18

Dialogue

Interaction may be viewed as a dialogue, that is, a conversation that occurs between two partners in a context for some purpose. This view draws on what we know about human–human communication and, in particular, how conversations are structured; see Chapter 9 for a summary. In human–computer interaction (HCI), the communication partners are an interactive system and the user of such a system. HCI researchers have developed a rich palette of theories to understand such dialogues. These theories explain what happens in dialogue and how it shapes the relationship between the partners. These theories also have implications for how we design interaction.

The dialogue view of interaction evolved early in the history of computing. From the 1960s to the 1980s, rapid advances in computer displays made it possible to present much richer information to users. This made it possible to structure interaction in the form of a dialogue, freeing the user from having to memorize commands, as required in command-based interfaces. Initially, text consoles were used to provide access to programs running on mainframe computers. These programs had very limited means for interaction—basically, just a text display and a keyboard. Simple menu-based dialogues evolved into control programs based on some form of dialogue. Later, and before touchscreens, multifunction displays allowed for assigning physical buttons to the options presented on a graphical display. The display could show both text and graphical objects associated with each option. However, the response options were limited to a small number determined by the number of physical buttons. Nevertheless, this advancement enabled dialogue where the user would select an option by pressing the button corresponding to the desired option shown on the display. Since the 1980s, graphical displays and speech-based interfaces have made it possible to extend dialogues to offer virtually any number of options in any state. The dialogue view remains important, for instance for prompting (see Paper Example 18.0.1)

The key idea in the dialogue view of interaction is the *organization of communication as a series of turns*. Dialogue evolves through communication turns between two or more partners. In one turn, an appropriate communication act is made by one partner based on the communication context. The act aims to get the other partner to do or understand something. This understanding then forms the context within which the other partner takes their turn.

This broad definition has several immediate and important consequences for HCI. First, dialogue, as a form of interaction, is not limited to speech and language even though this is often our first interpretation of the term "dialogue." In Figure 18.1, two nominally different types of interaction are shown: speech-based and graphical. Both are forms of dialogue. This means that *the concepts of dialogue are applicable across modalities*. For instance, the seven-stage model of interaction proposed by Norman [600] applies to all modalities of interaction; Section 18.1 explains this model in detail.

Second, both dialogues in Figure 18.1 evolve in a turn-based manner. Each turn redefines the communication context. Siri's reply and the selection of "compose" would make no sense without the context provided by the preceding exchange. In other words, in dialogue, acts of communication are *conditional*: The meaning of a turn depends on the communication context. Consider the

Introduction to Human-Computer Interaction. Kasper Hornbæk, Per Ola Kristensson, and Antti Oulasvirta,
Oxford University Press. © Kasper Hornbæk, Per Ola Kristensson, and Antti Oulasvirta (2025).
DOI: 10.1093/oso/9780192864543.003.0018

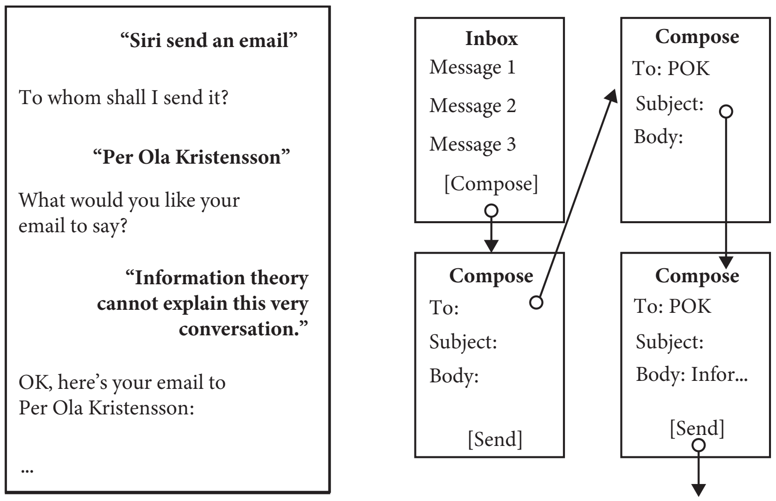


Figure 18.1 Two examples of dialogue-based interaction: speech-based dialogue (left) and graphical dialogue (right). Dialogue, as a form of interaction, is not limited to verbal communication.

dialogue to the left in Figure 18.1. The second utterance by the user (Per Ola Kristensson) would make no sense without the preceding exchange, which determined the context: sending a message and defining to whom to send it. Context, as we will see, can be understood through the concept of a *state*: The feasible communication acts and their effects are conditioned by the state of the partner. We will define the concept of state more precisely in Section 18.2.

Third, in dialogue, both the computer and the human participate in establishing a *shared* context. The computer does not simply receive a message; it also communicates the effects of that message. Therefore, the design of feedback, affordances, and cues is central to dialogue-based interaction. However, establishing and maintaining a shared context requires an intimate understanding of the situation and the contingencies that shape it. Section 18.3 outlines a view of dialogue developed by Suchman [804] that emphasizes the situated nature of dialogue.

Fourth, both the computer and the user may have *initiative*. For example, a pop-up window can be presented to confirm a risky selection. When there is a misunderstanding about the context of the dialogue, errors may happen, and the partners must recover from them. In interaction with interactive systems based on artificial intelligence, mixed-initiative dialogue is crucial. We discuss mixed-initiative interfaces in Section 18.4.

Dialogue is a particular view of interaction. It differs from the view of interaction as information and control (Chapter 17). In that view, a message is conveyed over a noisy, limited-capacity channel to the receiver. Interaction is understood as the success at the receiver's end in inferring the right symbols—the intended message. Note also that not all interaction is dialogue. For example, entering text on a physical keyboard is not considered dialogue. The symbols on the keyboard stay more or less the same throughout the task. Aside from the message to be typed, there is no need for the user to update a representation about the state of the computer. Next, we discuss the main views on dialogue in HCI.

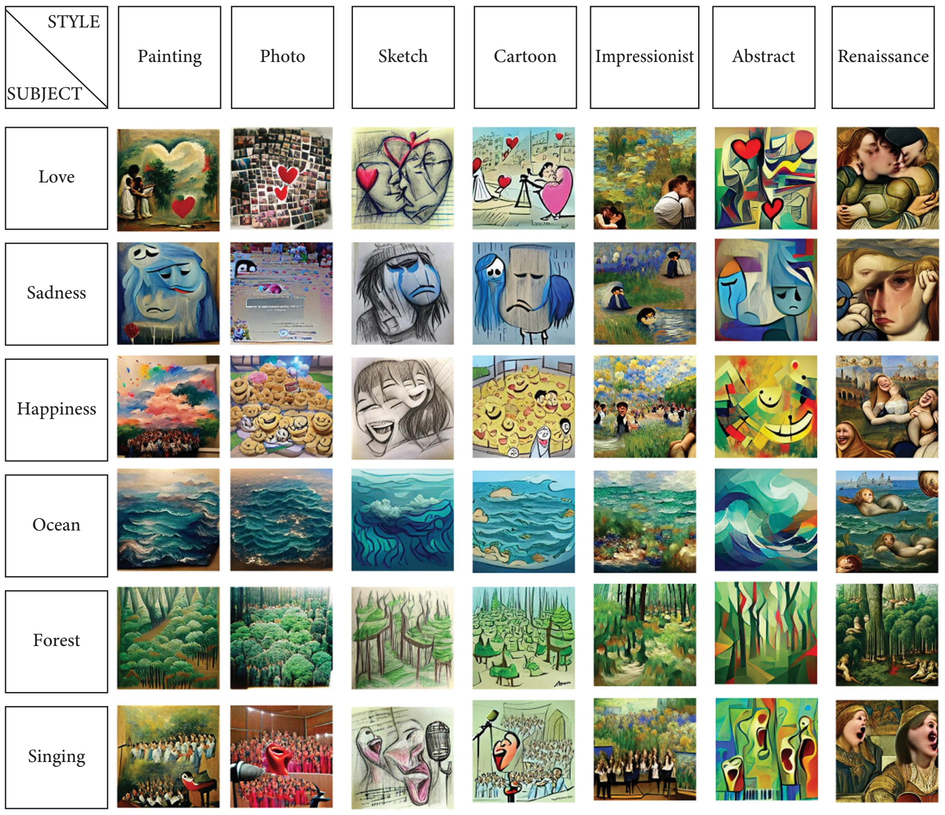
Paper Example 18.0.1: Prompting large language models

The view of dialogue as a model of interaction, which is as old as the field of HCI, continues to be relevant. New ways of interacting that rely on dialogue keep emerging; at the time of writing this book (early 2020s), large language models such as ChatGPT and Google Bard are making the headlines daily. The interaction with such models is primarily done through text prompts to which the model replies.

Liu and Chilton [488] noted that interaction with such models faces a dilemma. While it is possible to input anything as a prompt to such models, users must "engage in brute-force trial and error with the text prompt when the result quality is poor." The challenge here is sometimes described as *prompt engineering*—the search for prompts that give the output the user finds adequate for the task. In short, it is about improving the dialogue between the user, who issues prompts, and the system that outputs results based on the prompts.

Liu and Chilton [488] studied prompt engineering for text-to-image generation; see the figure in this paper example box, which shows examples of answers to the prompt "SUBJECT in the style of STYLE." Across five studies, they generated responses to different prompts and evaluated how well the results matched the subject and style. In addition, they carefully evaluated when the prompts were particularly successful and when they failed.

The results of the studies show that a small set of responses (3–9) may be sufficient to generate an idea of what a prompt can do; the computation of more responses might just waste time. The results also show that the SUBJECT would sometimes get lost in the STYLE; some prompts inadvertently led to grotesque or inappropriate images. The studies shed light on some of the difficulties of having a dialogue with a large language model.



18.1 **Dialogue as goal-directed action**

A significant early theory of dialogue interaction is the *seven-stage model* of Norman [600]. It considers interaction as goal-directed, turn-based dialogue. The model illustrates the seven stages of a communication turn (Figure 18.2). The stages are presented linearly, with each stage's output serving as input to the next stage. In the first stage, the user formulates a goal, such as "get money from the ATM" or "put images in the presentation on a line." The next stage is planning the actions to achieve that goal, that is, forming the intention of some action. It could be "give card to ATM" or "indicate the images to be aligned." Note that the user needs to know that the action is possible with the system. Next, the user specifies what actions they want to do, for instance, "insert card into slot" or "select all images." The user then needs to execute those actions, such as inserting their card into the ATM or clicking in a particular manner to select all images.

After this stage, the user aims to determine what changes they have made to the environment, for example, whether the ATM has accepted their card or the images have been highlighted. Then, the user needs to interpret the stage, that is, they need to make sense of what they have perceived ("Why are images highlighted?"). Finally, the user needs to evaluate whether what has happened constitutes progress toward their goal, that is, whether the feedback or lack thereof has brought them closer to what they want.

The model subscribes to a theoretical assumption about dialogue: The defining cognitive challenge in dialogue is understanding the communication partner such that the appropriate next turn can be taken. In other words, the dialogue is *intentional* or goal-directed: Users aim to drive the computer to a particular desired state. However, because of known limitations of human cognition, users cannot do this perfectly. They need to engage in (1) planning (deciding what to do) and (2) inference (interpreting the computer's state).

Norman offered two central concepts to help us understand these cognitive efforts: the *gulf of execution* and the *gulf of evaluation* (Figure 18.2). These two concepts describe inferential breakpoints for users seeking to express their intentions and interpret feedback from the system, respectively.

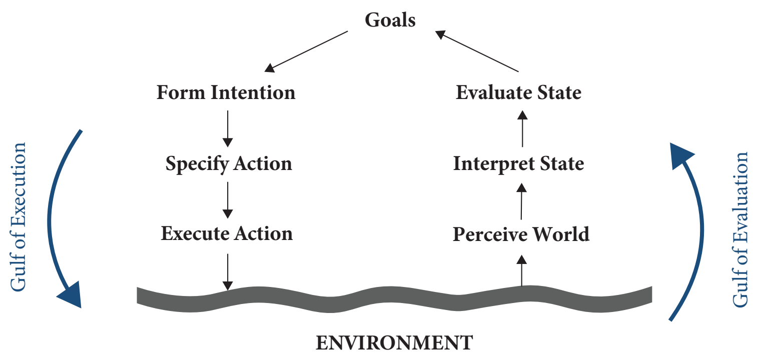


Figure 18.2 Donald Norman's seven-stage model views dialogue-based interaction through the lens of inference: The main challenge is to understand the state of the computer to plan actions to take it to the desired end state. The left-hand side of the model, from goal to environment, is called the gulf of execution. The right-hand side, from environment to goal, is called the gulf of evaluation.

Gulf of execution: This gulf is about knowing what to do to bring about a desired state change in the computer. For example, what should you do to get a piece of text copied to the clipboard and pasted in a specific location?

Gulf of evaluation: This gulf refers to knowing how a perceived change in the computer has moved it closer to the intended goal state. For example, imagine setting the temperature of an intelligent thermostat and not perceiving an immediate effect. How can you tell if your command had the desired effect on the system?

Norman's model inspired a decade of research on interface evaluation and design. In evaluation, which we discuss in Chapter 41, the model is part of the theoretical foundation that underpins the cognitive walkthrough. Norman's model stresses the need for users' acts to be understood by the computer and for users to understand the computer. Successful interfaces should also "provide a strong sense of understanding and control" [600, p. 49]. Paper example 18.1.1 contains an example of using the model.

Mapping and *feedback* are crucial concepts for understanding a computer's turn. Mapping requires the user to figure out how to accomplish a goal with an interface. It implies that "The user must translate the psychological goals and intentions into the desired system state, then determine what settings of the control mechanisms will yield that state, and then determine what physical manipulations of the mechanism are required" [600, p. 37]. Norman suggested that the ease of mapping is related to its directness, "where directness can be measured by the complexity of the relationship between representation and value, measured by the length of the description of the mapping" [600, pp. 28–29].

Affordance, which we discussed in Chapter 3, refers to how well users can interpret what actions are possible with a widget. *Visibility* is a handy related concept in design that underlies direct manipulation interfaces [416]. In direct manipulation interfaces (Chapter 28), the visual presentation of an object resembles its physical correspondent and can be directly acted on. For example, text in a text editor can be highlighted, deleted, or changed by point-and-click-style interactions. If a piece of text is selected, it is highlighted, which implies that actions can be performed on it. In this way, the user does not have to memorize the actions that are possible in a given state. Visibility can also concern the consequences of actions, such as when a progress bar shows the proportion of actions still needed to complete an installation.

The application of these and other concepts has also exposed some shortcomings in this approach to model interaction as goal-directed dialogue. The modelling subscribes to a linear account of the cognitive mechanism, going from goals to actions and back. However, according to current understanding in cognitive sciences, the picture is more complicated. One thing that is missing is an account of how beliefs about the computer are formed and updated and how they drive action specification. The current understanding is that users form internal models that predict how their actions produce perceived outputs, and they learn to minimize prediction errors. This explains why people explore interfaces (to develop better internal models) and why, eventually, they no longer need to compare outcomes against goals. Moreover, the model was initially used in a weak, heuristic sense and did not converge with efforts to implement interactive systems. This changed when formal models of dialogue were introduced, although these models do not cover all perceptual and cognitive aspects of dialogue.

Paper Example 18.1.1: Applying Norman's model of dialogue to the evaluation of service robots

Service robots are intended to assist or automate tasks in human activities, such as robots that serve food to customers or pick up orders in a warehouse. Designing interaction with service robots is challenging because of the limited means of communicating with a robot. Norman's model of dialogue helps to explain this. The model distinguishes between two "gulfs": the gulf of evaluation and the gulf of execution. According to Scholtz [745], the two gulfs manifest differently in the different roles a user may have when interacting with a robot:

Supervisor: In the supervisor role, the user monitors and controls the robot. This calls for continuous evaluation of the robot—in other words, the gulf of evaluation. "What is the robot intending to do?" If the supervising user wants to intervene, the gulf of evaluation becomes relevant.

Operator: In the operator role, the user modifies the robot's internal models to express what behavior is desirable and what behavior is not. In this role, the user must work with the gulf of execution to define which actions are suitable for the long-term goal of the robot.

Peer: In the peer role, the user acts like a peer, demonstrating goals to the robot and only intervening if needed. "Don't do it like that, do it like this." This role is similar to the supervisor role in the sense that both gulfs are important. The user both demonstrates correct behaviors (execution) and monitors the robot (evaluation).

Bystander: In the bystander role, the user can only intervene if the robot does something dangerous, for example, by walking in front of the robot to stop it. Here, the user deals with the gulf of evaluation: "What is the robot doing—is it potentially dangerous?"

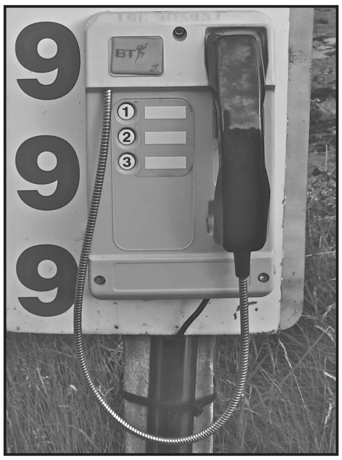
18.2 Dialogue as state-based interaction

Dialogue can be described using models of computation from computer science. Such models include finite state machines (FSMs), pushdown automata, and Petri nets. These models can be expressed with formal languages, including context-free grammars and graphs, and they can be implemented in event handlers in user interface (UI) software. We discuss some of these applications in the chapter on software engineering for UIs (Chapter 38); here, we focus on their applications for understanding dialogue.

Formal models of computation are suitable for describing discrete, moded dialogues. A *mode* refers to the variation in the interpretation of a user's input according to an internal state. In a modeless dialogue, all inputs are possible in all states and their interpretation is always the same. To understand modes and how dialogues can be captured using such a simple concept, we look at the theory of FSMs.

An FSM is a model of discrete computation applicable to dialogues. In computer science, an FSM is a special case of a Turing machine that reads but does not write on the tape. Formally, an FSM is a tuple $(\Sigma, S, s_0, \delta, F)$, where:

(a)



(b)

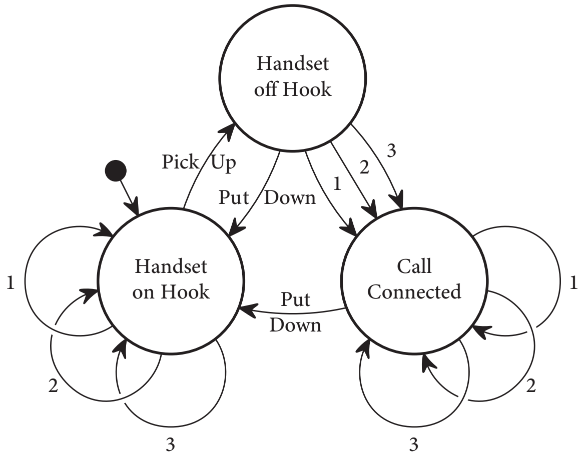


Figure 18.3 The picture on the left, (a), shows a UK coast guard phone meant for the public to make emergency calls. To make a call, there are three options: 1, 2, and 3. This is to save manufacturing costs. The diagram on the right, (b) shows the dialogue design as a FSM [120].

- Σ is the input, that is, a finite set of symbols;
- S is a finite set of states or modes;
- $s_0 \in S$ is the initial state;
- δ is the state transition function $\delta : S \times \Sigma \rightarrow S$;
- F is the set of final states, that is, a subset of S.

An FSM can be graphically presented as a graph where nodes represent states and edges represent transitions. Transitions are inputs by the user. States are modes that accept input designed by the transition function. FSMs are an effective way to describe how dialogue is structured; however, they are limited to memory-free dialogue.

A dialogue, as an FSM, is a sequence of input–state pairs from an initial state s_0 leading to a final state $f \in F$. An example is given in Figure 18.3. The figure shows a model of a telephone booth with three buttons and a handset that can be picked up and put down. This model helps us think through the states of the design. What, for example, should happen when the call is connected and the user presses a button again?

FSMs, as formal accounts of dialogue, are limited to transitions in a dialogue. They do not make assumptions about the way options or feedback are presented to the user. The same FSM could be implemented as an interface in multiple ways. FSMs do not make explicit assumptions about the user, either: FSMs are mute about how users perceive, reason, learn, and experience.

Despite their limitations, formal models offer several benefits as accounts of human–computer dialogues. One benefit is that formal descriptions of dialogues can be parsed interactively and used to operate a dialogue interface. They also aid design and engineering by highlighting desirable properties of a dialogue system [5]. For example, given a FSM, you can compute metrics for:

- **Consistency:** Are the same actions available, and do they have the same consequences across similar states? This can be computed, for example, by enumerating states that are similar and checking if available actions and their consequences are identical.
- **Dialogue length:** How many turns are needed to get from the initial state to the end state? In general, the fewer turns, the better. The duration of the dialogue is the average distance from the initial state to the final state.
- **Number of choices:** The number of options available to the user is a predictor of choice reaction time (Chapter 4). The choices are calculated by the number of edges leaving a state.
- **Error recovery cost:** If an error is made, how many turns are needed to recover from it? The fewer, the better.
- **Connectedness:** Can final states be reached from all initial states?
- **Strong connectedness:** Can final states be reached from all initial states via a particular action?
- **Reversibility:** Can the effect of a given action be reversed in one action?

Looking at Figure 18.3, we find the FSM does not have perfect consistency: Not all actions are available in all states. When the handset is on the hook, pressing any of the numbers is possible but has no effect. However, some actions are reversible: The state can be reset by hooking up the handset. Analyses like this have guided the design of dialogues for interfaces in safety-critical domains, such as the medical domain. However, analyses based on formal models such as FSMs only provide the formal structure of a dialogue, which may not be predictive of a user's typical experience. For example, whether an FSM is reversible does not determine the success or failure of a task.

18.3 Dialogue from a communication perspective

Communication between humans and computers is a long-standing topic of HCI research.[1] A cornerstone of this research is the book *Plans and Situated Action: The Problem of Human–Machine Communication* by Suchman [804].

Human–machine interaction, according to Suchman, is similar to but different from human–human dialogue. It is similar in the sense that people pursue a shared understanding: They actively work to make themselves understood. It is different in the sense that the communication abilities of computers are limited, which requires humans to adapt. Specifically, users adapt their approaches to match the capabilities of the machine rather than the other way around. For example, users may repeat utterances, increase the volume, or use simpler grammar and simpler words.

According to Suchman, robustness is a key consideration in the design of dialogue. Robustness refers to the communication partners' ability to achieve shared understanding even in light of misunderstandings and other unanticipated troubles. For example, when you visit a country whose language you do not speak, you still manage basic communication via gestures, facial expressions, and relying on context. Even if communication is less effective, it is relatively robust. Suchman argued that this is hard to achieve for computers, which rely on pattern recognition and cannot

[1] Not to be confused with the theory of communication proposed by Shannon, which we discuss in Chapter 17.

reframe communications as flexibly as humans. For example, when you talk with a human partner, you can always try to say the same thing using different words to correct a misunderstanding. Computer dialogues often do not offer that option.

What does the communication perspective add to our understanding of dialogue interaction? Research has drawn from linguistics, especially pragmatics, to understand how the way we talk with computers changes depending on the communication context. *Communication repair* refers to the "work of restoring shared understanding" when conversational partners misunderstand each other [60]. The partner who is talking needs to say something differently because the other partner did not understand what was said.

Code-switching refers to a switch in language to match the capabilities of the communication partner. For example, you likely use different language when talking with friends and with family. Code-switching can be analyzed linguistically from transcripts by looking at the meaning (semantics) or structure (syntax) of language. Such differences are important because depending on the communication context, people will have different expectations and styles they use in dialogue with a computer. Code-switching can be used for repair; it can also streamline communication and make it more robust.

Generally, repair strategies show sensitivity to the partner's actual or assumed communication abilities. A wealth of repair techniques has been identified, including clarification prompts such as "huh?" or "what?" In Paper Example 18.3.1, we discuss findings from a study of repair strategies with a digital home assistant.

Paper Example 18.3.1: Communication breakdowns in family conversations with Alexa

The introduction of speech-based assistance, such as Siri and Alexa, has increased expectations about the conversational capabilities of intelligent systems. However, how good are they really? A research group at the University of Washington [60] recruited 10 families and recorded their communications with Amazon Echo Dot (Alexa) for four weeks. Their focus was on *communication breakdowns*, that is, communications that failed to achieve the communicative purpose and must be recovered from in order to continue communication. The study looked at what types of communication breakdowns occurred and how communications were repaired by the families.

Their study exposed the limited nature of contemporary speech interaction from a conversational perspective. Although breakdowns were not that frequent—one occurred every four hours of use—they disrupted regular use and often required joint effort to overcome. The study found that families continuously modified their language to repair communications. For example, they changed prosody:

> Alexa . . . what is . . . the . . . temperature? [each individual word pronounced slowly and clearly instead of in a conversational manner]

They also raised their voices, like when talking with a human partner who does not listen:

> Alexa stop.
> Alexa stop! [louder]

continued

Paper Example 18.3.1: Communication breakdowns in family conversations with Alexa *(continued)*

The families also exaggerated articulation, a phenomenon known as hyperarticulation. The paradoxical effect of hyperarticulation is that despite trying to improve understanding, it can make speech recognition worse. The families also modified their use of words and their sentence structures and even provided definitions to Alexa. Additionally, more experienced users guided less experienced users on how to speak to Alexa. In the excerpt below, a mother encourages a five-year-old to ask a question and guides them in pronouncing the word “Alexa”:

(MOTHER): It’s A (pause) lexa. (emphasis on the last two syllables)

(CHILD): Uh (slight pause) leh (pause) ska. Is it going to rain for a little bit or is it going to be sunny for a little bit (said quickly and quietly), (pause) or both. (no rising intonation at the end to indicate a question)
(pause and no response from Alexa)

(RESEARCHER): That was a good question but it might have been a little too long, too many questions I heard.

(MOTHER): Let’s see. (pause) Alexa, is it going to be rainy all day or sunny all day or . . . (recording cut off)
(recording resumes with child giggling)

(MOTHER): Alexa, what did I ask you?

(ALEXA): Sorry, I’m not sure. (everyone laughs)

(MOTHER): Alexa, I asked if it was going to rain all day.

(ALEXA): Probably not. Each day of the next seven days . . . has at most a 30% chance of rain.
(child laughs)

18.4 Mixed-initiative interaction

Mixed-initiative interaction is the idea of organizing interaction in dialogue where both the computer and the human can take initiative. Unlike in the case of an FSM, the computing system can take action without a command from the user; the initiative is mixed.

As an example of such an interface, consider an email interface that automatically analyzes incoming emails. Such a system can infer whether an email is about scheduling a meeting and suggest or automatically invoke a calendar to assist the user in achieving this goal. This is possible if the system can accurately infer the user’s goal and assess the costs and benefits of providing an automated action or suggestion. Even if the system fails to trigger a calendar, the user can initiate this action at any point by clicking on the appropriate icon.

A mixed-initiative interface needs to infer the user’s goals so that it can act upon them. Several machine learning techniques are potentially suitable depending on the specific AI problem. Once the system has inferred the user’s goals, it needs to decide what action to take. One option is to take no action. To guide the decision on what automated action to take, the system can calculate the expected utility of each outcome for the user. Such utility can then be used as a threshold to only trigger an automated action if the estimated utility is sufficiently high. In practice, this is often hard because of the *poverty of cues* that the computer has access to. It may, for example, not have access to our gestural or facial expressions.

It is also possible to ask the user about their goal; whether to ask such a question can also be estimated by calculating its expected utility for the user. When an automated action is taken, it is important to consider the timing, as incorrectly timed automated actions can distract the user. It is sometimes possible to build models to time automated actions, for example, by estimating the time it takes a user to read an email. Generally, it is beneficial when mixed-initiative interfaces learn and adapt to individual users.

Horvitz [360] summarized the principles of mixed-initiative interfaces as follows:

Developing significant value-added automation: Only automate if a direct manipulation solution will be inferior. Identifying such opportunities may require user research and empirical evaluation.

Considering uncertainty about a user's goals: Systems should consider the uncertainty of users' actions and behaviors and incorporate such uncertainty into their automation mechanisms. For example, if the user says "home" to an intelligent assistant, does the user want to go to the home screen, view their actual home, or what? If there is ambiguity about what the user wants and wrong automation might harm the user, the system should ask for more information or not carry out the command.

Considering the status of a user's attention in the timing of services: Systems should be mindful of users' attention and the timing when invoking automated services. Systems should leverage such understanding when deciding on the costs and benefits of deferring actions to more beneficial times. This is particularly important in safety-critical domains such as driving.

Inferring ideal action in light of costs, benefits, and uncertainties: Systems should take into account the expected value of taking automated actions, given the costs, benefits, and uncertainties associated with context-specific decisions. For example, consider location-based recommendations presented to drivers. Information about a nearby cheap gas station (benefit) should be deferred until its potential effect on driving (cost) is minimal and there is little uncertainty about its value to the driver.

Employing dialogue to resolve key uncertainties: If the system is uncertain about the user's intent, the system should ask the user after having considered the cost of interrupting the user. While ambiguities can be resolved via dialogue, this principle warns against always asking the user: Every interaction bears a cost (e.g., time and effort) that should be factored in when deciding whether and when to engage in dialogue.

Allowing efficient direct invocation and termination: Since the system will be unlikely to always automate functions successfully, it is important that users can directly trigger and terminate functions. This means the status of automation should be made visible to the user. Opaque automation is stressful and prone to errors.

Minimizing the cost of poor guesses about action and timing: System-triggered alerts and suggestions should be designed such that if they are inaccurate or poorly timed, the cost of interruption of the user is minimized. This can be achieved, for example, by making suggestions less distracting, using natural timeouts if there is no user response, and supporting swift user actions to dismiss suggestions.

Scoping precision of service to match uncertainty, variation in goals: It may be beneficial to dynamically adjust the use of systems. If a system operates under a high uncertainty of the user's goals, the system should perform less automation to avoid interrupting the user with poor suggestions.

Providing mechanisms for efficient agent–user collaboration to refine results: systems should be designed to allow users to refine or complete an analysis initiated by the system. for example, if a robotic vacuum cleaner maps a room autonomously, users should be given an option to edit the map.

Employing socially appropriate behaviors for agent–user interaction: Any interruptions by a system should be compatible with the social expectations of the user being interrupted and offered automated services. For example, social media feeds may integrate AI-generated and human-generated content without disclosing the source. This may break the expectation that all content of social media feeds is generated by other (human) users.

Maintaining working memory of recent interactions: Systems should retain recent interactions and allow the user to refer to prior objects, actions, and services in their interaction with the system. This principle is important for efficient interaction and the coherence of behavior. It allows the system to avoid asking the same questions every time the user is in the same context.

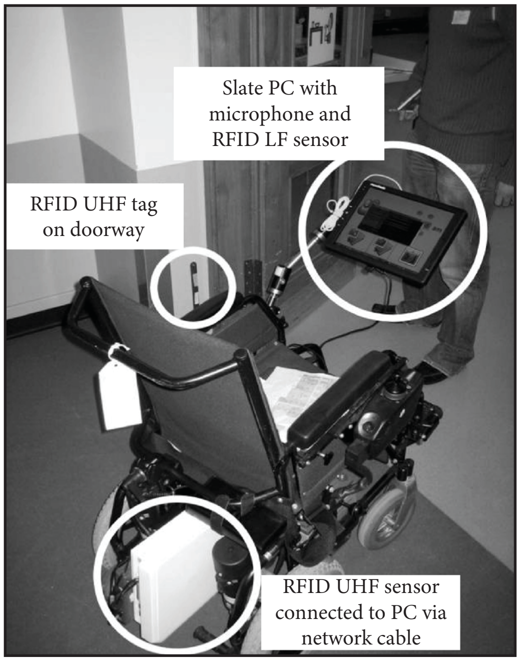
Continuing to learn by observing: Systems should continuously learn and adapt their models of users' goals and needs. Because users' goals and situations change over time, the system is never "ready."

Paper Example 18.4.1: Augmentative and alternative communication for personal narratives

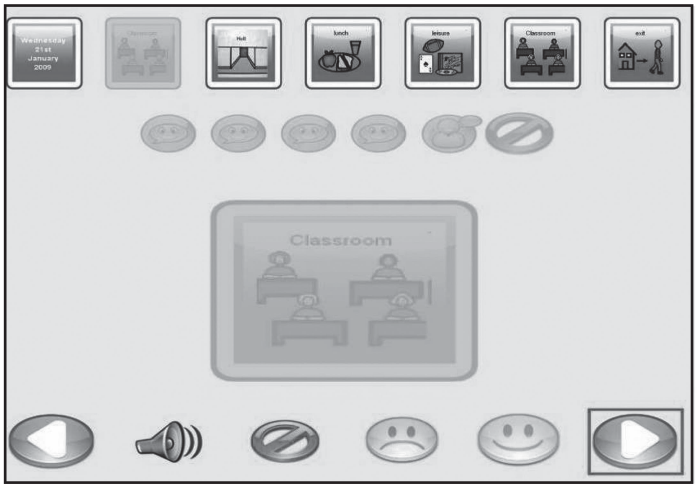
Augmentative and alternative communication (AAC) is a class of interactive communication applications for people with complex communication needs, such as those with little or no functional speech. For example, cerebral palsy (CP) limits the motor system and musculature, hampering articulation and gesturing with hands. Individuals with CP often have sensory and cognitive impairments that negatively affect their language development. AAC applications often use dialogue interfaces. They show graphical symbols representing words or phrases. Users can scan pages of symbols and select symbols to form sentences. Symbols are selected by pointing with a finger or an input device (e.g., a joystick). When the message is ready, the computer speaks out the composed sentence via a speech synthesizer. This process is painstakingly slow, which limits the users' ability to participate in communicative activities and lowers their quality of life.

Human–human communication is streamlined by common ground (Chapter 9); what if computers could have common ground with an AAC user?

The *How was School today. . .?* concept was developed iteratively with two children with CP and the help of school staff. This AAC system was designed to use *context* to facilitate the creation of personal narratives [75]. The authors called their approach "data-to-text": The idea was to add sensors to the environment and the wheelchair of a user. As the user traverses the environment, the locations create *events* that are made available as symbols. For example, when going around in the school, entering a room or meeting a person can create an event.



These events can be used to facilitate communication. The *event narration* UI allows the user to select and organize events that took place during the day. The event buttons allow the user to change the order of events. Users can also add annotations and invite the conversational partner to assist in composing the message. For example, if the user selects "Katie was there," the partner may respond "Do you like her?" by selecting the smiley button. The user may then respond "She is nice" or "I don't like her." Users can also express "Do not speak" to skip the discussion topic.



18.5 **Limits of turn-taking as a model of interaction**

This chapter has presented turn-taking as a model of interaction. However, it is not obvious that this is how dialogue with computers takes place. The cognitive scientist Kirsh presented a criticism of Norman's view of dialogue and developed an alternative based on the theory of embodied cognition [416]. This critique illustrates the limitations of viewing interaction as dialogue.

Kirsh points out that Norman's model makes an unrealistic assumption: The user is assumed to know the environment and its options and is merely picking an option. In practice, we do not always know what the options *mean* or even what options are available. Kirsh argued that users need to actively *explore* interfaces to become aware of the available functions and how they work. Via exploration, they also learn about their own abilities in using them. Consider the first time you launch an application; you probably try out various actions to see what happens. Kirsh argued that the *discoverability* of such options is as important as their visibility; however, discoverability is not covered well by Norman's theory.

Kirsh argued that we are not just passively reacting to computer-generated options. If we look at interaction at a higher level, beyond a single action, we see that users are also actively influencing their environments. Users are "architects" of their environments, as Kirsh put it. For example, users may change the settings to turn on or off a function or change the way it behaves. They also choose the applications they use. Such tailoring behaviors are not explained by Norman's intention–action–response–interpretation–evaluation cycle.

Building on such criticism, Kirsh proposed an alternate model, showing that every stage in Norman's model can have an interactive relationship with the environment. We learn about options by exploring the interface, discover how to specify actions by trying them out and observing the outcomes, position our bodies to better perceive environmental responses, and adjust the environment to facilitate response evaluation. This relationship of dialogue with the environment is depicted in Figure 18.4.

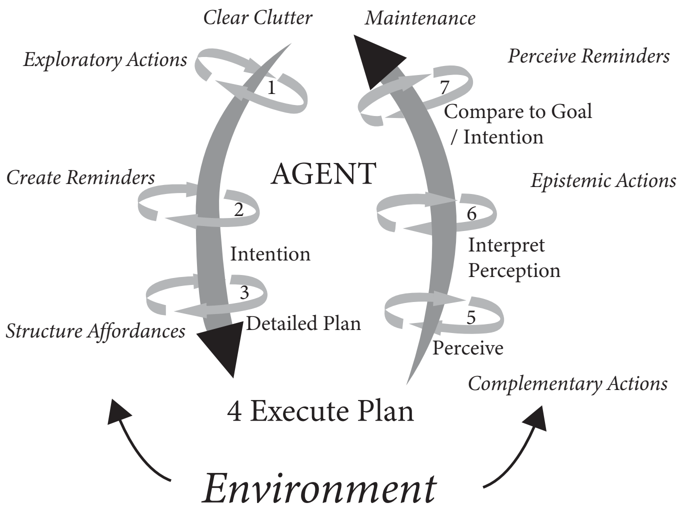


Figure 18.4 Kirsh's model of embodied interaction emphasizes the interactive role that the environment plays in action. We explore our environments to learn what is possible, change the environment to change our options, and position ourselves to better interpret and perceive relevant responses from the environment.

What does this mean for design? A good design allows users to try out options without penalizing trials. It allows them to customize and personalize the interface based on individual abilities. It presents not only options but also their consequences to the user. This serves to nuance the understanding of interaction as dialogue.

Summary

- Dialogue is about the organization of communication as a series of turns between communication partners.
- The core elements of dialogue are communication turns, the communication context, and turn interpretation.
- Dialogue interaction includes speech-based and graphical interactions.
- Dialogue can be understood as computation, goal-directed action, communication, or embodied action. Each perspective provides specific methods for the analysis and design of dialogue.

Exercises

1. Core concepts of dialogue interaction. Dialogue offers a rich conceptual framework for understanding interaction. First, choose an everyday interaction with which you are familiar. It can be anything from filling out a form to chatting with a chatbot. Then, choose a particular dialogue to focus on, for example, creating a user account or printing a document. Now, provide the following information for the dialogue:
   - **Communication partners**: Who are the actors in the dialogue?
   - **Communication goals**: What is the final state the computer should be in for the user to consider the task completed?
   - **Communication act**: What are the possible communication acts? In other words, what are the possible utterances or messages that can be delivered?
   - **Communication sequence**: Draw a sequence of the communication turns leading to the goal, similar to Figure 18.1.
   - **Initiative**: To which degree can each partner initiate communication on their own?
   - **Cue**: Which cues are shown to help the user understand the state of the computer?
   - **Feedback**: Which cues are shown to help the user understand the effects of their communication acts?
2. Theories of human–computer dialogue. Consider the following potential dialogue interfaces: (a) a user interacting with an automated chat agent from an airline to resolve a delayed flight; (b) a child uploading homework using a web interface; and (c) a user who is trying to show a picture on their mobile phone on a nearby television screen. Make any necessary assumptions about the interfaces and discuss which model of dialogue would provide the most insight for each interface: (a) FSMs, (b) dialogue as goal-directed action, (c) dialogue as embodied action, or (d) dialogue from a communication perspective.
3. Gulfs. Pick a graphical user interface, for example, something you use for education. Then, choose a task, for example, “sending a message to the teacher.” Assess this task through the lens of Norman’s two gulfs: the gulf of evaluation and the gulf of execution.

4. Mixed-initiative interfaces. Pick any AI-assisted feature that you are familiar with. Assess it against Horvitz's principles of mixed-initiative interfaces.
5. Comparing mode-based interactions. A device is designed to allow users to control the relative humidity in their house. The device has two modes. In Automatic mode, the system keeps the relative humidity in the 50%–60% range. In the Manual mode, the user can set the desired level of relative humidity and the system will attempt to maintain it. The device is a small wall-mounted unit with the following UI elements. (a) The visual display indicates the current level of relative humidity and whether the system is in Automatic or Manual mode. (b) The "–" and "+" buttons enable the user to reduce or increase the desired level of relative humidity, respectively. (c) The "Automatic" button puts the system in Automatic mode. If the user pushes the "–" or "+" button, the system switches to Manual mode and remains in that mode until the user pushes the "Automatic" button. (a) Draw a state diagram for this system. (b) By viewing interaction with this system as goal-directed action, explain the steps comprising the gulf of evaluation and the gulf of execution for this UI. (c) State the type and level of automation of this system. (d) Is this system a mixed-initiative interface? Justify your answer.