PsychNology Journal, 2004
Volume 2, Number 3, 304 - 330

# Analysing interaction problems with cyclic interaction theory: Low-level interaction walkthrough

Hokyoung Ryu♠♦, Andrew Monk♣

♦ IIMS, Massey University
Auckland, New Zealand

♣ Department of Psychology
University of York, York, England

**ABSTRACT**

This paper aims to develop a brief interface evaluation method on cyclic interaction theory, allowing low-level interaction analysis, i.e., action-effect problems, effect-goal problems, and goal-action problems. It illustrates low-level interaction problems from everyday examples and, in turn, based on cyclic interaction theory a designer review method, the low-level interaction walkthrough, is introduced. The method is a modified version of cognitive walkthrough and the analysis focuses on the issue of direct concern to the practitioner who intends to identify low-level interaction problems in their design specification.

Keywords: *Cyclic interaction theory, low-level interaction walkthrough, mode, goal reorganisation, goal-action matching*.

Received 11 June 2004; received in revised form 16 November 2004; accepted 16 December 2004.

## 1. Introduction

When users interact with a user interface they do so in a piecemeal and iterative way. In a series of iterations, the user performs various actions towards achieving desired outcomes. This is a very effective way of interacting with systems particularly graphical user interfaces, which are the most important application of Human-Computer Interaction (HCI). In HCI this idea of cyclic interaction was introduced by Card et al. (1983) as their recognise-act cycle. Norman's (1986) seven stages model also envisages a cycle of interaction, says "the action someone takes leads to changes to the environment. These are evaluated with respect to, and in a manner conditioned by, the user's current goals. This evaluation leads to the reformulation of goals and further action, leading to a new state of the environment, and so on". However, neither Card et al., nor Norman makes explicit how effects of actions on the environment could

♠ Corresponding Author: Dr. Hokyoung Ryu
IIMS, Massey University, Auckland, New Zealand
Phone: +64 (0)9 4140800 ext. 9140
Fax: +64 (0)9 4418181
E-mail: h.ryu@massey.ac.nz

generate the subsequent iterations of interaction or how the context of interaction would make differences in the user's interaction.

Recent researches on the HCI theories, e.g., *situated action* (Suchman, 1987), *distributed cognition* (Hutchins, 1996), and *activity theory* (Nardi, 1997), have advanced the account of cyclic interaction in a broader context, emphasising how actions are informed, in taking actions, and how users configure the next step in the different contexts. Building on these advances, Monk (1998) and Wright et al., (2000) have re-established the account of cyclic interaction, making explicit low-level interactions between the user and the system at each point to provide a clearer understanding of cyclic interaction. While they might impose great efforts on modelling interactions, they provide what a process results in, as well as what triggers that interaction in a particular context. It is thus possible for the designer to build interactive versions of the design so as to assess the assumptions made or being made regarding the interaction between the user and the system. This paper shows cyclic interaction theory may be applied well to walk through a proposed interaction design, as cognitive walkthrough (Polson, Lewis, Rieman, & Wharton, 1992) and activity walkthrough (Bertelsen, 2004) do.

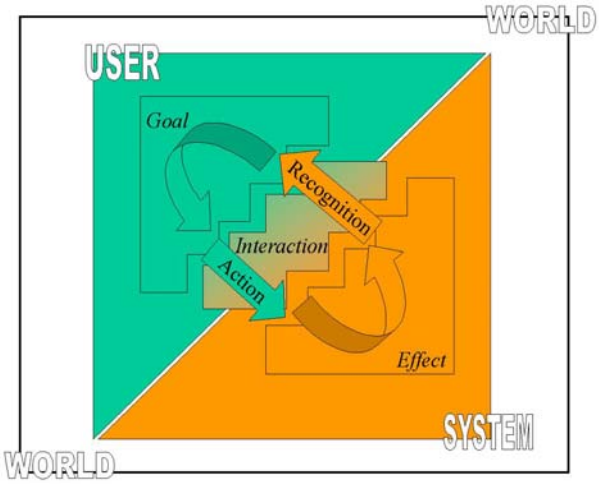



**Fig. 1:** Cyclic interaction theory. It illustrates a recognition-based interaction working on graphical user interfaces (GUIs). That is, action results from the user having some goal and recognition of the environment. The action leads to some effects on the environment. The new state of the environment (or world) is evaluated, leading to new goals and new recognition that in turn lead to new actions, and so on.

The key characteristics of cyclic interaction theory can be reasonably simplified as that of Figure 1, which depicts three paths in an interactive cycle: goal-action path, action-effect path, and effect-goal path. That is, the user begins an interactive cycle with the formulation of goals arising from the task or visible parts of the display in an interaction

context. The only way the user can manipulate the system is through an action, i.e., goal-action path. An action with the input device triggers system effects (action-effect path). The execution phase of the cycle is complete and the evaluation phase begins. The system is in a new state within the environment (or world), which must now be communicated to the user. The effect-goal path deals with changes in what are perceived and then continues to new goals in the interaction context.

Yet, this cyclic understanding of interactions between the user and the environment has been less applied to an analysis of user interfaces. This is because the models, e.g., State-Transition Scenarios (STS: Monk, 1999), based on cyclic interaction theory are more likely to provide a descriptive understanding rather than a formative perspective to analyse how the users tasks would be achieved in the course of interactions with the environment.

We do see that the walkthrough approach, e.g., Cognitive walkthrough (Polson et al., 1992), Activity walkthrough (Bertelsen, 2004), can make cyclic interaction theory operationalise for HCI practitioners, which is readily applicable for practical analysis of a design specification without real users in the evaluation stage.

In the following sections, we identify three classes of low-level interaction problems with cyclic interaction theory and develop a brief walkthrough analysis to detect them. Note that in this paper, we do not assume a specific interaction specification, but we will suggest how interaction specification of the designers' own use can be analysed in terms of cyclic interaction theory.

The three paths of interactions which are illustrated in Figure 1 in turn result in the three sets of interaction problem: *action-effect problems*, *effect-goal problems*, and *goal-action problems*. They have already been identified by the second author of this paper (Monk, 2000). We see that it may be an oversimplified assumption of all interaction problems, so this taxonomy would be very contestable with other classifications. Although it is open to dispute, we will show how this classification can provide many useful applications of the low-level usability check at the design stage.

Firstly, we will consider action-effect problems as unpredicted effects caused by user's action. That is, if the users are trying to take actions, expecting some effects in their head, however if the effects are not predictable, it results in action-effect problems. In turn, if system effects or environmental cues cannot generate any goals to be followed, the interaction would fail to proceed to the next action. The issues considered here are how the next goals are generated or eliminated by the system effects. This results in unpredicted goal-construction and elimination.

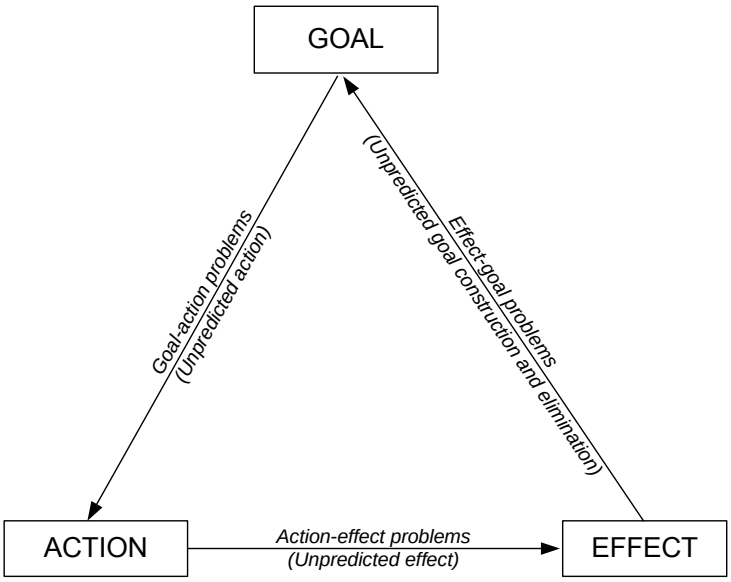



**Fig. 2**: Categorisation of interaction problems in terms of cyclic interaction theory.

Finally, the crux of the user interface design is how to assign physical actions on the system to achieve a particular goal or task. Therefore, if we imagine a user-friendly interface, most of the actions available to achieve any goal are very straightforward to find. Otherwise, the user is not able to match the correct action with the current goal state. This would be a goal-action problem, if the designer has not considered how the match between the goal and the action would be derived by the user. Figure 2 shows the three classes of interaction problems in terms of predictability.

In the following sections, the three sets of interaction problems will be illustrated with everyday practical examples and provide a brief walkthrough method to identify the interaction problems.

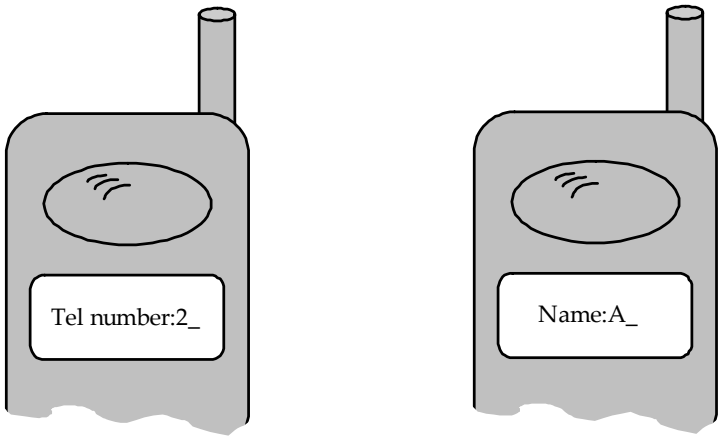



Fig. 3: Number-entry mode vs. letter-entry mode in a mobile phone. The user notices the current entry mode from the different prompts, i.e., 'Tel number' and 'Name' – Extended from Monk (2000).

## 2. Action-effect problems

Of all, an important issue to be analysed with respect to the action-effect path is the mode in which the same action leads to different system effects. Formally, this can be stated as the principle of action-effect consistency (Dix, 1991). That is, the consistent action-effect association reduces the effort needed to carry out a task and so makes learning by exploration much more effective.

Unnecessary modes in the interface should be avoided but sometimes they are inevitable. For instance, small devices like mobile phones employ the same action to perform various tasks. Thus, the action of pressing a particular button will lead to different effects depending on what mode the system is in. This moded interaction is less of a problem if a user is aware of what mode they are in (Monk, 1986). This is a noticeable mode. In normal calling mode pressing the button "2 abc" has the effect of putting the number '2' on the display; on the other hand, when the user edits their address book, pressing the button adds the letter 'A' first. They are less likely to feel confused about the different effects, because the mobile phone provides a clear mode signal using the prompt in the display (e.g., See Figure 3, 'Tel number:' for number-entry, and 'Name:' for letter-entry).

This character entry-mode example as shown in Figure 3 in a mobile phone demonstrates an instance where modes are signalled through the environment. Otherwise, mode errors are liable to occur. Mode errors, which give rise to action-effect problems, tend to occur where (i) *the mode change has been forgotten*; (ii) *there has been a failure to recognise the environment including information to indicate the mode*;

or (iii) *a misleading mode signal is perceived by the user*. The first possibility is a *hidden mode problem*, because the identification of the current mode depends on the recall of an earlier event rather than the recognition of external cues. The second possibility can be thought of as a *partially hidden mode problem* because of the relatively low salience of the mode signal given. One can attribute the third possibility to a wrong user model, which misleads users into believing that they are doing the correct action, i.e., *misleading mode signal*. Each of the mode problems is explored in the following sections.

### 2.1. Hidden modes

Most GUI guidelines (e.g., Apple computer, 1992) have emphasised a clear mode design, suggesting that all relevant information about modes should be noticeably presented in the environment. If the environment cannot signal the current mode, this causes the first type of mode error, i.e., *hidden mode*.

A typical hidden mode problem can be found in a UNIX `vi` editor. Mode errors in the Unix text editor `vi` – like many text editors, `vi` has a command mode, in which characters that are typed as input are interpreted as commands, and input mode, in which characters that are typed are inserted into the document being edited. Because there is (by default) no indication of which mode the editor is currently in, users often type in text thinking they are in input mode, but `vi` interprets their characters as commands because it is actually in command mode. This hidden mode issue can also be found in the L-0111 aircraft disaster (Vicente, 2004). The main cause of the disaster is that the autoflight system became disengaged as soon as a pilot inadvertently grabbed the control yoke, however, nobody noticed the aircraft was being manually controlled because there were no clear mode signals of whether it was in the manual or the automatic control mode.

For an everyday hidden mode example around us, see Figure 4. In the current New Zealand TV environment, in general, a television set has two different tuners: Terrestrial TV tuner (built-in tuner for each TV set) and Satellite TV tuner. When one wants to tune into a satellite TV channel from a terrestrial TV channel, one needs to change the mode from terrestrial to satellite, taking an appropriate action on the remote control. Yet, it is almost impossible for the user to know whether they are in satellite TV or terrestrial TV mode without any support from the environment as s/he is watching a TV programme. In the absence of a clear mode signal, the user has to recall what

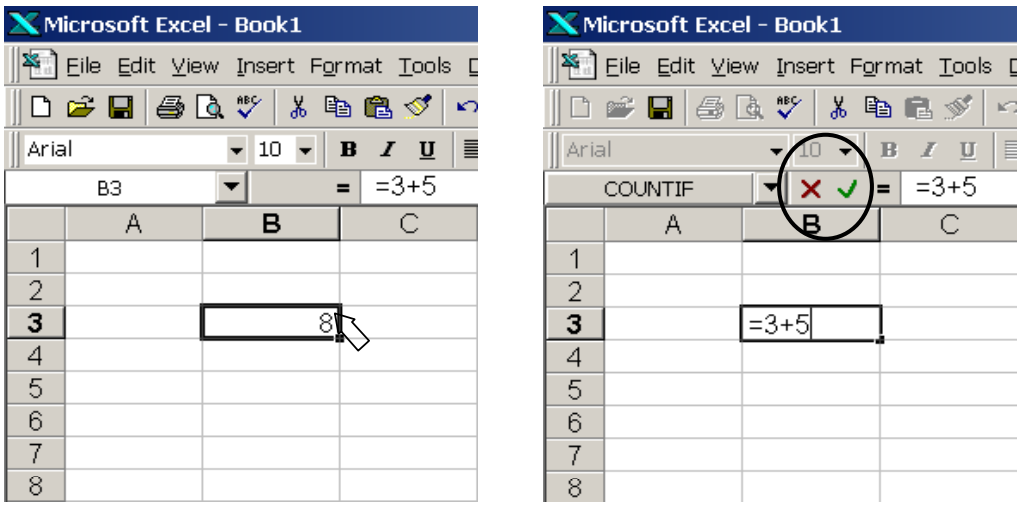
mode the TV set was last changed to. This is a hidden mode problem, because the user does not have any external cue to reason about the current mode or the mode reachable by the action.

**Fig. 4:** A satellite TV environment. It has at least two different tuners: one is a terrestrial TV signal tuner and the other is a satellite TV signal tuner. If the tuner selection is not easily recognised by the user, it may result in a hidden mode problem.

Whenever the mode change is likely to happen in any interaction situation, a designer should see if some recognisable signals of the current mode could be provided. In fact, some digital set-top boxes, for instance, signal the current mode explicitly by means of text or coloured icons in the display.

### 2.2. Partially hidden modes or poorly signalled mode

The second kind of mode ambiguity is where mode signals are not saliently designed. This is different from the first type of mode ambiguity, because mode cues do exist in the environment; however there is a difficulty in recognising them as mode signals.




**(a) Sheet-editing mode** **(b) Cell-editing mode**

**Fig. 5**: Two editing-modes in Microsoft Excel™ 97.

A poorly signalled mode is found in the Microsoft Excel™ 97 spreadsheet application (Dix, 2001). Excel™ 97 has two editing-modes: one is a cell-editing mode that enables the user to edit the contents of a single cell by, for example, adding a formula. The other is a sheet-editing mode that allows the user to edit or move sheets around. A mode ambiguity in Excel™ 97 occurs when the system changes the sheet-editing mode automatically into the cell-editing mode when users type anything whilst they are in the sheet-editing mode. Also, hitting the 'Enter' key returns the cell-editing mode promptly into the sheet-editing mode (Dix, 2001).

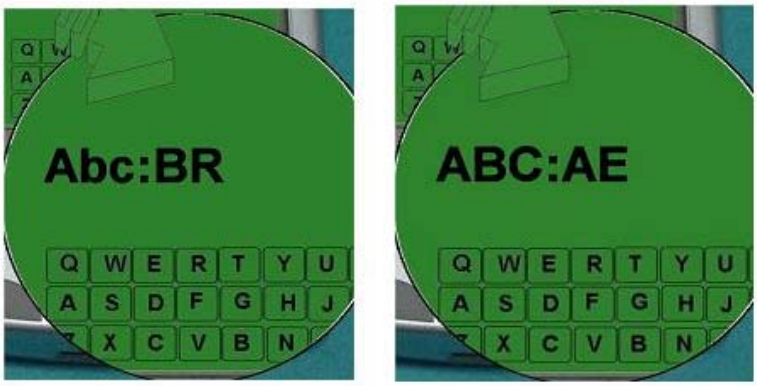
Consider the following situation from Dix: "if a user has selected a cell and can see the formula '=3+5' (see Figure 5(a)), the user may simply type '+2' thinking wrongly that they are in the cell-editing mode. However, this results in deleting the original cell contents rather than getting '=3+5+2' as probably expected. This mode error will be detected easily when the user is looking at the screen, and then they will try to undo the edit. However, if the user is looking back and forth to a paper list of numbers, it will be quite difficult to notice this error." See Figure 5(b). The application signals the current mode using only small icons, a cross (×) and a tick (√) in the formula box. The difficulty in recognising the current mode from the relatively low salience of signal (i.e., poorly signalled mode) may give rise to the partially hidden mode problem.

Whenever recognition of mode appears to occur, designers should ask themselves whether the mode signals provided are strong enough for mode changes to be

recognised. The partially hidden mode problem can be reduced by strongly signalled cues that enable users to reason about different outcomes of the same action. For instance, Dix (2001) suggests that a visual effect such as adding a very slight coloured tint over the spreadsheet would be effective to distinguish the two different modes in Excel 97.

### 2.3. Misleading mode signal

The third kind of action-effect problem may arise when the environment hinders the correct interpretation of the current mode. This commonly happens when mode signals are in conflict, in turn; it is ambiguous what the system status is in, at the time of interaction. It can account for the failure of highly moded interfaces, i.e., interfaces with many modes. In the aircraft flight circumstance, many accidents have been reported, which are caused by conflicting mode signals. For instance, consider the A320 Strasbourg disaster. Due to the confusing display of the mode reads 33 in the one mode and 3.3 in another, the aircraft descended at 3,300 ft per minute instead of 3.3 degree glide slope. That is, pilots selected the wrong mode of descent in a highly moded situation.




**(a)** **(b)**

**Fig. 6**: Various case mode signals (a) the first letter ‘B’ as a case mode signal, (b) both the prompt ‘ABC’ and the first letter ‘A’ imply the second letter ‘E’ will be upper-case.

Ryu and Monk (2002; 2004b) also carried out several mode experiments with respect to case-mode signals in hand-held devices. These experiments demonstrated that the users with the given information ‘Abc:A’ had more difficulty in learning the upper-case correct mode than those with ‘ABC:A’. It was argued that the poorer performance of ‘Abc:A’ to that of ‘ABC:A’ resulted from the fact that the prompt ‘Abc’ could imply that

the letters would be lower-case from the second letter on. See Figure 6(a). This shows that the designer employs the last letter 'B' as a case-mode signal for the case of the second letter. Yet, another plausible case-mode signal from the prompt, i.e., Abc, may turn to the user to reason about the case of the second letter as lower-case, instead. Therefore, whenever recognition of mode is likely to occur, a designer should ask himself or herself whether there are possibly competing mode signals.

*First phase: preparation*

- Step 1. Examine your interaction specification where the same action has different effects.
- Step 2. For each interaction specification where the same action has different effects, list system effects that may inform the user what the current mode is.

*Second phase: walkthrough*

To check mode problems, the questions are composed of three parts that will be answered in parallel, i.e., iteratively. The subquestions are interdependent because it is not possible to separate perception of modes. The three questions are partly redundant which helps the practitioners identifying more mode problems from the user's perspective.

- Q1. Hidden mode: Does the user recognise (rather than recall) the current mode from system effects?
- Q2. Partially hidden mode (Poorly signalled mode): Are system effects sufficiently salient for the user to discriminate the mode change from the previous interaction?
- Q3. Mode signals in conflict (Misleading mode signal): Is it possible that mode signals imply different modes?

*Third phase: walkthrough verification*

Finally, the mode problems identified from the walkthrough questions should be reviewed critically whether they are following mode design heuristics. Special attention is directed to how well the mode signal of the current interaction matches to the following heuristics:

- Do not rely on user's recall of mode changes.
- Provide mode signals with visibly or audibly salient signals in system effects, so that the user notices the mode change between interactions.
- Keep mode signals to indicate the same mode between interactions.
- Remove competing mode signals in system effects where there is more than one mode signal.
- Provide consistent mode signals across all tasks.

**Fig. 7**: The low-level interaction walkthrough of action-effect problems.

In summary, most systems have modes of one kind or another. This is a problem if the user is not aware of the contingency, i.e., if the mode is *Hidden* (no signal to the user); *Poorly signalled* (mode signal insufficiently salient to guide the user's behaviour); or *Inappropriate* (mode signal misleads the user). This classification accounts for why the user's perception of the current mode needs to be considered whenever there are modes.

However, these case-by-case analyses do not offer ready-made techniques and procedures. Furthermore, it must be concretised according to the specific nature of the technology-in-use under scrutiny. For this purpose, we do provide a walk-through procedure (or checklists), as depicted in Figure 7, in order to identify action-effect problems. Having written a complete interaction specification of some parts of a system the designer is asked to examine all relevant action-effect pairs where the same action has different effects. Having answered the questions in Figure 7, the designer will be satisfied that the assumptions made in the design are without mode problems. In this way, the designer can provide a credible argument for their proposed design based on the interaction behaviour of the user depicted in their design specification.

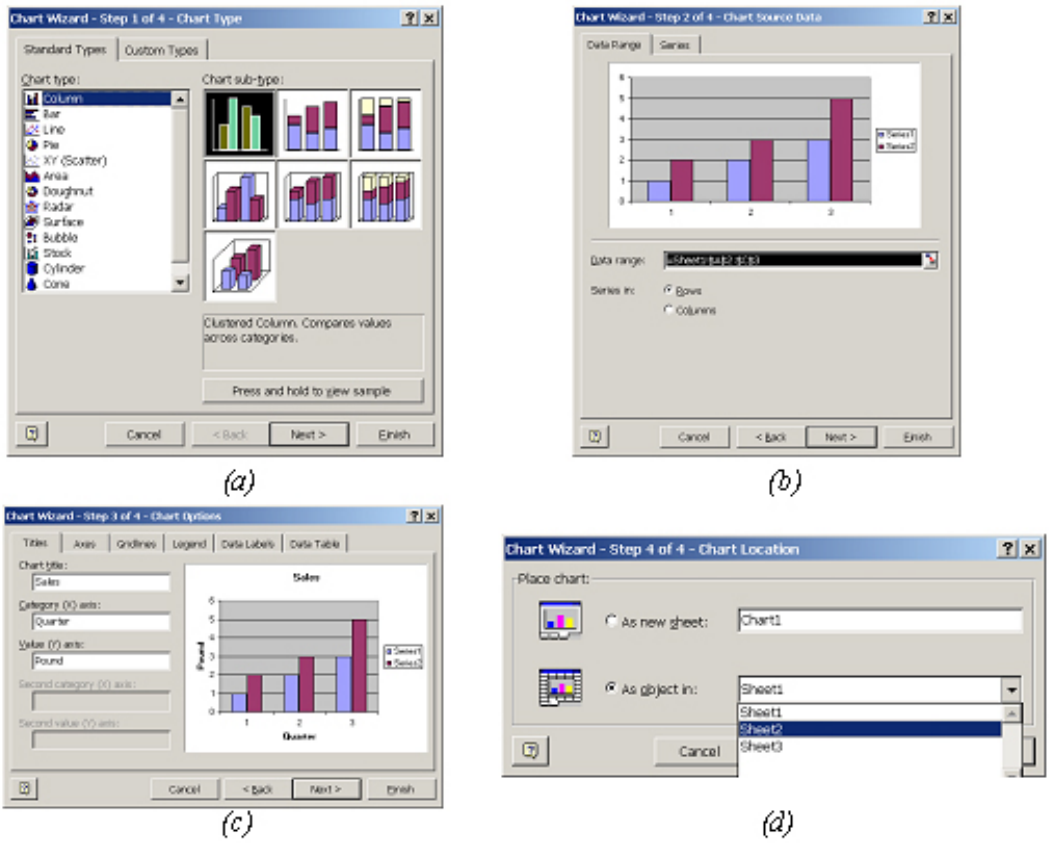



**Fig. 8**: Chart wizard from Microsoft Excel 97. (a) Step 1/4 – Select Chart type, (b) Step 2/4 –Select data to be referred, (c) Step 3/4 – Select chart options, and (d) Step 4/4 – Select chart location.

## 3. Effect-goal problems

In most HCI tasks, the way a user communicates with a system heavily depends on system effects. This is a very effective way of working and GUIs are arguably the single most influential application of HCI research. Of this, cyclic interaction theory explains that when visible and audible system effects are inadequate, the user will not be able to construct or eliminate appropriate goals. The only way to interact with such systems is to recall the correct task procedure. In turn, this kind of interaction results in more efforts to learn.

The effect-goal path in cyclic interaction theory can be effectively thought of as the goal reorganisation process. For instance, the chart wizard in Excel™ 97 helps users decompose a goal into a number of subgoals (Wright et al., 2000). In this way, they realise what has been achieved at each step and how many steps are left to accomplish the overall goal. In Figure 8, the controlled process of goal reorganisation in the chart wizard application is starting from a goal 'Create Chart'. It is very straightforward to imagine that the goal gives rise to the action 'Click Icon(Chart)' in Excel™. The environment in Figure 8(a) generates another goal 'Specify chart type'. The next interaction allows the user to eliminate the goal 'Specify chart type' and in turn initiate the subsequent goals 'Specify Chart subtype' in Figure 8(a), 'Specify data range on sheet' as in Figure 8(b), 'Specify X- and Y- axis' as in Figure 8(c), and 'Specify location' as in Figure 8(d).

Affordances (Djajadiningrat, Overbeeke, & Wensveen, 2002; Gibson, 1979; Norman, 1999) of each display help the construction and elimination of appropriate goals (Ryu & Monk, 2004a, 2004b). The main benefit of the wizard application results from the affordances of these objects on establishing and then removing subgoals appropriately. The wizard application example demonstrated the instance where the subsequent goals are naturally generated and the completed goals are explicitly terminated through the appropriate changes to the environment. Otherwise, effect-goal problems are liable to occur. An inadequate goal-reorganisation process thus can be classified into four categories as follows: (i) *missing cues for goal construction*– effects do not suggest appropriate goals; (ii) *misleading cues for goal construction*– effects suggest irrelevant goals; (iii) *missing cues for goal elimination*– effects do not delete completed goals; or (iv) *misleading cues for goal elimination*– effects delete incomplete goals. The first two possibilities may be followed by incorrect actions; the last two are likely to result in repeating attempts to achieve the goal.

### 3.1. Missing cues for goal construction

The user reasons about the subsequent goal from system effects, except when obviously given to the user. Thus all relevant cues for reasoning about the subsequent goal should be presented appropriately. The lack of appropriate cues may result in goal construction problems arising from *missing cues* or *misleading cues*. An extreme example of the former case is where cues or information are not presented in the system. The latter implies that inadequate cues are in the system, leading to irrelevant goals and, in turn, incorrect actions.

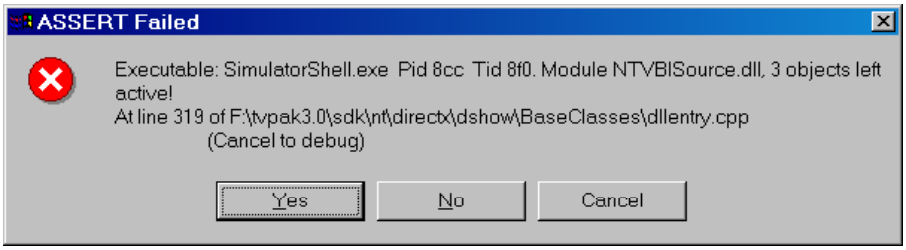


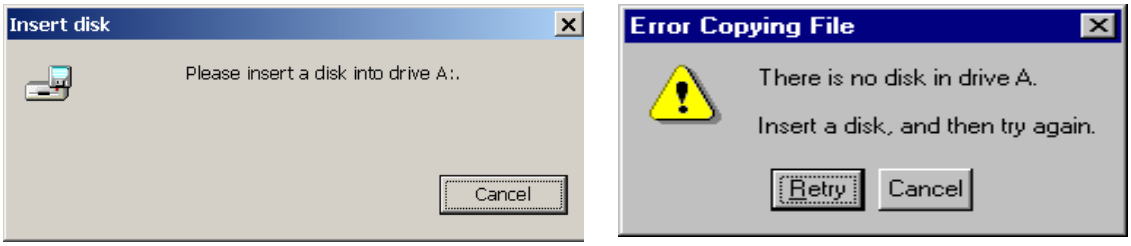

**Fig. 9**: An alert message in starting an application.

The first kind of goal construction problem refers to the instance where it is difficult for the user to reason about subsequent goals from system effects. In turn, it is less likely to take the subsequent correct action. An example of this kind of goal-construction problem can be found where a system generates an alert box when a user initiates an application as in Figure 9. A critical criticism of this dialogue box is that it is very difficult for the user to click the 'Yes' button to start the application, even though the 'Yes' button has been signalled as the default button. Indeed, this problem originated from the fact that an action indicating goal – i.e., Select 'Yes' – cannot be straightforwardly constructed from this system effect. They may be more likely to choose 'Cancel' as being safer. In fact, this will invoke the debugger causing the novice user further confusion. Whenever it is assumed that subsequent goals will be generated by system feedback, a designer should see if the goal could be added by the strong affordance of the system effect.

### 3.2. Misleading cues for goal construction

The second kind of goal construction problem is where system effects strongly imply irrelevant goals, thus leading to incorrect actions. This is common when the affordance

of a display (or object) is so strong that users instinctively establish inadequate subgoals that are related to the affordance of the display.




**(a)** **(b)**

**Fig. 10**: Copying a file onto a floppy disk in (a) Windows 2000™, (b) Windows NT™

For instance, consider the task of copying a file onto a floppy disk. In Windows™ 2000, when users have not inserted a floppy disk into the drive, a dialogue box as that of Figure 10(a) appears. In contrast, Windows NT™ employs an alert box like Figure 10(b).

It can be thought that the designer of Windows™ 2000 assumes that users will construct the subsequent goal, i.e., Insert disk, from the message 'Please insert a disk into a drive A'; however the designer overlooks a possibility that the strong affordance of the sole clickable object – i.e., the Cancel button – may suggest an irrelevant goal, i.e., Cancel first, that does not pertain to the overall goal. In contrast, Windows NT environment allows the user to plan appropriate subgoals from appropriate affordances of the display at the cost of an extra click (see Figure 10(b)). The affordance of the 'Retry' button and the message inform the correct sequence of actions such as 'Insert a disk, and then Click Retry'. In fact, we found that many users selected the 'Cancel' button instinctively in Figure 10(a), when asked to perform the task.

As stated above, this phenomenon is partly because the affordance of the 'Cancel' button is too obvious at the point of interaction. Polson et al. (1992) have also claimed that this problem arises from a dialogue box that cannot produce an 'and-then' goal such as would be required in the example shown in Figure 10(b). Here, the user has to 'Insert a disk' and then 'Retry it'.

*First phase: preparation*

- Step 1. Examine system effects in the system specification that are designed to construct the subsequent goals.
- Step 2. List the system effects and the subsequent goals.

*Second phase: walkthrough*

To check goal-construction problems, the questions are composed of two parts that will be answered in parallel, i.e., iteratively. The subquestions are interdependent because the two types of goal-construction problems are frequently co-existing. The two questions are partly redundant which helps the practitioners identifying more goal-construction problems from the user's perspective.

- Q1. Missing cues for goal construction: Do system effects strongly suggest the constructed goal?
- Q2. Misleading cues for goal construction: Do the other system effects suggest that the user conceive of goals that do not pertain to the overall goal?

*Third phase: walkthrough verification*

Finally, the goal-construction problems identified from the walkthrough questions should be reviewed critically whether they are following goal-construction design heuristics. Special attention is directed to how well the system effects matches to the following heuristics:

- Suggest next goals using comprehensive system effects (or strong affordance to imply subsequent goals).
- Inform the sequence of actions for the user to plan subsequent goals.
- Remove situations or system effects that can strongly suggest irrelevant goals.

**Fig. 11**: The low-level interaction walkthrough of goal-construction problems.

In summary, goal construction problems occur when the subsequent goal is ambiguous or irrelevant, i.e., if system effects do not suggest appropriate goals (missing cues); or, effects suggest irrelevant goals (misleading cues). These two problems tend to be followed by incorrect actions, thus they can be equally considered as effect-action problem. In any system specification, the two kinds of goal construction problems can be detected by examining all the effects which can construct subsequent goals and then determining whether all the goals pertain to the overall goal. Figure 11 describes a walkthrough approach to be applied for identifying goal-construction problems.

### 3.2. Missing cues for goal elimination

In addition to constructing subgoals, system effects in interactive systems remove any doubt about what the system is doing and how it is responding to the user's action. For example, where there is a long system delay a user needs to know whether the system is actually responding to their last action and how the command is progressing. That is,

system effects have to show whether the user's goals have been successfully completed or not. The lack of appropriate feedback results in goal elimination problems arising from *missing cues* or *misleading cues*. The former implies that feedback of goals completed is not presented in the system, therefore the user has to remember what they have completed. The latter suggests that feedback leads to the elimination of incomplete goals inadvertently.

The first kind of goal elimination problem refers to the instance in which completed goals are not eliminated from the current goal set due to the lack of clear system effects. This implicit feedback is also found in the DOS environment, when a user types 'del my.doc' to delete the word file, and the system responds with a new command-line prompt, if successful. The new prompt is feedback, however, one cannot identify whether the file was deleted. Here, the feedback is provided only with respect to the lower-level goal of typing a syntactically correct command. Consequently, in order for the user to eliminate the higher-level goal, i.e., delete a file, they have to refer back to the previous action taken – 'del my.doc'; or follow up with a DIR command to check the right file was deleted. In addition, the two experiments carried out by one of the authors (Ryu, 2003) demonstrated the importance of the notion of goal elimination process through feedback.

Whenever it is assumed that a goal will be eliminated in the interaction specification, a designer should see if the goal could be deleted by the system effect. The implicit feedback may not provide sufficient information that the goal has been achieved, in which case users will be compelled to take action to check whether the goals were completed.

### 3.3. Misleading cues for goal elimination

In an old automatic teller machine (ATM), many people used to leave their cash card in an ATM after withdrawing cash (Byrne, 1995). This error comes from the inappropriate task procedure, i.e., 'withdrawing money, and then taking out the card'. Indeed, as the users have cash in hand they believe that the overall goal (i.e., withdrawing money) has already been achieved and inadvertently leave their card in an ATM. This example illustrates the second kind of goal elimination problem. That is, a strong cue (i.e., cash in hand) before the overall goal is completed would eliminate the subsequent goal (i.e., take out the cash card).

A sensible way to remove this 'super-goal kill-off' phenomenon by misleading cues (Byrne, 1995; Wharton, Bradford, Jeffries, & Franzke, 1992) on an ATM is to modify the interaction procedure from 'Withdraw money → Take out card' to 'Take out card → Withdraw money' that is adopted in the present ATMs. As a consequence, the user cannot retrieve the money from an ATM until they take out the card.

*First phase: preparation*

- Step 1. Examine system effects in the system specification that are designed to eliminate the current goals.
- Step 2. List the system effects and the goals eliminated

*Second phase: walkthrough*

To check goal-elimination problems, the questions are composed of two parts that will be answered in parallel, i.e., iteratively. The subquestions are interdependent because the two types of goal-elimination problems are frequently co-existing. The two questions are partly redundant which helps the practitioners identifying more goal-elimination problems from the user's perspective.

- Q1. Missing cues for goal elimination: Do system effects sufficiently allow the user to recognise (rather than recall) that the goal has been achieved?
- Q2. Misleading cues for goal elimination (Super-goal kill-off): Do system effects prompt the user to eliminate the overall goal even though subsequent interactions are needed?

*Third phase: walkthrough verification*

Finally, the goal-elimination problems identified from the walkthrough questions should be reviewed critically whether they are following goal-elimination design heuristics. Special attention is directed to how well the system effects matches to the following heuristics:

- Provide clear feedback regarding completed goals
- Don't rely on user's recall of the previous action to eliminate goals.
- Rearrange interactions so that a higher-level goal is accomplished at the end of interactions.

**Fig. 12:** The low-level interaction walkthrough to identify goal-elimination problems.

In summary, goal-elimination problems can be classified into two categories: (i) implicit goal-elimination arising from missing cues; (ii) irrelevant goal-elimination from misleading cues. The two kinds of goal elimination problems can be detected by examining subgoals which are eliminated in each interaction specification and then determining whether all the elimination arise from system effects.

Figure 12 shows a walkthrough for tracking down goal elimination problems. Having followed these steps, the designer will find that the assumptions made for the interaction would not be reasonable because of the implicit goal-elimination process.

In conclusion, this section has discussed how one can identify four kinds of effect-goal problem. In fact, effect-goal problems make the user's task performance even worse. In the first category, i.e., the appropriate goal is not suggested, it is difficult to complete the task because the effect does not reduce the knowledge demands on the

user (Norman, 1988), so that the user may need to recall events in the past. In the second category, when inappropriate goals are suggested, the problems will divert the user from the overall goal and may lead to an action that is not relevant to the overall goal. In the third category, where the completed goal is not eliminated, there is a dramatic effect on the task performance in that it will compel users to redo the task. In the fourth category, where an incomplete goal is deleted, this is related to the super-goal kill-off phenomenon (Wharton et al., 1992). This may lead the user to prematurely believe that the ultimate goal has been achieved.

## 4. Goal-action problems

Early studies of interface design (e.g., Payne & Green, 1986; Young, 1983) have claimed that the adequate connection between goal and action is one of the most important design issues. In particular, Payne and Green (1986) have emphasised that similar goals should be accomplished by similar action sequences, establishing a predictable relationship between the goal and the action, i.e., goal-action matching.


**Fig. 13**: Ejecting compact disk in the old Macintosh environment (Mac OS B1-8.6).

An example of the goal-action matching problem can be found in an old version of the Macintosh desktop environment (e.g., Mac OS B1-8.6). Consider Figure 13. To eject a compact disk users had to drag the disk icon to the trash can. Novice users on the Mac environment, particularly familiar users of Windows™, may be wary of dragging their compact disk icon to the trash can icon to eject it, because this is the same way one

deletes a file. Even though the two goals are obviously different, the two actions are almost the same.

Indeed, the primary criticism of this Mac environment is the affordances of objects (Djajadiningrat et al., 2002; Gibson, 1979; Norman, 1999). That is, affordances of the trash can icon and the other icons (compact disk and file) do not tell any difference between the two goals, i.e., ejecting or deleting.


(a) (b)

**Fig. 14**. (a) Delete a file or folder in the new Macintosh environment (Mac OS X). In contrast, (b) the trash can icon is automatically changed into the eject icon as a floppy or a compact disk is dragging to the icon.

To some extent, this problem is eliminated in the new Macintosh environment (e.g., Mac OS X) by providing objects with different affordances. See Figure 14. In contrast to the previous Mac environment, the trash can is automatically changed into the eject icon as a floppy disk is dragged. This modification allows the user to construct two different kinds of goal-action matching in terms of the recognition of the objects: Eject→(Floppy→Eject icon), and Delete→(File→Trash can icon). At the same time the system remains consistent with previous Mac OS.

As the example above shows, goal-action problems are commonly observed on the occasions where unpredictable actions are designed to accomplish the current goal. The analysis considers the following unpredictable actions: (i) *the weak affordance of the correct action*; and (ii) *the strong affordance of the incorrect action*.

The first case results in the user being able to take any other action due to the weak affordance of the correct action (or the object designed for the correct action). The second case results in an incorrect action. Either possibility involves unnecessarily complex actions. That is, the first case may compel the user to look for documentation, the latter will require that the user has to redo the task.

### 4.1. Weak affordance of the correct action

The first type of goal-action matching problem refers to an instance where the affordance of the correct action is not clear. As a consequence, a user may make considerable efforts to find the correct action. The problem of the old Macintosh environment stated above can be explained by the 'Trash can' icon having a weak affordance as a cue for the correct action for ejecting a floppy disk.

Consider another working example where one wants to set the clock on a portable MP3 player (e.g., AIWA™ CDC-MP3). Figure 15 depicts the control panel of the MP3 player. By holding down the jog dial for two seconds, the clock (AM 12:00) appears in the display. In order to set the hour the user has to press button '▲', then rotate the jog dial to change the hour. In contrast, the user has to press button '▼' to set the minute, then rotate the jog dial. The goal-action matching seems to be very arbitrarily assigned. This way of interaction is very confusing as the '▲/▼' buttons normally afford increment or decrement.

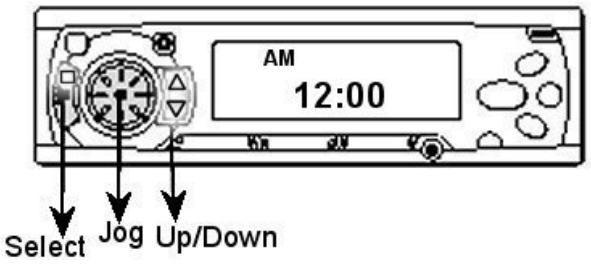



**Fig. 15**: AIWA™ CDC-MP3. In the clock-setting mode, the '▲/▼' buttons are used to initially indicate hour or minute, not for increasing or decreasing the figures on the display.

If the recall of the correct object for the relevant action fails to happen, the user either guesses and tries incorrect actions or looks for assistance from other resources such as the documentation (Mack & Montaniz, 1994). A sensible way to reduce this weak goal-action matching is to employ a consistent way of performing a particular task. For instance, in order to set the clock the user rotates the jog dial until the required time is displayed and then activates it by holding down the jog dial for a predefined time. This procedure may avoid a further recall of each object, thus supporting goal-action consistency as suggested by Payne et al. (1986).

### 4.2. Strong affordance of the incorrect action

The second case of goal-action matching problem refers to occasions where the affordance of the incorrect action is too strong, so that the incorrect action may frequently be selected. In order to avoid incorrect actions through strong affordance, for example, the common GUI environment deactivates certain menu options, thereby restricting the user from choosing incorrect actions.

Consider a working example where one wants to tune the radio waveband on a commercial audio player, AIWA™ RX 408. The player has a control labelled 'MODE' that one would think would tune the wave band. Yet, the control labelled 'MODE' only changes the mode from 'Tape Mode' to 'Radio Mode', and vice versa. Instead, a different control 'RADIO ON' changes the waveband between AM and FM. The difficulty in tuning the waveband arises from the inadequate label (i.e., RADIO ON), facilitating the selection of the incorrect control 'MODE'. The designer seems to consider that the affordance of control 'RADIO ON' would indicate more directly the radio function rather than control 'MODE' does. Yet, it is not so easy for the user to select the correct control 'RADIO ON', partly because the correct 'RADIO ON' button has a weak affordance to be selected, and partly because the incorrect 'MODE' button has a strong affordance.

A sensible way to reduce the difficulty is to use a different identifier for the 'MODE' button such as 'TAPE ON/OFF', thereby restricting the user to choosing the 'Mode' button to tune in the waveband.

*First phase: preparation*

- Step 1. Examine actions in the system specification that are designated for the current goal set.
- Step 2. Examine system effects in the system specification that indicate the actions.
- Step 3. List the actions, the system effects and the current goal set

*Second phase: walkthrough*

To check goal-action problems, the questions are composed of two parts that will be answered in parallel, i.e., iteratively. The subquestions are interdependent because the two types of goal-action problems are frequently co-existing. The two questions are partly redundant which helps the practitioners identifying more goal-action problems from the user's perspective.

- Q1. Weak affordance of the correct action: Can the user associate the action with the affordance of the corresponding object?
- Q2. Strong affordance of the incorrect action: Do system effects prompt the user to take an incorrect action from the strong affordance of the corresponding object?

*Third phase: walkthrough verification*

Finally, the goal-action problems identified from the walkthrough questions should be reviewed critically whether they are following goal-action matching

design heuristics. Special attention is directed to how well the actions on the system matches to the following heuristics:

- Avoid arbitrary and ambiguous goal-action matching.
- Don't assign any irrelevant goal to objects with other affordances.
- Provide strong affordance of the correct action

**Fig. 16**: The low-level interaction walkthrough of goal-action problems

In summary, it may be difficult for the user to find the correct action in a particular situation. This can be explained by goal-action problems. This is a problem if the user cannot correctly match the goal-action, i.e., if the correct action has an inappropriate affordance (no or weak indication of the effect of the action); or the incorrect action has strong affordance (the strong indication misleads the user).

To pinpoint the two kinds of goal-action problems in a proposed design, the designer must be able to provide a credible answer for their proposed design with the procedure as depicted in Figure 16. The procedure suggests that the two kinds of goal-action problems can be detected by examining the action in each interaction specification, and then determining whether the action can be triggered by the strong affordance of the display for the action.

This section has discussed how one can identify two kinds of goal-action problems with the notion of affordance. Goal-action mismatching accompanies complex actions. The problem of complex action is that users tend to forget the correct action. Therefore, complex actions should be avoided when the user may have insufficient knowledge to choose the correct action. For example, wizard applications (see Figure 8) will lessen the possibility of goal-action problems, because most work by the strong affordances of the display for the correct actions.

## 5. Conclusions and Discussion

This paper has developed an analytic framework of human-computer interaction, by setting out a way of thinking about cyclic interaction theory. The walkthrough approaches represented in Figure 7, 11, 12, and 16 were also developed to identify the classes of interaction problems according to cyclic interaction theory.

Cyclic interaction theory (Monk, 1998, 1999) has made it possible to envision interactions between the user and the environment in a relatively simple way. Three classes of interaction problems have been proposed and described. First, action-effect problems can be equally thought as mode problems in which the same action leads to

different system effects, categorising them into hidden mode problems, partially hidden mode problems, and misleading mode signals. Unexpected effects caused by inappropriate mode settings have been discussed. Second, effect-goal problems have been set out as goal-reorganisation problems, weighing up missing or misleading cues. Unpredicted goal construction and elimination caused by poor system specification has been described. It may be difficult or almost impossible to analyse these problems in that user goals are reorganised in the course of interactions and are not likely to be observable. Though it may be true to some extent, a specific assumption of a goal reorganisation process may produce detailed accounts that do match observed behaviour. Finally, two goal-action problems have been explained in terms of the concept of affordance. It has provided a possibility of understanding how the user would match their current goals with actions (or objects) in the system. Of course, one design error may result in interaction problems in several of these classes because the influence of each problem propagates rapidly through the interaction cycle. For instance, effect-goal problems may be detected where a user is expected to have difficulties in selecting the subsequent correct action, and that this could be equivalently considered as a goal-action problem.

Whilst this classification of interaction problems is no mutually exclusive nor exhaustive in explaining all interaction problems posed in the HCI research, they provide a brief base by which to contribute not only to an understanding of user's possible attitudes and responses to the system (or user interface), but also to the substantive HCI research seeking to understand specific interaction problems.

The low-level interaction analysis presented here has two different potential uses. One use is in the process of design. The other is in generating a modelling approach with the potential of extending HCI theory.

### 5.1. Cyclic interaction theory as a modelling tool

In this paper, there is no specific interaction specification presumed. This is partly because it is beyond the scope of this paper, and partly because different designers have their own preference for a specific type of interaction specification. For instance, they may describe interactions using state transitions or formal notations (e.g., Dix, 1987; Harel, 1988; Monk, 1999). More recently, some designers are very keen on using UML (Unified Modelling Language), which is a general-purpose notational

language for specifying and visualizing complex software, especially large, object-oriented projects.

Indeed, the main purpose of this paper is to apply cyclic interaction theory for identifying low-level interaction problems for an interactive version of their design, not to propose a modelling tool. It may result in a weak aspect of this paper in that there is no thorough mechanism specified with detailed explanation of each interaction specification. This paper sets out a research idea, for future study of models of cyclic interaction for improved this low-level interaction analysis. Further, at least the low-level interaction should be described on both the system and the user side at the same time and at the same level. That is, in order to interact with the system a user model must also generate low-level actions such as keystrokes and button presses. If this kind of model is developed, the analyses presented here can be further formalised (Ryu & Monk, 2004a).

### 5.2. Cyclic interaction theory as a design tool

From a theoretical point of view, the paper exploits only cyclic interaction theory, too easily clearing the findings of recent and important HCI theories that underline the situated and pragmatic nature of the human-computer interaction. Actually, we do not intend to trivialise other works. Instead we aim to provide a practical framework to evaluate low-level interactions with the practitioner, which we see what the cyclic interaction theory can provide.

In this context, the low-level account of user behaviour has a value in analysis of the potential interaction problems introduced at the design stage. For example, the mode detection walkthrough, given in Figure 7, makes it possible to reason about the cycles of interaction required to reduce mode problems in a highly moded interface. This analysis has a very similar purpose to the Cognitive Walkthrough (CW) analysis (Wharton et al., 1992) in that it also analyses interaction at the level of recognition and action cycles. In both cases the analysis focuses on the issue of direct concern to the designer, that is, identifying points in the human-computer interaction where the system may lead to inappropriate action, recognition or goals (Monk, 1999). However, sometimes following the steps in the CW is not simple. In such situations, the CW does not help the designer get insights into what is understandable and how things make sense from the users' point of view (Bertelsen, 2004). Thus, the simple question about visibility may be difficult to answer without detailed knowledge about how users

interpret what they see. By contrast, this low-level account of interaction problems can make explicit what problem a process results in, as well as what triggers that problem. It is thus possible for the designer to build interactive versions of the design so as to assess the assumptions made or being made regarding the interaction between the user and the system.

Hence, a main advantage of this paper is to provide the designer with the ability to evaluate their design when reasoning about new tasks and new systems. Any potential design could be checked against the assumptions that make explicit: the effects on the system of the different actions needed; what the user must perceive in the display; and the goals that have to be generated. To this end, the designer must be able to provide a credible answer to their proposed design as to why they assume the interaction behaviour of the user and the system as depicted in their system specification. If the answer to any one of these questions is negative, then this may indicate a potential low-level usability problem.

## 6. Future work

This paper provided a promising alternative to analyse low-level interactions in human-computer interaction tasks, simulating the user's behaviour. However, the user's behaviour cannot be independent of 'technology-in-use'. We see that technologies should be seen 'in use', inside activity settings meaningful to the user, in a broader context. Making hypotheses about the technology's use or simulating it, which is purported in this paper, without clarifying the conditions of such a simulation, brings to results of uncertain validity.

In this context, this paper may oversimplify the problems of the human-computer interaction without empirical data that support the conclusions and the walkthrough approach. For this purpose, we are currently working on establishing empirical understandings of this work more thoroughly.

## 7. References

Apple computer. (1992). *Macintosh Human Interface Guidelines*. Reading: Addison-Wesley.

Bertelsen, O. W. (2004). *The activity walkthrough: an expert review method based on activity theory.* Paper presented at the NordiCHI, Tampere, Finland.

Byrne, M. D. (1995). *A Working Memory Model of a Common Procedural Error* (GIT-CS-95/06). Atlanta, GA: Georgia Institute of Technology.

Card, S. K., Moran, T. P., & Newell, A. (1983). *The Psychology of Human-Computer Interaction*. Hillsdale, NJ: Erlbaum.

Dix, A. J. (1987). *Formal Methods and Interactive Systems: Principles and Practice.* The University of York, York.

Dix, A. J. (1991). *Formal methods for interactive systems*. London: Academic Press.

Dix, A. J. (2001). *Excel mode error* [Personal Web]. Retrieved 11, 05, 2004, from the World Wide Web: http://www.comp.lancs.ac.uk/computing/users/dixa/casestudy/excel-mode/

Djajadiningrat, T., Overbeeke, K., & Wensveen, S. (2002). *But how, Donald, tell us how? on the creation of meaning in interaction design through feedforward and inherent feedback.* Paper presented at the DIS 2002, London.

Gibson, J. J. (1979). *The ecological approach to visual perception*. Boston, MA: Houghton-Mifflin.

Harel, D. (1988). On visual formalisms. *Communications of the ACM, 31*, 514-530.

Hutchins, E. (1996). *Cognition in the Wild* (Second ed.). Cambridge, Massachusetts: MIT Press.

Mack, R. L., & Montaniz, F. (1994). Usability Inspection Methods. In R. L. Macks (Ed.), *Usability Inspection Methods* (pp. 295-339). New York: John Wiley.

Monk, A. (1986). Mode errors - a user-centered analysis and some preventative measures using keying-contingent sound. *International Journal of Man-Machine Studies, 24*(4), 313-327.

Monk, A. (1998). Cyclic interaction: a unitary approach to intention, action and the environment. *Cognition, 68*(2), 95-110.

Monk, A. (1999). Modelling cyclic interaction. *Behaviour & Information Technology, 18*(2), 127-139.

Monk, A. F. (2000). Noddy's Guide to Consistency. *Interfaces*(45), 4-7.

Nardi, B. A. (Ed.). (1997). *Context and Consciousness: Activity Theory and Human-Computer Interaction*. Cambridge, MA: The MIT Press.

Norman, D. A. (1986). Cognitive Engineering. In S. W. Draper (Ed.), *User Centered System Design* (pp. 29-61). Hillsdale, NJ: Lawrence Erlbaum.

Norman, D. A. (1988). *The Psychology of Everyday Things*: Basic Books.

Norman, D. A. (1999). Affordance, conventions, and design. *ACM Interactions,* 38-42.

Payne, S. J., & Green, T. R. G. (1986). Task-action grammars: a model of the mental representation of task languages. *Human-Computer Interaction, 2*, 93-133.

Polson, P. G., Lewis, C., Rieman, J., & Wharton, C. (1992). Cognitive walkthroughs - a method for theory-based evaluation of user interfaces. *International Journal of Man-Machine Studies, 36*(5), 741-773.

Ryu, H. (2002). *Will it be upper-case or will it be lower-case: can a prompt for text be a mode signal?* Paper presented at the CHI 2002, Minneapolis, MN.

Ryu, H. (2003). *Modelling cyclic interaction: an account of goal-elimination process.* Paper presented at the CHI 2003, Ft. Lauderdale, FL.

Ryu, H., & Monk, A. (2004a). *An interaction model: from a user model to an environment model.* Paper presented at the OZCHI, Wollongong, Austrailia.

Ryu, H., & Monk, A. (2004b). Will it be a capital letter: signalling case mode in mobile devices. *Interacting with Computers*, Revised.

Suchman, L. A. (1987). *Plans and Situated Actions: The Problem of Human-Machine Communication*. New York: Cambridge University Press.

Vicente, K. (2004). *The Human Factor: Revolutionzing the Way People Live with Technology*. New York: Routledge.

Wharton, C., Bradford, J., Jeffries, R., & Franzke, M. (1992). *Applying Cognitive Walkthroughs To More Complex User Interfaces: Experiences, Issues, And Recommendations.* Paper presented at the CHI.

Wright, P. C., Fields, R. E., & Harrison, M. D. (2000). Analyzing human-computer interaction as distributed cognition: the resources model. *Human-Computer Interaction, 15*(1), 1-41.

Young, R. M. (1983). Surrogates and mappings: two kinds of conceptual models for interactive devices. In D. Gentner, and Stevens, A.L (Ed.), *Mental models*: Hillsdale.