level Primal Message.

One issue that often occurs in interface design is how to keep the user "in control". This issue comes up largely because of a mindset that leads a designer to think of what the user should *do*. By thinking of what the user should do, and providing for just those actions, the designer constrains what perceptions the user is permitted to influence at any phase of the interaction. The user who is so constrained is, almost tautologically, "not in control".

PCT takes the opposite view. Throughout the design process, the PCT-based designer asks "what might the user want to perceive?" rather than, "what should the user do?" What the user "wants to perceive" is a reference value for some controlled perception, or

at least for a perception the user might choose to control. The output of each control system contributes to the reference values for lower-level control systems, and those reference values determine the perceptions that the user may want to achieve at those levels. Eventually, the lowest level outputs directly cause muscular actions, which affect the perceptions at all the levels. The designer treats the user's actions only as modifying displayed information in ways that will allow the user's perceptions at each level to approach a state desired *by the user*. The existence of effective action mechanisms is implied by the need for the user to affect the controlled perceptions, and they tend to fall out almost as a byproduct of considering the hierarchy of controlled perceptions. In each Protocol Node, the user's actions should influence a cooperative (machine) partner in such a way that the machine's actions directly and obviously influence the user's perception in that Protocol Node.

In human–machine interaction, by concentrating on the user's perceptions and ensuring that perceptions are supported by available actions that reduce conflict while ensuring effective performance on the task, the PCT/LPT approach ensures that the design process is *user-centred* rather than *product-centred*.

3.6. MULTIPLEXING AND DIVIPLEXING

In the helicopter CDU, we use as an example in the latter half of this paper, several lower-level protocol nodes that may be identified as supporting a "Waypoint Setting" node: radio, mode, frequency, long/lat, etc. These perceptions are required so that the pilot can legitimately believe that a waypoint with the desired characteristics has been established. The support of a top-level belief by several independent lower-level beliefs is called *diviplexing* (Taylor, 1989; Taylor & Waugh, 1999*a*, forthcoming). Top-level beliefs are functions of lower-level component beliefs that may be independently controlled, each possibly using a different implementation mechanism. Satisfying the goal belief of each (supporting) component helps ensure accomplishment of the higher-level (supported) belief.

Virtual messages must pass between the originator's Coder and Decoder and the recipient's Decoder and Coder, respectively. We call such a Coder-Decoder-Coder-Decoder loop a "channel" for the virtual messages. A virtual message within a single PN at level k is implemented by the satisfaction of several belief states at lower levels. Each type of controlled belief if represented by a supporting protocol node and channel at level k-1. If each PN in an n-level hierarchy were supported by two lower-level protocol nodes (a transmitting node supporting the Coder and a receiving node supporting the Decoder), which in turn were supported by two more, and so on, then at the nth level there would be 2^n PNs and channels (e.g. 2^n corresponding devices and displays).

It is possible and usual, however, for one channel at lower level k-1 to support two or more independent PNs at higher level k. In the helicopter CDU, both "Radio Setting" and "Waypoint Setting" use perceptions of radio, mode and frequency. That is, the messages relevant to waypoints and those relevant to communication are *multiplexed* onto the same supporting channels. When this is done, the number of PNs at any one level is not constrained by the number of PNs at the levels above. In practice, time and space constraints force the designer to design multi-function interactions, multiplexing

their functions. At the low levels in the CDU interface, multiplexing must occur since the same buttons and screen real-estate support all the different functions of the CDU.

Multiplexing reduces the otherwise exponential growth of PNs that would be needed to describe an interaction. The designer must decide where multiplexing is required for an optimal design, and how it is to be done. If multiplexing is done at too high a level, the design may generalize parts of the conversation that need to be specific. If multiplexing is postponed to lower levels, then the problem becomes to ensure that a possibly large number of protocols can be supported by a small number of effectors, without undue mutual interference.

Multiplexing is not the opposite of diviplexing: multiplexing means that one lower-level channel supports more than one higher-level PN, whereas diviplexing means that a higher-level message channel is supported by more than one lower-level PN simultaneously. Both can occur simultaneously, as shown in Figure 7. In Figure 7, the output of Node $A2$ is diviplexed onto transmitting nodes $P1$ and $Q1$. The outputs of $A2$ and $B2$ are multiplexed onto $Q1$, and the inputs of $A2$ and $B2$ are multiplexed onto receiving node $R1$.

A fundamental problem in multiplexing is to ensure that messages are distributed appropriately at the receiving end of the channel (that they are demultiplexed properly). There are various ways of achieving correct redistribution. In electronics, one way, known as time-division multiplexing, is to send messages from different sources to their intended receivers at separate times known to the sources and receivers. Another method is frequency-division multiplexing. The same channel carries many messages, but the messages are distinguished by the frequency band on which they are carried, and are separated out at the receiving end by different filters. All radio signals are multiplexed in this way onto the radio spectrum.

Both kinds of multiplexing have their analogy in interface design. The analogy to time-division multiplexing is called "being modal". In computer interface design, it is generally considered a good idea to avoid mode-switching, and to try to design modeless interfaces (Norman, 1988). Modeless interfaces do not avoid multiplexing, in the LP sense. Rather, they are analogous to frequency-division multiplexing. We call it "broadcast multiplexing" (Taylor & Waugh, 1999*a*, forthcoming). In broadcast multiplexing, the messages for several higher protocol nodes are intermingled on one supporting channel, and are interpreted by possibly many higher-level nodes at the receiving end. Each higher-level node has access to all the messages, but interprets only the messages

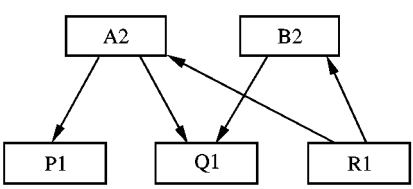

FIGURE 7. Multiplexing and diviplexing. The output of node $A2$ is diviplexed onto supporting (transmitting) nodes $P1$ and $Q1$; the outputs of $A2$ and $B2$ are multiplexed onto supporting node $Q1$. The inputs of $A2$ and $B2$ are multiplexed (not diviplexed) through (receiving) node $R1$. Nodes $A2$ and $B2$ may be either transmitting or receiving nodes, since each requires support for both input and output.

intended for it, just as a radio receiver is subjected to the signals of all the stations in the region, but its filters pass only signals at the frequency of the station to which they are tuned. Messages of all kinds are possible at all times, but their import is determined by their content, not by an explicit switch based on the time at which the message arrived.

To support multiplexing, therefore, the designer must determine whether the linkage between a transmitting PN and its intended receiving partner is asserted explicitly (by means of a separate message on the same or another channel) or through the content of each message. If the former, the designer must try to assist the human user to perceive (and remember) which link the channel is currently supporting; if the latter, the designer must ensure that the structure of each message allows the messages of different types to be discriminated by the receiver. At present, these design decisions are made using the Annotation Views in the LPTool, but a revised tool would allow for explicit representation of the way multiplexing is implemented.

4. Design using layered protocols

The rest of this paper illustrates the PCT/LPT approach to the design and analysis of interfaces. To assist the designer, a first prototype of a design and analysis tool, called "LPTool" has been produced for the Macintosh. We illustrate the design process using a helicopter CDU as an example (Figure 8).

Operational pilots had noted difficulty in using the CDU, which mediates the setting of waypoints and radio communication links in the helicopter. Farrell and Semprie (1997) used LPTool to analyse the existing interface to the CDU, and to design a revised interface. The Layered Protocol analysis identified some specific deficiencies, which were corrected in the revised design.

Many interfaces are developed by a designer who imagines what users would want to do. The resulting interface is then tested by asking representative users to try it. Most interfaces are too complex for the designer to keep everything in mind at once, which implies that the test prototype will contain deficiencies. Correcting these deficiencies may introduce new problems in other aspects of the design. The existing CDU interface is no exception. When using the CDU, pilots must navigate through 70 displays, 10 softkeys, and several embedded menu structures, and may easily lose track of their place within the interaction.

LPTool is intended to alleviate this problem by helping the designer to partition the problem into the individual perceptions that the user may control. For the control of each perception, the designer provides the means to sense the state of the variable (a point that might seem obvious, but one that is sometimes lost, and was lost in the existing CDU interface). The designer also ensures that the user has a means of influencing the controlled perception, ordinarily by setting a reference value for some lower-level perception.

In providing for the user to influence the controlled perception at one level by altering the reference value of a controlled perception at a lower level the designer asserts that there exists another perception that the user might wish to control and that must be considered in the design. Perhaps less obviously, by providing the pilot with mechanisms for sensing the state of some variable inside the CDU, the designer is asserting that there

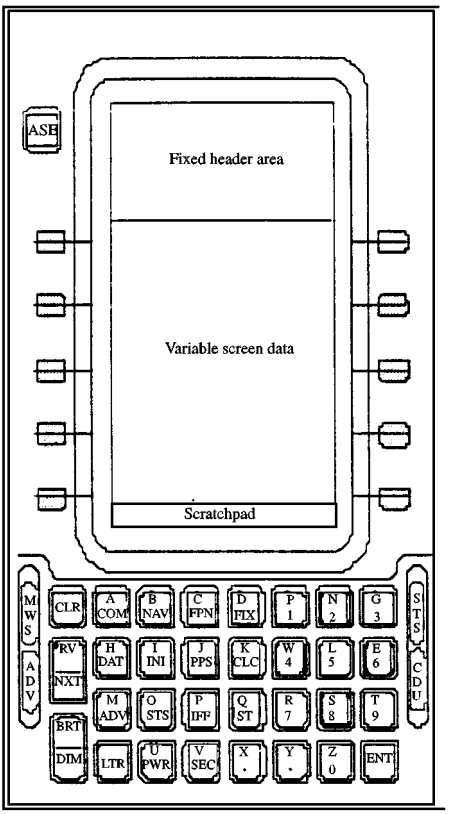

FIGURE 8. A CDU Front Panel (by permission of Canadian Marconi Company). This arrangement is taken as given in redesigning the interface.

is some variable controlled by the CDU; it also must be incorporated in the design as a lower-level perception to be controlled, but by the CDU, not by the pilot. It is a Primal Message from the CDU to the pilot, just as the pilot's intention to change the state of the CDU is a Primal Message from the pilot to the CDU. The two-way message flow that allows the pilot to control the higher-level perception constitutes a dialogue.

The Layered Protocol design process designs the dialogue between the pilot and the CDU at several perceptual levels, each of which may define a sub-dialogue. Each controlled perception is prone to error—a difference between a desired and actual value of the perception. Both pilot and CDU act to reduce the errors, as discussed earlier in this paper, and especially in the companion paper (Taylor *et al.*, 1999, this issue). If improperly designed, these actions may interfere with the pilot's control of perceptions in other protocol nodes or in the pilot's other tasks (such as flying the machine). The designer's job in any interface design is therefore to design Protocol Nodes that will help the user to control the beliefs necessary to the tasks for which the product is intended, without unduly interfering with the control of other perceptions important to the user.

Rather than design the actions that must be performed by a pilot entering, say, waypoint co-ordinates, the designer using LPT first looks at how the pilot's actions might change her perceptions *at the conceptual level of the waypoint*, and at whether these changes will help the pilot determine whether task-level perceptions are approaching their reference values.

For example, the pilot may want to set a waypoint at a location he or she thinks of by name, say as "Badgerton". The designer must at some point consider how to implement displays that will help the pilot achieve the desired perceptual state (perceiving a waypoint that has been set to be at Badgerton). If the designer provides a display that identifies the waypoint only by its map coordinates, the pilot has to translate mentally the place name into the corresponding coordinates, or vice versa. This not only adds to the pilot's work load, but introduces a potentially dangerous source of error at the task level—a wrongly located waypoint—that might not be corrected by the pilot.

Rather than providing a display of the alphanumeric grid coordinates entered by the pilot, the designer might allow the pilot to see a map on which names such as "Badgerton" are marked, or perhaps an alphabetic list of named locations suitable to be used as waypoints for the particular mission. There are many different possibilities. Whatever the display, for the pilot to perceive that a waypoint is set at "Badgerton", it must be such that pilot both can identify that a waypoint is set somewhere and can see that where it is set corresponds to "Badgerton".

The best way to make such a display depends on various factors, such as whether the location of waypoint possibilities is limited to a named set or could take any coordinate value; if the latter, whether the pilot's perception might be aided by referencing to known locations rather than to an abstract coordinate system. Is the pilot expected to understand the nature of the mission in such a way that a map display of the waypoint settings would clearly point up a setting that was dramatically wrong? Such factors might differ from mission to mission, and cannot be taken for granted in the design of the CDU. Other factors affecting the "best" display are more general. Could, for example, the pilot misperceive displayed coordinate data and accept a wildly wrong waypoint setting? Or would a word-level error in perceiving the map reference to be "Badgerton" be likely when the display shows "Beaverton"? Whatever the answers, the questions all deal with the perception "Is this the correct waypoint, correctly set?" Questions about how to act to change the display come later. At this point, the designer must only note that some action must be available to allow the pilot to change a waypoint's properties, and a display must be provided that allows the pilot to perceive those changes.

As the designer continues to develop the hierarchy of controlled perceptions, the design of actions available to the pilot falls out almost implicitly. In the "Badgerton" example, for instance, if the designer decides that a sketch map display is a good way to ensure that the pilot will not make gross errors in the setting, then the designer must permit actions that will allow the pilot to perceive and create changes in the location of some marker on the display. A natural way to do this is to allow the pilot to influence some two-dimensional kinesthetic perception, perhaps using a mouse or a joystick.

At this point, the designer perhaps notices that the pilot also has to fly the aircraft, and to fly the aircraft requires control of the same lower-level perceptions as would be required for using a joystick to move a marker on the CDU screen. The two tasks will interfere. Realizing this conflict, the designer then can ask whether control of the

perception of changes in waypoint location might perhaps be supported by some different lower-level perceptions. Perhaps the map display is good, but the display on the map of location changes might be accomplished in some other way, such as by allowing the cursor to move only to the locations of named viewpoints that are identified by voice. There are possibilities, each of which implies different specifications for lower-level perceptions the pilot would have to control if those actions were to be employed.

Symbolic perceptions are particularly well suited for voice communication, so the designer must ask whether a symbolic representation such as using named waypoints might reduce the potential for interference while retaining the ease with which the pilot can control the "waypoint setting" perception. The designer must now ask whether the control of voice perceptions to support waypoint setting interferes with voice used in support of higher-level perceptions associated with the usual missions of the helicopter. Does the pilot need to use voice for communication, and if so, how should voice used for interaction with the CDU be distinguished from voice used through the same microphone for other purposes? Eventually, sets of actions that minimize conflict will have emerged from a consideration of the perceptions to be controlled in the various tasks the pilot must perform if the mission is to be accomplished.

Such an approach may also reduce the risk that the user's actions will have untoward side-effects that are undetected until too late. It is not enough that the design allows the user to perform the requisite task with ease. Situations in which "the operation was successful but the patient died" are not acceptable. Side-effects are a potential source of problems that the designer must consider carefully, since by definition the user cannot be aware of them—at least not at the level of perceptual control where they are generated. If the outputs of the perceptual control systems at each level are designed so as to be most effective in influencing the controlled perception, they are least likely to have strong effects in other areas, thereby minimizing the problem of side-effects.

4.1. DEVELOPING A DESIGN

In performing the Layered Protocol analysis and redesign of the CDU using LPTool, Farrell and Semprie (1997) developed a general procedure, which will illustrate in the rest of this paper, using the CDU as an example:

1. *Decide whether the design is to take the user's or the machine's viewpoint.*
2. *Choose a protocol node to design, starting and ending at levels the designer can affect.*
3. *Define the lexicon of perceptions the originator (human or machine) might want to control (i.e. define the Primal Message types the chosen Protocol Node will handle).*
4. *Annotate completely the General Protocol Grammar for the chosen node before generating and annotating supporting nodes.*
5. *Ensure that Coders are connected to transmitting nodes at the level below.*
6. *Ensure that Decoders are connected to receiving nodes at the level below.*
7. *Repeat steps 2 to 6 until the analysis or design is complete.*

We consider only the beginning of a design or an analysis, as a full analysis of the CDU is rather lengthy (Farrell & Semprie, 1997). Since we are dealing with a device that is in actual operation, the design process must take the hardware as given, changing only the software. Likewise the tasks for which the pilot might use the CDU are taken as given. In

Step 2 of the procedure, therefore, the designer is constrained to design a protocol node that handles Primal Messages relevant to the predefined tasks, and for which the available hardware limits the perceptions that can be controlled by both human and machine.

Step 1: *Decide whether the design is to take the user's or the machine's viewpoint.*

Since a Layered Protocol description of communication treats the two partners as having a reciprocal arrangement of Transmitting and Receiving protocol nodes, the designer could take the viewpoint of either partner when initiating the design. Design from one viewpoint implies a design from the other. It is, however, confusing for the designer to try to take the viewpoint of both partners in a single design, and LPTool does not permit it. Since the designer cannot affect the human user except by specifying selection and training requirements, and is usually intending to affect the internal detail of the machine, it is often more convenient to take the machine's viewpoint.

If the designer were to take the pilot's viewpoint, one of the pilot's controlled perceptions would be that the CDU should be seen to have accepted a desired radio or waypoint setting. The pilot is therefore the "originator" of the top-level Primal Message. To control the perception of whether the CDU has accepted a desired setting, the pilot would have to be provided with some means of specifying to the CDU what was desired, and some means of seeing whether the CDU has accepted the new setting as intended. From the viewpoint of the CDU, it will act as if it has a perception of how satisfied the pilot is with the present settings of the waypoints and a reference value for that perception of "completely satisfied." The CDU is therefore the "recipient" of the top-level Primal Message. It is not controlling what the setting of the waypoint is to be, but instead is allowing the pilot that control. It must be provided with inputs that allow it to perceive both what the Primal message is, and how satisfied the pilot is with its ongoing interpretation. If the pilot is seen to be trying to change a setting, the CDU will thereby perceive that the current setting is not satisfactory, will interpret the setting the pilot seems to want, and will display to the pilot the result of that interpretation. These are the normal functions of a receiving Protocol Node.

No matter which viewpoint the designer takes, the design process has implications for both partners. A design from the user's viewpoint can and should specify at least some internal detail of the machine, and a design from the machine's viewpoint can and should specify at least some selection and training requirements for the human user. The prototype LPTool does not require, but it permits the designer to take account of the "other" partner requirements implied by the design. Since the designer is designing the CDU rather than the human (unless the design is of a training procedure for the CDU), it is probably easier to design from the viewpoint of the CDU, which is what we will do, using waypoint setting as an example. The user's actions are taken to be inputs, and the CDU's displays as outputs.

Step 2: *Choose a Protocol Node to design, starting and ending at levels the designer can affect.*

What the pilot can see of the CDU is a rectangular alphanumeric display screen with an alphanumeric keyboard below it and a column of soft keys (variable function buttons)

on either side, as shown above in Figure 8. What the CDU can see of the pilot is input through the keyboard or possibly by voice. These characteristics cannot be readily altered by the interface designer, so the design process starts with a predetermined lexicon of low-level messages through which all the higher-level messages must be implemented. This limitation sets the lowest level of protocol nodes that could be considered in the design process.

The highest level of protocol nodes to be considered is set by the prespecified functions of the CDU—to mediate the settings of radio links and of waypoints. The helicopter has several radios that the pilot can use for communication with other named stations. The CDU can be used to set the frequency, mode, and security of each radio link, and to set a list of waypoints (a waypoint is an x–y position along the flight path that the pilot wishes to follow; waypoints may also have frequency values and names, since they may correspond to navigational beacons). The designer must at least provide protocol nodes that handle messages relating to these two functions, and that can distinguish between messages related to the radios and messages relating to the waypoints.

The designer could choose to provide either one top-level protocol node to handle messages of both kinds, or two independent top-level protocol nodes, one for each kind of message. Whichever choice is made, the designer must ensure that somewhere in the protocol hierarchy the CDU can perceive which kind of setting is not satisfactory to the pilot. It could be done in a "mode-switching" protocol that accepts messages to say what kind of setting is to be modified and then diverts subsequent messages to the appropriate protocol node, or it could be done in what Taylor and Waugh (1999, forthcoming) call "broadcast mode", by ensuring that radio-setting messages have a form different from the form of waypoint-setting messages. However it is done, the complete design must link the top-level perceptions to the preset low-level messages (keystrokes or voice) in such a way that all the perceptions that should be controlled can be influenced by available actions, preferably without those actions also disturbing other independent controlled perceptions.

Step 2 requires a choice of a single protocol node to design. As an example, we will choose a node that handles waypoint-setting messages.

When LPTool opens, it displays the interconnections of all the protocol nodes so far created, in a view of the design we call the "Network View". If this is a new design, the Network View is blank, and an initial node must be created. The designer does this by selecting the icon of either a receiving or a transmitting node from a palette, and dragging it onto the design space. Once the icon of a new protocol node is placed in the design space, it must be named. In our example, we will arbitrarily name this first node "Waypoint setting".

The icon of any protocol node in the Network View serves as a set of active links to generally useful views of the internal structure of the node. One of these views is onto the Model component of the node (which includes the specification of the lexicon of Primal Message types that the node can handle). Among the other views linked through the icon are views onto the Coder and Decoder of the node, and onto its General Protocol Grammar. Staying within the Network View, the node can be linked to other nodes at higher or lower levels of the hierarchy by clicking on the Model element of the lower icon and either the Coder or Decoder of the higher, depending on whether the lower-level node is a transmitting or a receiving node, respectively.

Step 3: *Define the lexicon of perceptions the originator* (*human or machine*) *might want to control* (*i.e. define the Primal Message types the chosen Protocol Node will handle*).

From the icon in the Network View, the designer access a view onto the node's Model. The Model in every receiving protocol noded should be able to compare the pilot's satisfaction with the reference value "completely satisfied". Because this is an element of every protocol node, the designer does not need to specify it explicitly. Likewise, the designer need not specify that the Model in a transmitting protocol node should be able to compare the recipient's interpretation of the Primal Message with its reference value, since all transmitting protocol nodes must have this capability.

The Model component of a protocol node includes a list of the messages or message types that the node can handle. We call this set of lexical items the node's *lexicon*. A lexical item for a protocol node in verbal communication between humans could be a word, a gesture, or a phase of an argument, depending on the protocol node and the level of abstraction. It might be the value of a continuous variable, such as the pitch of a voice or (at a higher level of abstraction) the degree of irritability of the talker. For the CDU a lexical item might be a display, a field within the display, a frequency within the field, a number within the frequency, and so on.

In the prototype LPTool, the node's lexicon appears only as a plain-text annotation of the node's Model, but in a fully functional version it might appear as a formal grammar, as a program, as access to a database or in some other active form. In the Farrell–Semprie procedure, the designer should specify this lexicon before proceeding to other elements of the design.

We will concentrate the present discussion on a single kind of Primal Message at a high level in the dialogue hierarchy: the CDU wants to believe that the pilot is satisfied with the settings of the waypoints. This protocol node is therefore a receiving node for which the Primal Message is a waypoint setting. The interface designer specifies this very simply. In a text-based "Annotation View" onto the Model for the "Waypoint Setting" node, the designer types "Settings for a single waypoint". The node handles only this kind of Primal Message content.

In most cases, the Model consists of more than just a lexicon of Primal Message types. It also contains any information available to the node in support of its function of interpreting or constructing messages across the interface. In particular, this includes the possibility of anaphoric references and the ever-changing state of each partner's beliefs about the recipient's understanding of the Primal message. For example, in a waypoint setting, if the CDU believes that the waypoint being set is "Badgerton", the possibility that it can remember the name for future reference is an aspect of the Model that must be included in the design, and potentially made perceptible to the pilot.

The Decoder will use the Model to aid its interpretation of an incoming message. The reason we think of such information as being in the Model rather than in the Decoder is two-fold. Firstly, we think of the Decoder as process rather than data, and secondly, the Coder will use much of the same data in its own process of constructing virtual messages directed to the other partner.

The Model may be able to do much more than just store data and compare current beliefs with their reference states. It may be able, for example, to determine the acceptability of a waypoint specification within the context of a mission. Whether the Model in

any specific protocol node has such additional capabilities is a matter for the designer to determine.

Because the protocol node being designed handles messages about waypoint settings, its Decoder must at least be able to interpret the specifications of a waypoint and to assess the syntactic acceptability of the specification. For example, the input might have been "Waypoint 5 should be set to HPT GPS N 60 59.30/W073 30.00", but the same information might have been provided by the spoken words "Next Waypoint Badgerton".

To specify the various structures that can constitute a legal waypoint specification, the designer using LPTool accesses an annotation view onto the Decoder. Whereas the Model annotation can be very simple, the decoder annotation must be specific enough to ensure that all possible ways to specify a waypoint will be properly interpreted. It can, of course, refer to any anaphoric capability the designer has specified for the Model, such as, for example, that the "Next" waypoint is "Waypoint 5". Again the existing LPTool prototype allows only plain-text annotation, but the designer is expected to enter a proper specification of a waypoint, which might include that the waypoint has a geographic location, a name, an optional frequency and any other relevant characteristics.

In addition to the Model and the Decoder, the "Waypoint Setting" node must incorporate a Coder that can specify for output display the characteristics of a waypoint, so that the pilot can see how the CDU is interpreting her input. The designer specifies the encoder in the same way as the other components, by a text annotation in the prototype LPTool, but a more functional description should be part of a fully realized LPTool.

Once a waypoint is established to the pilot's satisfaction—i.e. the Primal Message appears to have been adequately interpreted—the CDU passes the waypoint specification to onboard computers for use during the mission. The interface designer is not concerned with how this transfer is done, but is concerned with assuring that the pilot will be confident that the specification being passed is indeed the specification the pilot wants.

Step 4: *Annotate completely the GPG for the chosen node before generating and annotating supporting nodes.*

The GPG is the core of the design of each Protocol Node. It requires the most time and effort from the designer. The GPG is concerned not with the actual content of the Primal Message, but with the controlled perceptions of the partners relating to the progress of the Primal Message interpretation. In LPTool, a view onto the GPG can be accessed directly through the Network View icon, or from various other views onto the design. We discuss the details of the GPG in the companion paper, and will not repeat them here. In simple form, however, the GPG consists of four stages through which a message "passes" in the process of interpretation, not all of which may be implemented in any specific protocol.

1. *E-feedback*. The potential receiving node displays to the potential originating node its current state of receptivity. For example, if the "Waypoint Setting" node were in a state to receive a new message, it would attempt to display that information in a way that would contrast with a display showing it to be busy already. If the CDU

had a modal interface, requiring it to be in "waypoint" or "radio" mode, this display might conflict with a similar display by the radio-setting protocol. The designer would have to ensure that only one of the two "ready" displays actually was presented to the pilot. But that possible conflict is irrelevant when the designer is considering the waypoint-setting node itself. It is an issue only for the lower-level node onto which both displays are multiplexed. What the designer of every node does have to decide is what E-feedback is to be provided, if any (Engel & Haakma, 1993; Engel, Goosens & Haakma, 1994).

2. *Opening* (*Primary*) *message*. The opening message (called the Primary message in other writings on LPT) has the following two functions.

   (a) It disturbs the recipient's perception that the originator is satisfied, which in a cooperative dialogue probably leads to some action by the recipient to restore that perception to its reference level.

   (b) It may provide some or all of the content of the Primal Message for the recipient to interpret. If it provides all of the content, and if the interpretation is so trivial and assured that the recipient will perceive the originator to believe that the interpretation will have been correctly made, then the Primal Message's disturbance to the recipient's "originator satisfaction" perception may vanish when the interpretation is made. More commonly, however, either the recipient is not assured that the initial interpretation would satisfy the originator, or is not assured that the originator automatically perceives the interpretation to be satisfactory. In this more common situation, the recipient's action to restore the perception "the originator is satisfied" leads to stage 3.
3. *Convergence*. The recipient remakes and continually modifies an interpretation of the Primal Message, until the originator is perceived to be satisfied with the interpretation. This phase may be trivially simple, even null if the originator trusts that the opening message will be sufficient to ensure a correct interpretation, or it may be extraordinarily complex, if, for example, the Primal Message is that the recipient should understand perfectly the *Principia Mathematica*. Most communications fall somewhere in between. The GPG is largely concerned with the details of the Convergence process, which depends on the interactions among the complex beliefs that are composed of nine controlled perceptions in each partner plus, for the originator, the controlled perceptions involved with the Primal Message content.
4. *Completion*. Message transmission is completed either by success, meaning that all the relevant perceptions have come to their reference values, or by failure, meaning that at least one partner no longer cares enough to continue acting to improve the recipient's interpretation, even though the interpretation still fails to match the originator's goal. The GPG includes both possibilities, under the labels "Commit" and "Abort". The completion phase consists of each partner assuring the other that they believe the communication of the Primal Message to be terminated.

In these four stages, the detailed GPG describes 23 combinations of belief states, and 47 different kinds of message that may be transmitted in one direction or the other to affect those belief states. In the design process for a particular protocol, the designer must determine for each of the 47 kinds where it will be used. If it is, the designer must ensure that a lower-level protocol node exists that can handle it as a Primal Message.

In most protocols encountered in human-computer interaction, many of the 47 possible message types will not be used. Sometimes only the opening message is used (as in a protocol for the input of a single character on a touchscreen), sometimes the opening message and a return message equivalent to "OK, I got that" (as, for example, if a beep occurs when a character is detected as having been input on the touchscreen), and sometimes a full panoply of correction and questioning messages is used (as when the message is the architectural design of a house). The designer must decide for each protocol whether each of the 47 possible messages is to be available. The Farrell–Semprie procedure suggests that all such decisions should be complete before the design moves on to the next stage.

Step 5. *Ensure that Coders are connected to transmitting nodes at the level below.*

Step 6. *Ensure that Decoders are connected to receiving nodes at the level below.*

Steps 5 and 6 can be considered together, although it has been found best to deal with them in order during the design. The LPTool's Network View shows the various protocol nodes and their links. In Step 4, the GPG was used to decide what messages might need to be passed in each direction for E-feedback, opening message, convergence, and completion. Step 5 makes sure that the designer provides protocol nodes to implement the messages outgoing from the viewpoint partner, and Step 6 does the same for messages incoming from the other partner.

For example, the CDU may be required to let the pilot know that a setting that has been input to the "Waypoint Setting" protocol node is not interpretable within the mission context because it is out of flying range from the previous waypoint in the list. If so, then there is a message within the GPG that conveys the concept "Illegal waypoint for this mission; Waypoint out of flying range". Such a message would be in the Coder lexicon, and the designer must ensure that the Coder is connected to a transmitting protocol node that can handle Primal message of this type.

The Farrell–Semprie design procedure requires the designer to create protocol nodes for all the Primal messages the GPG has shown to be required as outgoing from the viewpoint partner, and then to create protocol nodes for all the Primal messages input to the viewpoint partner. Very often, all or most of the outgoing messages will use the same lower-level transmitting node, and all or most of the incoming messages will use the same lower-level receiving node. Rarely are more than two transmitting and two receiving nodes required to support any one protocol node, and often a node that supports one higher-level protocol node will also support another. At the lowest level, the keyboard and (possibly) the vocal input nodes support all the others.

Since at least some low-level protocol nodes must multiplex messages for more than one higher-level node, the designer must choose for each whether this multiplexing is to be time-division (creating a modal interface) or broadcast. Broadcast multiplexing (or a modeless interface) may be inappropriate for pilot-CDU interaction where information must be displayed as explicitly as possible, but the possibility is open to the designer to decide whether to allow or to prohibit it at each level where multiplexing occurs.

Step 7: *Repeat steps* 2 *to* 6 *until the analysis or design is complete.*

Eventually, all the top-level Primal Messages will have been accommodated by completely designed protocol nodes, and all the messages specified by the GPG will have

been implemented in the prescribed hardware. At this stage, the functional design of the interface is complete, and what remains is to implement the design by coding the functions that the designer has determined to be required.

A further stage, in a future fully functional version of LPTool, would be to link the messages into a simulation, so that the simulated CDU could be used by a "test pilot". The Primal Message "navigation" in the simulation would cause the "test pilot" to change the settings in the display. This capability would allow the LPTool to serve as a fully fledged prototyping and testing environment.

4.2. THE PROCEDURE IN PRACTICE

The Canadian Forces helicopter community found that the current CDU interface was awkward to use and did not provide the necessary feedback to establish with any confidence that a radio link had been set. The current CDU interface was analysed using LPT to confirm and discover underlying deficiencies within the pilot–CDU interaction. It was hypothesized that the deficiencies would be apparent within the interface of the unit, and a solution would be proposed that addressed the interaction deficiencies.

Farrell and Semprie (1997), using the LPTool, reported on a Layered Protocol model that investigated the interactions supporting the Primal Message, "I want to see that a radio link is established." This high level Protocol was completely annotated, and the network of supporting protocols was explored. A list of deficiencies was generated by determining all the required virtual messages and checking whether they were instantiated in the current interface design. One of the most surprising deficiencies was the lack of an explicit display that indicated if a radio link was established (a top-level virtual message). Without such a display, the pilot would have to attempt to keep the individual elements in memory and surmise whether the current state of the link is satisfactory, and ultimately talk over the radio link before he/she could confirm that a radio link had been established.

Having analysed the existing interface, Farrell and Semprie proposed a new interface that addressed the deficiencies and supported all the virtual messages at all levels of abstraction. A sketch of a screen display during the setting of a radio frequency is shown in Figure 9.

One principle followed by Farrell and Semprie was that the designer should always have in mind the kinds of belief states desired by the user and the machine (the Primal Messages of their transmitting nodes). Typically, the designer will start by discussing this problem with representative members of the set of target users (e.g. Marken, 1999, this issue), to find out not what they want to do, but what they want to achieve under different task conditions. The procedure is reminiscent of orthodox task analysis, but the concentration on the user's controlled perceptions gives it a rather different flavour.

Not only must the machine and the user be able to interpret each other's Primal Messages, but also the machine must be able to determine whether the user is satisfied with its interpretation of the user's Primal Messages. This important and sometimes neglected aspect of design is the main topic of the companion paper on the GPG (Taylor *et al*., 1999, this issue). Here, it suffices to say that the designer must decide whether to

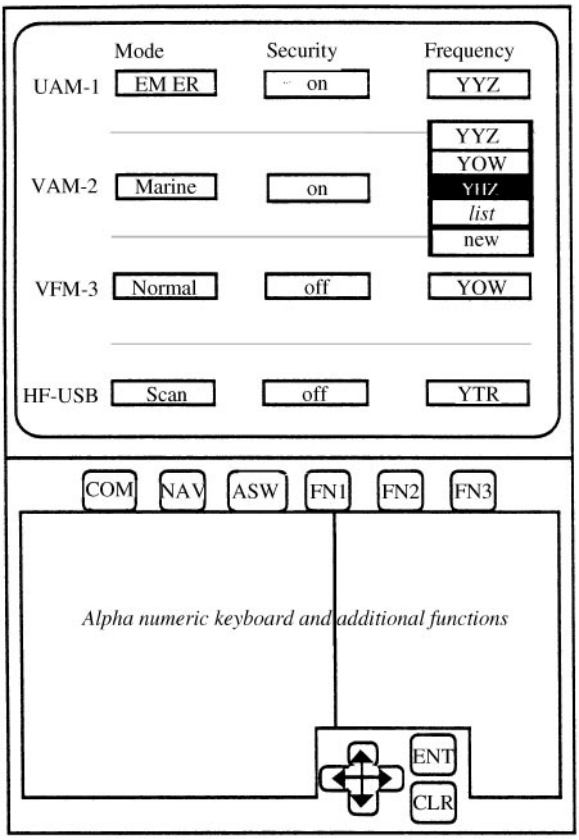

FIGURE 9. One possible layout for the CDU interface that incorporates on a single screen and one nested menu structure all the necessary information for establishing a communications link (Farrell & Semprie, 1997).

provide opportunities for the user to edit the machine's interpretation of each message before it changes, say, the waypoints used by the navigational computer. This is the "convergence" phase of the GPG.

An initial state of the CDU might contain a set of current or previously entered waypoints. The desired state (or primal message) is to perceive a particular waypoint set. Virtual messages are transmitted between the CDU and the pilot to alter the pilot's belief about the CDU state from their initial to their desired state and keep them there. Once all the parameters have been set, they presumably retain their settings until the pilot alters them, but the designer should consider whether the pilot is likely to remember them. If so, then they need not be redisplayed, once set, but if the pilot may forget, then at least an optional redisplay should be provided.

If an interface has modes, the designer must make them distinguishable. For instance, the radio and waypoint belief states have some aspects in common. Each might be perceived by the pilot as having a name; each can have an associated frequency, and so forth. A plausible protocol to support either might be one through which the pilot can set

a frequency. Would this frequency be used to set a radio or waypoint parameter in a higher level protocol? In a modal interface, as in the present CDU, the problem is resolved by directing all messages either to radio setting or to waypoint setting, depending on the current mode. The existing CDU does this by restricting the user to move through two separate, but similar menu structures after the pilot sets the mode by sending an explicit message to the CDU that any following messages are for the specified device. The pilot has to remember that the CDU state is "Radio-relevant" or "Waypoint-relevant", which can be difficult when using one of two menu structures that have much in common. (One is reminded of an early computer game in which one of the locations was described as "A maze of twisty passages, all alike".)

The revised interface by Farrell and Semprie (1997) retains the modal character, in that the pilot must specify whether messages are going to be radio-relevant or waypoint-relevant. But instead of simply requiring the pilot to set one parameter after another by means of a succession of menus that have much in common, the display presents the current state of either the radios or the waypoints (Figure 9). Rather than having to rely on memory, the pilot can not only see whether messages are currently radio-relevant or waypoint-relevant, but also can both see and adjust the current state of all the radio or waypoint settings. The "maze of twisty passages" is replaced by a map through which the pilot navigates easily.

The Farrell–Semprie interface multiplexes the radio and waypoint messages onto the single display screen in two ways. At the top level, waypoint setting messages are distinguished from radio setting messages by being on the screen at different times. At that level, the interface is modal. But within either the radio-setting mode or the waypoint-setting made, all the messages are displayed simultaneously, using broadcast multiplexing. The interface is modeless with respect to these protocols. At the level of "wanting to establish a radio link", the virtual messages expected by the CDU are explicitly displayed as rows within the matrix (E-feedback). At the level of "wanting to establish a mode, or frequency, or security setting", E-feedback shows an element within the matrix, identified by its location in a column of the display. At the level of "wanting to enter a frequency call letter" the operator needs only to select an item from a pull-down menu. The display provides information about the desired state, the current state, and the states most likely to be desired for the elements, as well as identifying the belief structure to which the element belongs.

The screen protocol includes with each message its location on the screen, which allows the messages to be distributed correctly to the protocols the screen protocol supports. Multiplexing at these levels is modeless, in contrast to the multi-level modal multiplexing of the hierarchic menus in the existing interface.

5. Conclusions

The user's controlled perceptions are the core of the human factors of an interface. The user's actions need be considered only insofar as they enable the user to control his or her perceptions. As the central maxim of PCT says: "All behaviour is the control of perception". The top-level perception, whose control drives the rest, is a perception associated with satisfactory completion of the task at hand. The output of a control system that controls this perception simply provides reference values for other

perceptions at lower levels. If those perceptions achieve their desired reference values, the top-level perception will have achieved its own reference value. The interface design has to be based on ensuring that the user can control all the perceptions related to the completion of the task, at all levels of abstraction. The software and hardware components of that interface are the peripheral tools for controlling those perceptions.

To apply PCT formally to an interface is impractical because of the great number of parameters involved in the many control loops, but to apply it informally can be fruitful. LPT, though developed independently of PCT, can be seen as an application of PCT to communication between partners. The simple "perception" of PCT is replaced by a complex "belief" in much the way that molecules replace atoms when moving from physics to chemistry.

This paper has presented an introduction to LPT as applied to interface analysis and design, in the context of a CDU through which a helicopter pilot sets and changes radio communication links and navigational waypoints. A prototype of a Layered Protocol analysis and design tool, called LPTool, was used. LPTool is intended to help the designer face and resolve difficulties in interface design before the product is implemented and made available to users. The various views it provides on the interface may help the designer to highlight issues that might well be missed in design methods that at first appear simpler. The apparent complexity of the design process may be compensated by a product that better serves the purposes of its user.

References

ENGEL, F. L., GOOSENS, P. & HAAKMA, R. (1994). Improved efficiency through I- and E-feedback: a trackball with contextual force feedback. *International Journal of Man-Machine Studies*, **41,** 949–974.

ENGEL, F. & HAAKMA, R. (1993). Expectations and feedback in user-system communication. *International Journal of Man-Machine Studies*, **39,** 427–452.

FARRELL, P. S. E. & SEMPRIE, M. H. (1997). *Layered protocol analysis of a Control Display Unit*. DCIEM Report no. 97-R-70.

HAAKMA, R. (1998) Towards explaining the behaviour of novice users. *Int. J. Human-Computer Studies*, **50,** 557–570.

MARKEN, R. S. (1998) PERCOLATe: perceptual control analysis of tasks. *International Journal for Human-Computer Studies*, **50,** 481–487.

NORMAN, D. A. (1988). *The Design of Everyday Things*. New York: Doubleday.

NORMAN, D. A. & DRAPER, S. W., Eds. *User Centred System Design*. Hillsdale, NJ: Erlbaum.

POWERS, W. T., MCFARLAND, R. L. & CLARK, R. K. (1960). A general feedback theory of human behavior: Part I, *Perceptual and Motor Skills*, **11,** 71–88. Part II, *Perceptual and Motor Skills*, **11,** 309–323.

POWERS, W. T. (1973). *Behavior*: *The Control of Perception*. Chicago: Aldine.

POWERS, W. T. (1978). Quantitative analysis of purposive systems: Some spadework at the foundations of scientific psychology. *Psychological Review*, **85,** 417–435.

POWERS, W. T. (1992). Learning and Evolution. In W. T. POWERS, Ed. *Living Control Systems II*, Gravel Switch, KY: The Control Systems Group.

ROBERTSON, R. & POWERS, W. T. (1990). *Introduction to Modern Psychology*: *The Control-Theory View*. Gravel Switch, KY: the Control Systems Group.

TAINSH, M. A. (1985). Job process charts and man-computer interaction within naval command systems. *Ergonomics*, **28,** 555–565.

TAYLOR, M. M. (1988*a*). Layered protocols for computer-human dialogue. I: principles. *International Journal of Man-Machine Studies*, **28,** 175–218.

TAYLOR, M. M. (1988*b*). Layered protocols for computer-human dialogue. II: some practical issues. *International Journal of Man-Machine Studies*, **28,** 219–257.

TAYLOR, M. M. (1989). Response timing in layered protocols: a cybernetic view of natural dialogue. In M. M. TAYLOR, F. NÉEL & D. G. BOUWHUIS, Eds. *The Structure of Multimodal Dialogue*, Amsterdam: Elsevier-North-Holland.

TAYLOR, M. M. (1993). *Principles for intelligent human-computer interaction*: *A tutorial on layered protocol theory*, DCIEM Report No. 93-32, Department of National Defence, North York, Canada.

TAYLOR, M. M., FARRELL, P. S. E. & HOLLANDS, J. G. (1998). Perceptual control and layered protocols in interface design: II the general protocol grammar. *Int. J. Human-Computer Studies*, **50,** 521–555.

TAYLOR, M. M., MCCANN, C. A. & TUORI, M. I. (1984). *The interactive spatial information system.* DCIEM Report 84-R-22, Department of National Defence, North York, Canada.

TAYLOR, M. M. & WAUGH, D. A. (1999a). Multiplexing, diviplexing, and the control of multimodal dialogue, M. M. TAYLOR, F. NÉEL & D. G. BOUWHUIS, Eds, *The Structure of Multimodal Dialogue* II, Amsterdam: John Benjamins (to appear).

TAYLOR, M. M. & WAUGH, D. A. (1999b). Dialogue analysis using Layered Protocols. H BUNT & W. BLACK, Eds. *Pragmatics of Language* (to appear). *Understanding Systems.*

THOMAS (1978). A design-interpretation analysis of natural English with applications to man-computer interaction. *International Journal of Man-Machine Studies*, **10,** 651–668.