20

Automation

Interactive systems have evolved to increasingly automate work that people used to do. They typically do so by using some kind of AI that takes over part of a task from the user. This can take many forms. For instance, when entering text on a mobile device, language models propose completions of phrases and correct typos. Another example is the use of AI models for calculating insurance fees based on customers' health and other data. Semi-autonomous vehicles can drive parts of a route autonomously. They may need to hand over control to a human driver in parts they cannot handle. Recommender engines are used not only for movies and music but also for creative activities. For example, a recommender engine can propose possible slide layouts to an information worker or poster designs to a graphic designer. The designer can explore the recommendations, pick an interesting one, and use it as the basis for the final design. Large language models are used to summarize long texts and generate images.

An exciting and distinctive aspect of this type of interaction is that users interact with something that may have *agency*. Automation technology is capable of doing things on its own. We call such technology an *agent*, also sometimes called a software agent, intelligent agent, conversational agent, personal digital assistant, or intelligent interactive system. Interaction between users and agents opens up new ways of thinking about interaction. In some instances, an agent may only respond when the user provides input, similar to command-line interfaces and graphical user interfaces. In other scenarios, the agent may be continuously monitoring and analyzing the user's actions and proactively acting to assist the user.

The increasing autonomy of computing systems raises new challenges for human–computer interaction (HCI). The performance of the system must be reliable and controllable. Its behavior should be safe, and the way it is designed and used should be ethical [768]. Users need to trust the system's decisions and ability. It should be made clear to the user what it can and cannot do. For example, a recommender system can tell the user that the user *may* be interested in certain recommendations, thereby indicating the system is not completely certain these recommendations are relevant [19]. When the system fails, users need to be able to redirect it. To avoid biases and discrimination, some level of transparency and explainability is required.

Research in HCI has shown that interaction with partially automated systems is nontrivial (Paper Example 20.0.1 contains a more extended example of the tradeoffs in automation).

- Roy et al. [715] explored what happens when users doing a task with a simulated crane need to choose between further using automation and manually continuing the task. The authors showed that the decision of whether to use automation is affected by the users' perception of its accuracy as well as how easy it is to do the task themselves.
- Long-term engagement with automation has taken place for more than a decade in the home. Brush et al. [105] showed how lighting control, motion detectors, and multi-room audio/video systems were perceived as inflexible, difficult to manage, and difficult to set up securely.

Introduction to Human-Computer Interaction. Kasper Hornbæk, Per Ola Kristensson, and Antti Oulasvirta, Oxford University Press.
DOI: 10.1093/oso/9780192864543.003.0020

- Cai et al. [117] interviewed 21 pathologists who used a deep neural network to aid in the diagnosis of prostate cancer. The interviews showed that pathologists needed to learn more about the network's strengths and limitations to use it effectively. They also wanted to know the design objective of the network and the kind of data on which it was trained. This study indicates that automating tasks is only one part of what makes an interactive system work in its intended context.

This chapter discusses interaction with automation and AI. A major role of AI has been and continues to be automation, so we discuss automation first. In HCI, automation poses the foundational problem of *task allocation*: Who does which task or subtask—the system or the user? Moreover, how much should we automate? This raises the question of whether some functions that are difficult for an AI to perform should be allocated to users. However, AI is more than just automation; later in this chapter, we discuss emerging forms of interactive AI and related challenges.

Paper Example 20.0.1: Automation versus direct manipulation

The paper "Direct manipulation vs. interface agents" [770] contains a classic debate between Shneiderman and Maes on the future role of direct manipulation (a technique for graphical user interfaces discussed in Chapter 28) and "software agents"—agents that enable task delegation. The debate serves as a good introduction to some basic questions at the intersection of AI and HCI.

The expressed views and ideas, published in 1997, are still relevant today. Software agents may allow users to achieve their goals in complex environments with limited expertise. For example, a software agent may offload work from the user and offer suggestions without being prompted. Maes noted the benefits of such agents:

- A software agent knows the individual user's habits, preferences, and interests. This is important to help keep up with the increasing complexity of interactive systems and their applications.
- A software agent is proactive. It can serve as an extra pair of eyes or ears for the user, who can delegate tasks to the agent.
- A software agent can be long-lived. It can persist over sessions and systems, further aligning itself with the user's habits and interests.

However, it is still important for users to be in control. Shneiderman argued the following points supporting direct manipulation and powerful tools:

- Direct manipulation allows the user to predict what will happen, explore the system, and feel in charge.
- By showing information to users and letting them filter, select, and focus on interesting information, users get an overview of information and actively control which information to drill down on.
- Human capabilities and responsibilities should not be confused with those of machines. Direct manipulation does not anthropomorphize user interfaces or deskill users.

Later in the debate, both speakers agreed on the importance of allowing users to directly manipulate software agents. This crystallized into mixed-initiative interfaces, which are

discussed in Chapter 18. More than 20 years later, a similar discussion was held [869]. This suggests that the uses and trade-offs of automation are a central problem in HCI.

20.1 Human–automation interaction

Automation refers to technology that assists users by performing a task or a subtask on their behalf. It has been defined as "a device or system that accomplishes (partially or fully) a function that was previously, or conceivably could be, carried out (partially or fully) by a human operator" [638]. Throughout the history of automation, a central challenge in HCI has been posed by the complexity of autonomous systems. Users must be able to understand and control them. They need to find and integrate information from dynamic and different sources. They need to understand how to delegate tasks, supervise their execution, and intervene if needed. Perhaps the most significant aspect for HCI is that safe and effective operation critically depends on the user and, therefore, the user interface.

The study of human–automation interaction emerged from the need to develop automation in safety-critical systems. For example, a stock analyst may be monitoring several sources of real-time information on trading algorithms. An operator of a submarine demining vehicle may need to decide whether an underwater mine is present based on noisy detection signals from sonar sensors. An air traffic control (ATC) operator may need to continuously scan several displays showing real-time ATC information and make decisions on which instructions and what information to communicate to pilots and other ATC operators. In all cases, getting the right balance between automation and human control is crucial. This depends, among other things, on how tasks are shared.

20.1.1 Task sharing

A central concern in the design of automation and AI is *task sharing*. For the sharing of tasks to occur, there have to be at least two tasks or subtasks, which can then be shared in two different ways. First, tasks can be *time-shared*, which implies that the user performs multiple tasks. In general, perfect time-sharing with no degradation in performance occurs only for tasks that are automatic, such as speaking while walking. Such tasks can reach *automaticity*. Second, tasks can be shared in terms of *control*, which means that some control over the tasks is assigned to another agent, such as another user or a machine.

There are three dominant theories of time-sharing in human cognition. The first is called the *single channel theory*, which posits that there is limited capacity in the human information processing system in a time-sharing scenario. When the channel capacity is exceeded, multiple tasks transition from parallel processing to serial processing. This theory is contradicted by empirical research showing that a multi-processor model better explains empirical data. The second theory is the *multiple resources model*, which states that resource limitation concerns the entire system rather than a channel (Chapter 5). The third theory is *information processing analysis theory*. If at least one task can be carried out automatically, the other task can be carried out with little or no impact on performance (at an appropriate time–error trade-off point). If both tasks demand controlled processing, then the strategy in processing is split into two mechanisms: facilitation and inhibition. The implementation of such a strategy requires attentional resources, which can lead to task interference when the demand exceeds the available capacity.

Successful time-sharing depends on the strategy and difficulty of the task in terms of temporal constraints—how many tasks are processed in a given interval—and task complexity—the quantity of information that needs to be processed for a given task. In a task-switching situation, the user must activate resources for the second task and inhibit resources for the first task. If the user fails to do so efficiently, performance is reduced, sometimes dramatically.

Another way to share tasks is to *share control*, which is also covered in Chapter 17. There are three principal ways of achieving this type of sharing. First, control can be shared via an *extension* that allows a machine to amplify human ability. An example of such control sharing is power steering in a car: The car provides additional work to allow the driver to turn the wheels with less effort. An HCI example is mouse acceleration, which allows a user to move the cursor on the screen farther than the physical movement of the mouse.

Second, control can be shared via *relief*, which means that the overall burden on the human is reduced by the machine. An example is automatic shift transmission, which relieves the driver of the task of changing gears in a car. An HCI example is text entry using autocomplete, which prevents the user from correcting typing mistakes as they type.

Third, control can be shared via *partitioning*. In this case, a task is decomposed into parts that can be addressed by humans and machines separately. An example of such control sharing is semi-automatic parallel parking, which provides the driver with some braking ability while the machine controls the speed and steering of the car. An HCI example is automatic spell checking, where the system detects and highlights incorrectly spelled words but does not change them. Instead, the user has to take an explicit corrective action, such as selecting a misspelled word and choosing an alternative.

Independently of the chosen strategy, some tasks are to be done by the interactive system and some by the user. The allocation of such tasks is called functional allocation.

Paper Example 20.3.1: Shared control and the H-metaphor

So far in this chapter, we have discussed the case of a single user controlling a computer. What if we have two agents both capable of controlling it? How could we *share* control between them? The question of shared control is timely; semi-autonomous vehicles are only partially autonomous. They need the human to assist them and, therefore, some way of handing control over to the human driver. They also need to have guidance from the driver, for example, on the choice of route.

Shared control is about carrying out a task together with a competent partner [1, p. 511]: "In shared control, human(s) and robot(s) are interacting congruently in a perception–action cycle to perform a dynamic task that either the human or the robot could execute individually under ideal circumstances." What does this mean?

The *H-metaphor* is a metaphor for understanding shared control [246]. Here, the "H" stands for "horse": the horse metaphor. In short, it means that shared control is like riding a horse. The metaphor makes two points about shared control. First, communication is vital for sharing control, and this can happen at different levels; second, both agents must have internal models of each other to understand what those communicative acts mean.

When riding a horse, the rider communicates high-level information (e.g., the goal) to the horse but must be ready to guide the horse at a lower level. When the horse knows what to do, for example, if the route is familiar, the rider may not need to engage in low-level control. This form of control, called *loose rein control*, is possible if the horse knows what the rider wants.

If the horse is not controlling the way the rider wants, the rider must communicate at a lower level to tell the horse what to do. In horseback riding, the subtle movements of the rider's body, such as applying pressure with the legs, guide the horse, just like voice commands ("Whoa!"). This is called *tight rein control*. The horse can also communicate with the rider at all times regardless of the type of control.

The H-metaphor is instructive: When two agents sharing control have asymmetric capabilities, both loose and tight rein control should be available. In this respect, our everyday interfaces appear clumsy when compared to the relationship that an experienced horseback rider and a horse can achieve. Control does not need to be either/or like in many semi-autonomous vehicles. Four levels of shared control can be distinguished [1]: strategic (e.g., setting a destination), tactical (e.g., doing a specific maneuver like merging into a lane), operational (e.g., maintaining a certain distance from another car), and execution (lowest-level of controlling locomotion, steering, and so on).

20.1.2 Task allocation

Task allocation is a central challenge in HCI and automation. Which system functions should be automated, and to what extent should they be automated? There are three main strategies here.

The first strategy is called *maximum automation*. Here, each task that can be automated is allocated to a machine. The aim is to increase efficiency, reduce costs, or both. This is a popular strategy for automation. However, this strategy has two implications that may cause issues. First, by automating everything that can be automated, the user is left with functions that, by definition, the designers find hard, expensive, or difficult to automate. Therefore, this automation strategy defines the roles and responsibilities of users in terms of automation instead of the other way around. The second implication is that as automation approaches its maximum, the role of the human operator becomes increasingly critical. For example, automated errors may remain unchecked until the user intervenes. This latter point, to which we will return later, is referred to as the "irony of automation" [36] in the automation literature.

The second allocation strategy is assigning each function to the most capable agent, which can be either a human or a machine. While this may sound good in theory, it is difficult to successfully implement this strategy in real-world automation problems. When functional allocation was first introduced in the literature, Fitts (whose work we also discussed in Chapter 4) made lists of what people are good at and what interactive systems are good at. These so-called Fitts' lists [242] provide a basis for functional allocation by indicating whom to assign a task (see the exercises at the end of this chapter for an example).

The third allocation strategy is to allocate each function in a way that maximizes economic efficiency. This is a very attractive strategy for many stakeholders. However, it can also be very risky since it requires accurate modeling, which is difficult to achieve in practice.

All three strategies have a common deficiency in that they may not always be able to adhere to the principle of human-centered automation, whereby a human has final control. This is also called the authority problem.

20.1.3 Types and levels of automation

This discussion highlights that simple automation strategies are unlikely to be successful. For this reason, designers can benefit from frameworks that support system design that involves automation. We now discuss one such framework: the *types and levels of automation framework* [639].

The framework uses two sets of evaluation criteria to help designers determine the appropriate type and level of automation for each application. The primary evaluation criteria concern the impact of the chosen types and levels of automation on human performance. The secondary evaluation criteria include automation reliability and the cost of decisions or outcomes.

Levels of automation: The levels of automation range from high automation to low automation. The numerical mapping of automation levels is given in Table 20.1. The table shows that the level of automation ranges from 10, where the computer system is a fully autonomous system that ignores the user, to 1, where the user is in complete control.

Types of automation: The types of automation can be understood by viewing a human operator as a simple four-stage model of human information processing:

1. Sensory processing
2. Perception and working memory
3. Decision-making
4. Response selection.

This four-stage model of human information processing is then mapped to the types of system functions that can be automated (acquisition, analysis, decision, and action), as shown in Table 20.2. In addition, adaptive automation is an emergent type of system function automation that captures a combination of these four types of automation.

Acquisition automation corresponds to the first human information processing stage, sensory processing, and it is realized by the system sensing and registering input data. An example of low-level automation is assistance in sensor adjustment, such as a system mechanically moving a radar sensor to lock on a detected target. An example of moderate automation is a system organizing information according to criteria such as a priority list or highlighting information based on static or dynamic criteria. This could be, for example, a display highlighting the rate of change in some variable of interest. This could be indicated by increasing the intensity of some pixels more rapidly than others in the display.

Table 20.1 The levels of automation and their descriptions, ranging from no automation (1) to complete automation (10) [639].

Level of automation	Description
10	The computer decides everything, acts autonomously, and ignores the human.
9	The computer informs the human only if it decides to do so.
8	The computer informs the human only if asked by the human.
7	The computer executes tasks automatically and informs the human.
6	The computer gives the human a set time to veto a decision before automatic task execution.
5	The computer performs a task automatically if the human approves it.
4	The computer suggests one alternative action.
3	The computer offers a narrowed selection of options.
2	The computer offers a complete set of options.
1	The computer offers no assistance: The human must make all the decisions.

Analysis automation refers to the automation of information analysis and involves inferential processes. It corresponds to the second human information processing state: perception/working memory. An example of low-level automation is the extrapolation or prediction of data over time, such as a system predicting a trend for the output of an industrial plant based on historical sensor data. An example of moderate- to high-level automation is a system integrating multiple sources or input variables. This could be a display with emergent perceptual features, such as an optical see-through display with a landing strip intended to assist a pilot in landing an aircraft. An example of high-level automation is a context-dependent summary of data.

Decision automation means deciding and selecting appropriate actions among alternatives. This type of automation corresponds to the third human information processing state, decision-making, which the machine is either augmenting or replacing altogether. Examples of decision automation include route planning and adaptation, such as to avoid bad weather, and systems providing medical diagnosis support. The difference between analysis automation—inference—and decision automation is that in the latter the system must make implicit or explicit assumptions about costs and values inherent in all decisions. The levels of automation reported in Table 20.1 represent the various levels of automation that can be applied to decision automation. For example, using decision automation to avoid aircraft ground collisions may be level 4 automation: A decision is recommended, but the pilot can choose to ignore it. As another example, an automated ground collision avoidance system could be levels 5–8 automation: The system can assume control if the pilot fails to act. The specific level depends on whether the system gives the pilot some time to act and the conditions for informing the pilot (Table 20.1).

Action automation means the machine is partially or fully executing an action choice. There are many levels of machine execution, such as automating human manual actuation by hand or via voice commands. A common example is automating a series of manual steps, each requiring one or more key presses, into a single key press. A photocopier is another example of action automation: It can perform various levels of action automation, from the automatic sorting of copies to the automatic collation of papers and automatic stapling. An example of very high-level action automation is modern aircraft automatic landing systems, which track user interaction and automatically execute context-specific subtasks. In the HCI domain, a common example is the automatic execution of a procedure when certain conditions are met, such as an operating system engaging in an automatic update when the update has been fully downloaded and the computer is idle.

The type and level of automation can also be allowed to vary depending on the context, giving rise to a fifth type of automation: *adaptive automation*. The type or level of automation may vary in changing situational settings, for example, when a computer display shows a sudden influx of information that the user needs to process quickly. An example of adaptive automation is aircraft

Table 20.2 The types of automation mapped to stages of human information processing.

Type of automation	Human information-processing stage
Acquisition automation	Sensory processing
Analysis automation	Perception/working memory
Decision automation	Decision-making
Action automation	Response selection
Adaptive automation	Context-dependent

navigation automation, which can be high or low depending on how quickly the joint aircraft–crew system must react to an event to avoid a catastrophic outcome. A similar mechanism can be implemented in cars: The car may engage in manual handover if the system is unable to manage an ongoing situation. A hypothetical example of adaptive automation is a system automatically correcting spelling and grammar mistakes, which may dynamically adjust the system's level of autocorrection depending on the type of text the user is writing.

20.2 What makes automation good?

User-centric evaluation criteria: The study of human–automation interaction emerged in safety-critical domains, such as driving and aviation. Besides accidents, four criteria are used to evaluate automation from the operator's perspective: mental workload, situation awareness, complacency, and skill degradation.

Automation can both reduce and increase users' *mental workload*. For example, automation can enable an organization of information that makes it easier for users to understand. Automation can also avoid searches or communication, for example, by providing pertinent context-appropriate data to the user. In addition, automation can collate relevant information sources and present them to the user. Furthermore, automation can provide predictions that may reduce the user's workload, such as predictive flight path displays in aircraft, which have been shown to reduce pilots' mental workload. All these are examples of automation reducing the user's mental workload. However, the mental workload can also increase due to systems that, for example, are difficult to initiate, understand, or engage with, or require extensive physical work to operate.

Automation can also affect users' *situational awareness*. Automation may reduce users' awareness of the system, including its dynamics and outputs. In general, humans tend to be less aware of changes in a system or environment if those changes are under the control of another agent. As such, the use of automation almost always implies some level of delegation of authority. Another concern is that automatic decision-making based on the information available may make the operator less aware of the information since the operator is not actively engaged in evaluating it as a basis for making decisions. This increases the risk that decisions are based on incorrect or low-quality information.

Another consequence of automation on human performance is *complacency* arising from overtrust or overconfidence in automation. For example, a user may fail to monitor automation or information sources if the automation is at a high level and is perceived to always be accurate. There is also a risk of errors being introduced due to users overly trusting imperfect automated information analysis, such as automated data filtering or prediction. Several well-known issues in AI systems exacerbate this risk, including the system's failure to assess the quality of the data, estimate the uncertainty in the data, and mitigate bias in the data. For example, attention-guiding systems such as automatic cueing systems can result in operators not paying attention to uncued areas. A similar effect has been observed in spell checking [267] and when indicating low-confidence words outputted by a speech recognizer [849].

The fourth consequence of automation on human performance is *skill degradation*, that is, users forgetting how to carry out functions or witnessing their skills decay due to automation.

System-level evaluation criteria: While user-centric evaluation criteria look at the system from the viewpoint of its operator, we can also look at the *joint* performance of the human–automation system. This is often done via two evaluation criteria: automation reliability and the costs of decision and action outcomes.

Automation reliability captures the extent to which the system is effective in automation. Depending on the type and level of automation, several factors may be important. One important aspect may be the *true positive rate*, also known as *sensitivity*, which is a measure of the ability of a system to correctly diagnose a condition. For example, an automated spell checker function with high sensitivity can flag the vast majority of words that are misspelled. Another important factor may be the false positive rate, also known as the *false alarm rate*. It is a measure of the system's inability to separate positive from negative cases. A high false alarm rate means the system is constantly incorrectly flagging information. An example would be a spell checker that keeps flagging words that are spelled correctly. A high false alarm rate may cause fatigue among users and may foster distrust in the system, which may well be justified.

Lower automation reliability may be acceptable in some circumstances, for example, if the users are provided with alternative means of assessing information, such as access to raw data in an easy-to-digest format. Another example is the case where users can be trained to understand not only the decision-making tasks but also the underpinning capabilities and limitations of the automation solution.

Automated systems that classify or predict events have inherent errors in their models, as such models are imperfect. Therefore, such systems should be designed to take into account the fact that automated results will inevitably be incorrect on occasion. Generally, automated systems that indicate when automation may fail or has failed are more likely to gain an appropriate level of trust from users.

The *costs of decision and action outcomes* are also important. Any potential benefit of automation has to be compared with and weighted against any possible disadvantages. Examples of such disadvantages include an increased mental workload, reduced situational awareness, higher complacency, and a higher risk of skill degradation. Therefore, it is often wise to carry out risk analysis, which is discussed in the chapter on safety and risk (Chapter 37).

Application: A workflow for determining the type and level of automation is presented in Figure 20.1. The application of this framework has six main steps:

1. Identify the types of automation that are applicable to the task.
2. Identify the levels of automation that are suitable for the task.
3. Based on the primary evaluation criteria, evaluate the consequences of automation on human performance. This may require iteration, such as revisiting the identification of types and levels of automation in light of insights from the evaluation.
4. Determine the preliminary type and level of automation of the task.
5. Apply the secondary evaluation criteria: automation reliability and the costs of decisions and outcomes. This step may also require iteration and revisiting the identification of types and levels of automation in light of insights from the evaluation.
6. Determine the final type and level of automation of the task.

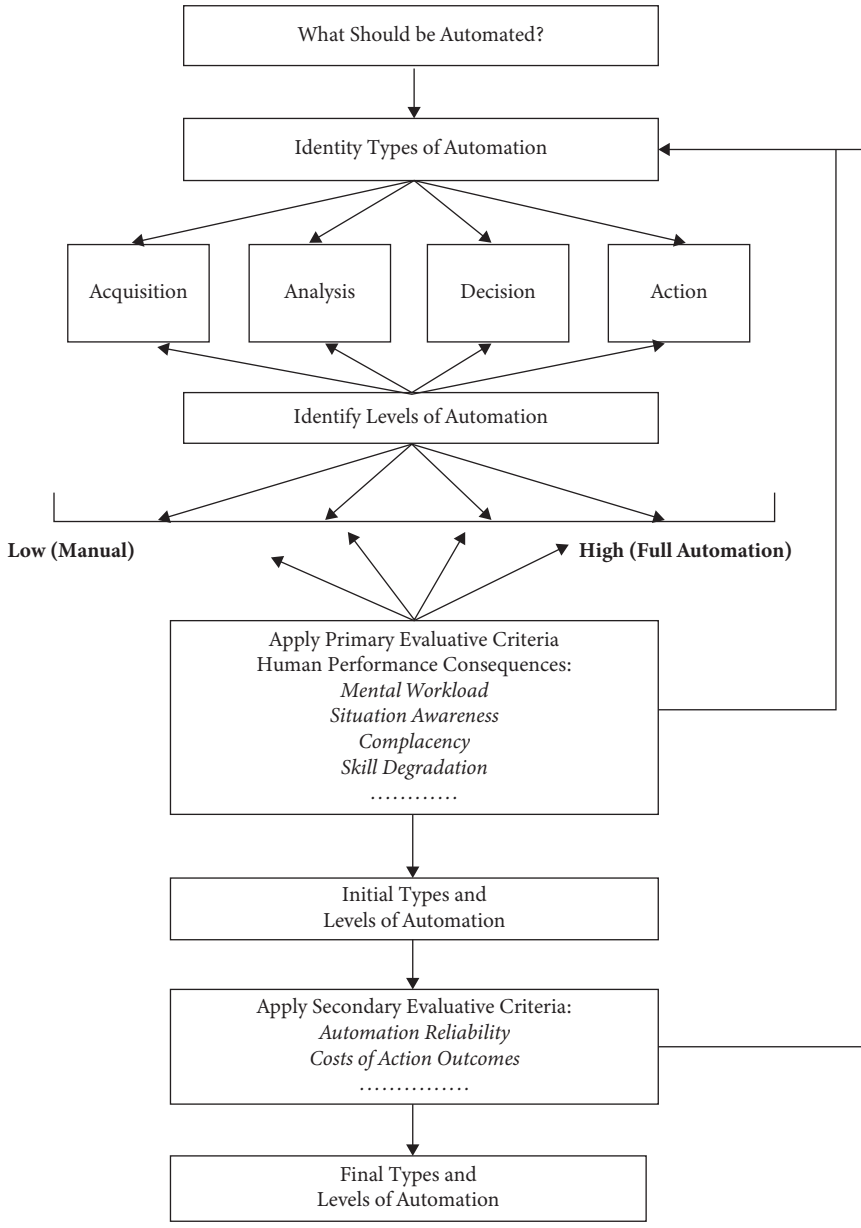


Figure 20.1 A workflow for using the types and levels of the automation framework [639].

20.3 Interactive AI

AI can be used to automate many functions. For example, Microsoft Excel can extrapolate from a user typing numbers in cells. If the user types 10 in one cell and 20 in a neighboring cell, selects the two cells, and extends the selection, the contiguous cells in the extended selection will read

40, 50, and so on. Notably, AI has brought automation to nonprofessional users. In this context, we briefly review four common applications of AI:

1. Augmentation
2. Dialogue
3. Monitoring
4. Recommendations.

First, some forms of AI provide more than just automation; they amplify users' abilities. For example, AI can allow users to type faster and sloppier on a mobile phone keyboard by autocorrecting incorrectly pressed keys. This is an example of AI using a formal model of user interaction to translate observations of user input (the touchpoints on the keyboard) into hypotheses of the user's intention (the intended text).

Second, AI can be used in dialogue interaction where a user engages in some form of conversation with an AI agent. Such a dialogue can enable user interaction with simple commands, such as the user clicking various options. Alternatively, users can use natural language as input when speech recognition is available. Spoken dialogue interfaces recently became popular (e.g., Siri and Alexa). They provide users with direct access to functions and content via spoken language. Text-based dialogue interaction is common in support functions, such as online banking and shopping. With large language models, this type of interactive AI is expected to become more popular.

Third, AI can improve engagement by monitoring the background and reacting at opportune moments. This can be achieved in two different ways, as it can be driven by user or non-user input. A simple example of a non-user input system is a thermostat-based heating system that regulates the temperature in a building or vehicle based on temperature data from a sensor. An example of a system based on user input is a 3D gesture recognition system that monitors the user's hand and finger movements using hand tracking. Such a system looks for specific gesture patterns that trigger specific commands. One commercial example of such a system is the Microsoft HoloLens 2 optical see-through head-mounted display. It uses continuous hand tracking to detect when the user is triggering one of its two recognized key gestures: a pinch, which tells the system to select something, and a bloom gesture, which tells the system to open the system menu.

Fourth, AI can make suggestions to users. Such systems use recommender systems to propose particular solutions. Examples of such systems are car navigation systems, other map-based navigation systems, and systems that provide recommendations on online shopping sites and streaming services.

As these examples demonstrate, interactive AI is already ubiquitous and available for a range of direct and indirect interactions; yet, in many ways, users do not realize this is the case. These examples of deployed interactive AI systems also show that it is possible to design, build, and release interactive AI systems that users can interact with, achieving widespread adoption and use in some cases.

Nonetheless, there are novel and partially unresolved challenges related to interactive AI systems. Some of these challenges are similar to those of non-AI interactive systems. As such, the knowledge and methods taught in this book—from understanding users, interaction, and user research to designing and evaluating a system and its risks—will still be helpful. Other challenges are unique to systems infused with AI, calling for complementary perspectives and methodologies.

20.4 Ironies of automation

In this chapter, we have learned that direct attempts to "replace" humans with automation or AI are likely to fail. Attempts to automate do not replace the human; they only change the type of engagement. Even highly automated systems require human supervision. In her 1983 paper titled "Ironies of Automation," Bainbridge summarized the negative effects of automation on human operators [36]. The irony here is that automation can have the direct opposite effect of what was intended during its design: Instead of relieving a human from chores, automation makes human contribution more crucial than before. Although she studied process control and aircraft pilots, the ironies she identified apply to modern AI, too.

The first irony concerns *deskilling*: When the level of automation goes up, the user is less engaged and may therefore lose both manual and cognitive skills over time. The operator of a chemical plant, for example, may lose the ability to understand how the automated plant works. A pilot may lose the skills needed to fly a plane without automated assistance. Similarly, users who rely on spell checkers may gradually lose their language skills [267]; users who rely on navigation aids may lose their spatial skills [533]. This means the user may not have sufficient skills to recover from a situation where automation fails. This, in turn, means that automation must now be designed in a way that does not depend on user skill, or that users' skills must be maintained by other means, for example, through training.

The second irony concerns vigilance. Vigilance refers to the user's ability to maintain attention on something over time. For example, in semi-autonomous vehicles, drivers are expected to maintain attention on the road and be ready to take over control when needed. In such cases, because the user is not continuously engaging with the task, boredom is likely to occur. Bored users will engage with secondary tasks or just get tired. The irony here is that they are *less* able to intervene than without automation.

The third irony concerns *agency*: While a system may be designed to empower the user, it may reduce the user's felt agency. When a user feels certain that an outcome is due to their action, the user is said to experience high agency. If the outcome seems less connected to the user's action, the user experiences less agency. A system that performs many functions for a user can result in less felt agency, as the user feels less in control. However, this is not always the case: A system can be designed so that users are fully aware of what is going on and can take action at any point. Vice versa, a system performing a single function may result in a low felt agency if the way it is achieved does not provide the user with a sense of ownership of their actions. An example is an auto-aim function in a first-person video game. If the auto-aim function is too generous, the player may not believe they hit the target due to their skills.

Lack of agency is detrimental to well-being in the long term because it contradicts the basic psychological needs for autonomy and competence (Chapter 6). Because of a lack of agency, a worker whose job is partially automated can easily become demotivated. This may be compounded by their professional skills being less appreciated by others. Automation is likely to decrease the perception of the profession: "Are you actually doing anything or is it the automation?" Workers may be less motivated to do their jobs, which then decreases the level of performance of the whole system. This, in turn, can lead to feelings of alienation in users. They may not feel like they have agency in their work and tasks: "It is the automation doing it, not me."

As we learned in this chapter, a central design challenge is providing users with the *right level of control*. Providing complete control may be unrealistic and even defeat the purpose of AI assisting the user to begin with. Moreover, complete control assumes the user has a thorough knowledge of the algorithmic principles that drive the AI, which is practically impossible to achieve. On the

other hand, providing insufficient control may result in the user experiencing frustration when the AI requires steering or even lead to a disaster if the user and the AI work toward opposing goals. An infamous example of the latter is the Boeing 737 MAX disaster. The aircraft used a sophisticated anti-stalling control system that pilots were unaware of [827]. When the system failed due to faulty sensor readings, the pilots effectively worked against the system in trying to stabilize the aircraft.

These ironies of automation also apply to interaction with modern AI-infused systems. Moreover, research on contemporary AI-infused systems has identified one additional challenge: Generally, users and other stakeholders should be able to *understand and trust* the output of AI. Without such understanding, it may be difficult to accept the AI's recommendations or trust an AI that performs actions or makes suggestions that appear nonsensical or unintuitive to the user. The AI's output may even be unsettling: For example, users may find some accurate recommendations creepy because they are based on data harvested from them. Some AI systems may also have important consequences in real life. Consider, for example, systems that recommend insurance fees or potential love partners. A new research area has emerged at the intersection of HCI and AI research called *explainable AI*. See Paper Example 20.3.2.

Paper Example 20.3.2: Explainable AI

AI systems invariably use mathematically complex models, such as deep learning or logic models. Whenever an action or decision has consequences for the users, they may need to understand the AI's reasoning. Explainable AI refers to techniques for explaining the outputs of machine learning models. *Model visualizations* can be used to show how the internal layers of a deep neural network work, for example, which pixels in a photo contribute the most to its classification. The figure in this paper example box shows an example of saliency modeling applied to medical imaging [277]. The figure is used under Creative Commons License. *Interactive techniques* allow users to trial a model, for example, by testing it with different inputs. *Counterfactual explanations* provide "what if?" alternatives to the user. For example, when explaining an insurance decision, the user may be told how the insurance policy or fee could be changed. AI could say, for example, that "By stopping smoking, your monthly fee would go down by 20 euro."

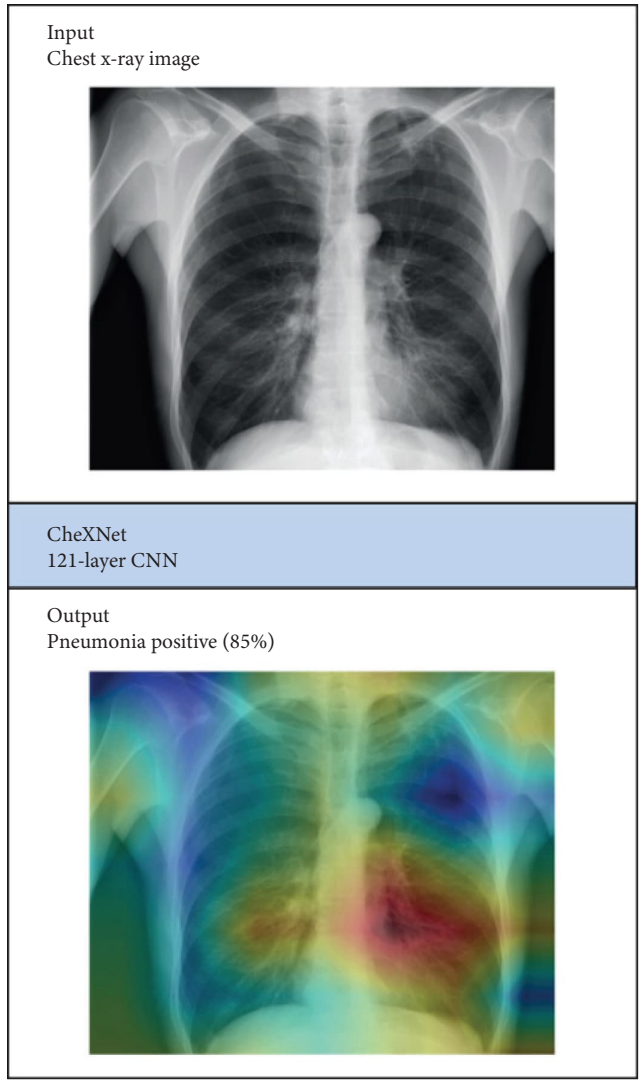
Explainable AI is wickedly challenging to design; so far, no generic technique has been identified. This is well-reflected in clinicians' experiences with saliency-based explanations:

> The clinician cannot know if the model appropriately established that the presence of an airspace opacity was important in the decision, if the shapes of the heart border or left pulmonary artery were the deciding factor, or if the model had relied on an inhuman feature, such as a particular pixel value or texture that might have more to do with the image acquisition process than the underlying disease.

Besides the fact that these models often *do* reason in a nonhuman manner, one persistent issue is that users are not interested in investing time in reading complex explanations. Another is that those explanations may not be understandable. For example, to understand a deep learning model with a random forest model, it is first necessary to understand how random forest models work. While simplifications can aid understanding, they lower the accuracy of the explanations. This is called the explainability–accuracy trade-off.

continued

Paper Example 20.3.2: Explainable AI *(continued)*



Wang and colleagues reviewed cognitive factors affecting people's ability to understand explanations [868]. They pointed out that explanations must be built in a way that appreciates the capabilities and limits of human cognition, which we covered in Chapter 5. They summarized five implications for design:

- Support forward reasoning: Show feature values and attributions before class attribution. This can avoid confirmation bias in users.
- Support coherent factors: Avoid showing factors in explanations that go against their typical relationships.

- Support access to source and situational data: Show the original (full) dataset and its situationally relevant subset if asked by the user.
- Show model uncertainty: Express the uncertainty to the user, for example, in the form of the posterior probability distribution of classes.
- Integrate multiple explanations: Show diverse explanations to provide users with different perspectives.

Summary

- An automation problem is a function allocation problem. It is about determining which functions should be automated and with what type and level of automation.
- Interactive AI enables users to work with AI in many different ways to complete their tasks. Such interaction can be seen as a collaborative activity between the user and an AI agent.

Exercises

1. Understanding human–automation interaction. Consider a display that is meant to assist the user in identifying targets among a set of distractors. An AI system is used to automatically detect targets and visually flag them on the display to assist the user in quickly identifying such targets.
   (a) Suggest preliminary types and levels of automation that can be used to provide such functionality.
   (b) Evaluate the preliminary types and levels of automation using the following primary evaluation criteria: (i) mental workload, (ii) situation awareness, (iii) complacency, and (iv) skill degradation.
   (c) Based on your evaluation using primary evaluation criteria, refine your selection of the types and levels of automation for this design problem.
   (d) Evaluate your updated design using the following secondary evaluation criteria: (i) automation reliability and (ii) costs of decision and action outcomes.
   (e) Consider a radical redesign in which the system is extended to use eye tracking to check whether the user has looked at all the relevant targets identified by the AI system. If the system detects that the user has not looked at all the targets, the system amplifies the saliency of unattended targets to attract the user's attention. The *irony of automation* is that increasing the level of automation may fail to solve automation problems and require even more elaborate system solutions or skilled users to cope with the effects of additional automation. Discuss the extent to which the original system and the extended system risk falling victim to the irony of automation.
2. Types of automation. State and justify the types of automation that may apply to the following functions: (a) autocomplete on a keyboard, (b) a spell checker, (c) a movie recommendation interface, and (d) a self-driving car.
3. Levels of automation. Consider a word processor that provides automated functionality for the following tasks: (a) shortening text up to a set percentage; (b) identifying spelling and

grammar errors, explaining problems, and suggesting in-depth alternative solutions; and (c) automating tasks such as finding appropriate figures and inserting them into a document. Use the types and levels of automation framework to determine the appropriate type and level of automation for each function. Discuss whether certain functions or subfunctions are more suitable for humans to execute than automation.

4. Mixed-initiative interaction. Use the principles of mixed-initiative interfaces to analyze a few representative tasks in the following interactive AI systems: (a) a spreadsheet application, (b) an email application with integrated calendar functionality, and (c) an interface to a "smart" TV. Which principles seem to be followed the most? Which are followed the least? Are there any benefits to incorporating more principles of mixed-initiative interfaces into these systems?
5. Function allocation. Fitts' lists show which things people are good at and which things interactive systems are good at. They serve as a basis for functional allocation. The following table shows such a list from Fitts' early work [242].

Humans are good at	Interactive systems are good at
Ability to detect a small amount of visual or acoustic energy	Ability to respond quickly to control signals and to apply great force smoothly and precisely
Ability to perceive patterns of light or sound	Ability to perform repetitive, routine tasks
Ability to improvise and use flexible procedures	Ability to store information briefly and then to erase it completely
Ability to store very large amounts of information for long periods and to recall relevant facts at the appropriate time	Ability to reason deductively, including computational ability
Ability to reason inductively	Ability to handle highly complex operations, i.e., to do many different things at once
	Ability to exercise judgment

Based on what you know about people and interactive systems, update the list. Also, think about whether functional allocation should be done only based on such lists.

6. Intelligent assistant. A software agent is incorporated into a spreadsheet application. The agent observes the user's work in the spreadsheet and automatically detects when data are suitable for visualization. At that point, the agent proposes the most suitable visualization method for the data. Meanwhile, the user can tell the agent to propose a visualization at any point by clicking on an icon representing the agent. If the visualization is not suitable, the user can ask the agent to generate more suggestions or show a gallery of all possibilities. Based on such interaction, the agent learns the user's preferences. (a) State the type and level of automation of this service. (b) Using the user-centric and system-centric evaluation criteria from the types and levels of automation framework, suggest three potential design issues with this system and justify your choices. (c) Determine whether the system adheres to the principles of mixed-initiative interfaces.