To err is AI

Alba Bisante
bisante@di.uniroma1.it
Department of Computer Science, Sapienza University of Rome
Rome, Italy

Alan Dix
alan@hcibook.com
Computational Foundry, Swansea University
Swansea
Cardiff Metropolitan University
Cardiff, Wales, United Kingdom

Emanuele Panizzi
panizzi@di.uniroma1.it
Department of Computer Science, Sapienza University of Rome
Rome, Italy

Stefano Zeppieri
zeppieri@di.uniroma1.it
Department of Computer Science, Sapienza University of Rome
Rome, Italy

ABSTRACT

In this work, we analyze the different contexts in which one chooses to integrate artificial intelligence into an interface and the implications of this choice in managing user interaction. While AI in systems can provide significant benefits, it is not infallible and can make errors that seriously affect users. We aim to understand how to design more *robust* human-AI systems so that these initial AI errors do not lead to more catastrophic failures. To prevent failures, it is essential to detect errors as early as possible and have clear mechanisms to repair them. However, detecting errors in AI systems can be challenging. Therefore, we examine various approaches to error detection and repair, including post-hoc estimation, the use of traces and ambiguity, and multiple sensor layers.

KEYWORDS

HCI, AI, errors, failures, error detection, error repair, user perception, interaction design

ACM Reference Format:
Alba Bisante, Alan Dix, Emanuele Panizzi, and Stefano Zeppieri. 2023. To err is AI. In *15th Biannual Conference of the Italian SIGCHI Chapter (CHItaly 2023), September 20–22, 2023, Torino, Italy*. ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3605390.3605414

1 INTRODUCTION

> *Emanuele has a car that automatically opens when he takes the driver's door handle with the keys in his pocket. However, one winter day, pulling the handle, he notices it is ice-covered, and the car is not opening. So he had to look for the keys in his jacket pockets and manually open the car.*

Intelligent systems can make a real difference in our lives when they work, but they do not always work.

In this paper, we analyze the different contexts in which one chooses to integrate artificial intelligence into an interface and the implications of this choice in managing user interaction. As a driving example, we will use an application to help users deal with car-related tasks. This application attempts to minimize explicit interaction, so our focus is particularly oriented toward implicit interactions. However, many of the design issues identified appear to have a broader scope. Where possible we will choose the simplest examples that exhibit a particular phenomenon; some are not very "intelligent" but share one or more of the three C's of AI-based systems: Complexity, Uncertainty, or Coadaptation.

The paper will focus on explicating different types of errors, recognizing an error state, and managing and resolving these inevitable situations. We aim to understand how to design more *robust* human-AI systems so that these initial errors do not lead to more catastrophic failures. However, first, we will motivate the general area and case study.

1.1 To err is AI

To err is human – as researchers and professionals in human-computer interaction, part of our expertise and culture is understanding our users' glories and fallibilities. We do not expect our users to perform like automatons; we know there will be lapses of concentration, misinterpretation of data, limited experience, and physical slips.

Effective human-computer interaction design creates systems that work and that are robust despite these occasional human lapses or mistakes.

We design systems with features that scaffold the users' memory, for example, through recognition rather than recall; features that highlight potential slips, for example, spelling checkers; features that reduce the impact of errors, for example, undo; and features that help the users detect and repair problems, for example ensuring rapid feedback on the effects of actions. All in all, these design elements prevent user errors from becoming system failures.

Note the contrast of the human with an automaton: the machine that performs flawlessly time after time, processing billions of calculations, printing millions of payslips; infallible albeit limited and unimaginative. Of course, hardware can fail, especially

CHItaly 2023, September 20–22, 2023, Torino, Italy

ACM ISBN 979-8-4007-0806-0/23/09…$15.00
https://doi.org/10.1145/3605390.3605414

for very large-scale computation. Much of the complexity of cloud-computing infrastructure and algorithms, for example, MapReduce, is about making the overall computer system behave as if it is flawless [8]. The expectation is that the overall system should behave … like an automaton. Recall that HAL, the AI in Kubrick's 2001, becomes homicidal precisely because it had made a mistake and was trying to cover that up.

Of course, we know AI systems are not like this. They are trained on limited data and often use limited sensor data. They are not simply following pre-determined rules but attempting to interpret the environment, particularly the behaviors and intentions of users. The results of an AI system are richer and more nuanced than an "automaton" but, consequently, not flawless.

Effective human-AI interaction design creates robust systems that work, despite these occasional AI lapses or mistakes.

1.2 Implicit Interaction

Nowadays, AI-based systems are considered sophisticated and cutting-edge, therefore, often preferred to more traditional ones. AI allows the introduction of a component into the system that the user perceives as "magic", which usually takes the form of automation of a mechanism that otherwise the user would have had to operate manually. From an interaction point of view, therefore, AI makes it possible to design implicit-interaction-based systems, i.e., systems capable of acting by implicitly perceiving the user's intentions, sometimes even predicting them. All this makes the system even more "magical" and astonishing in the eyes of the user and generally reduces user effort.

Implicit interaction is extremely useful in specific contexts where it is important not to distract the user or it is hard to get the user's attention and the needed time to complete the given task.

1.2.1 When it is a good idea? Using AI-based systems with implicit interaction can be a great idea when the AI system can perform complex tasks that are difficult or time-consuming for users to perform manually. The user experience can be substantially improved by, for instance, chatbots or virtual assistants that can comprehend natural language and offer individualized advice or answers.

Additionally, implicit interaction can be useful in situations where the user is not able or willing to provide explicit input, such as in the case of a driving user.

1.2.2 When it is not a good idea? Implicit interaction may not be the ideal strategy in some circumstances. For instance, specific human input may be required if the system demands high accuracy or precision to guarantee the desired result. Implicit interaction may also be viewed as obtrusive or confusing when the user needs total control over the system.

1.3 Driving examples

The examples we propose in this work are about employing AI solutions to ease car-related tasks. A running example of this situation could be when drivers use certain apps to allow them to complete different parking tasks more easily [4][3][2][15]. Implicit interaction in the context of smart parking can have several benefits, including the reduced requirement for driver attention, increased efficiency in parking space utilization, reduced search time for drivers looking for parking spots, and potentially reduced traffic congestion as drivers may need to travel shorter distances to find available parking spots [1].

In this work, we imagine an intelligent parking app that implicitly understands if the user is driving or not, hence when and where it parks its car, and that shares this information with other interested users (e.g., other family members that share the same car). To extend this example to critical situations, we imagine that the AI underlining the app is sufficiently smart to lock and unlock the car automatically when the owner approaches. Another example of implicit interaction we use is the automatic payment of parking fees. When a vehicle enters a parking space, sensors can detect its presence and record the start time of the parking session. When the vehicle leaves the space, the sensors can detect its departure and calculate the parking session duration. Based on the duration and the applicable parking rates, the AI system can automatically calculate the parking fee and charge it to the user's account without requiring any action from the user.

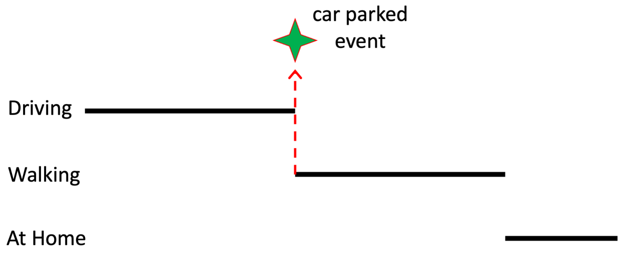


Figure 1: Ideal case – perfect sensing and an event is generated at the moment the user's activity changes

As the most recurring example, we imagine that the user Emanuele is using an AI-based app to track the parking location of his car, and uses it also to share the information with his son.

In the ideal case, the app generates an event if it senses a core change with high certainty. For example, if the system thinks the user has been driving and then they start to walk, a 'car parked' event is generated (figure 1).

Problems arise when raw sensing input is in some way ambiguous. In these cases, there may be a period of uncertainty instead of a direct transition from driving to walking. If this is short, the lower level sensing layer may generate the 'car parked' event. If the uncertainty persists, there may be no natural trigger point and no event generated at all, or, perhaps worse, it is generated so late that a significant error in the parking location estimation is done.

In our example, we imagine Emanuele parks at night, and his son uses the app in the morning to find the car. However, the night before, the sensors on Emanuele's phone were uncertain of its driving state for a considerable time and never generated a 'car parked' event, or generated one significantly misplaced. Emanuele might have remembered for himself where the car was parked, but his son has no idea where it is exactly.

1.4 Paper Outline

This work is organized as follows. Section 2 provides a literature overview of previous human and system error and repair research.

In section 3, we define and categorize errors that can occur in human-AI systems. In section 4, we discuss methods for detecting errors and the roles and responsibilities of the different actors in the detection and repair process. Section 5 explores the challenges of sensing and detecting failures, including cases where sensor data is inconsistent or ambiguous. In section 6, we discuss design strategies for addressing the challenges identified in the previous sections, including improving early detection and communication, ensuring human-AI system reliability, and clarifying responsibility for detecting and repairing errors. Conclusions and ongoing work are reported in section 7.

2 RELATED WORK

Errors in human-AI systems have become a critical topic in recent years as AI systems are increasingly integrated into various aspects of our lives. Despite the significant progress made in AI research, AI systems are still prone to errors that can have serious consequences, particularly in safety-critical applications such as autonomous vehicles, medical diagnosis, and financial decision-making. This has led to a growing body of research focused on understanding the nature of errors in human-AI systems and developing strategies to minimize their occurrence and impact.

2.1 Levels of automation

Before AI was born, several automatic systems were built. The work by Parasuraman et al. [16] proposes a model that describes the different types and levels of human interaction with automation. The model is based on three types of interaction: direct, supervisory, and indirect. Direct interaction involves the human controlling the automation directly, while supervisory interaction involves the human monitoring and intervening in the automation's actions. Indirect interaction involves automation making decisions, and taking actions on behalf of the human.

The model also defines four levels of interaction, ranging from the lowest level, where the human is only observing the automation, to the highest level, where the automation can perform tasks autonomously without human intervention. When considering automobile automation, the Society of Automotive Engineers has made further distinctions and defined five levels [18, 19], which have become influential and have been adopted or adapted by many national and international standards:

However, Shneiderman argues that the one-dimensional view of automation implied by these levels of automation is too simplistic. Instead of a single dimension between human control and computer automation, he suggests considering a two-dimensional framework with higher and lower levels of human control compatible with higher and lower levels of levels of automation [20]. Crucially he considers the point at which *both* human control and computer automation are high, working together, as sweet spot for "reliable, safe and trustworthy" AI systems.

2.2 Intelligent interactions

One of the iconic early uses of AI in user interaction was EAGER (Extraction, Analysis, and Generalization Environment for Repetitive tasks), which allowed users to program repetitive tasks by example [7]. EAGER is based on the idea that users can demonstrate how to perform a task once, then the system will automatically extract the relevant information and generalize it to perform the task in other instances. The paper describes the design and implementation of EAGER, including the algorithms used to extract and generalize examples, and presents several case studies demonstrating the system's effectiveness. The authors argue that EAGER has several advantages over traditional programming methods, including ease of use and increased productivity. Since EAGER there has been continuous work within the intelligent user interfaces community, albeit until recent years more limited uptake in deployed systems.

In a paper from 2017 [17], Human Information Interaction (HII) refers to the process of humans interacting with information to achieve a specific goal. This can involve searching for information, processing it, and using it to make decisions. HII is a complex process that can be influenced by a wide range of factors, including individual differences, the nature of the task, and the characteristics of the information itself. Crucially HII and AI have increasingly overlapped in areas of big data analysis and systems using big data to generate models for intelligent interactions.

2.3 Errors in human-AI systems

We now proceed to review the existing literature on errors in human-AI systems, with a particular focus on the causes of errors, the types of errors that can occur, and the approaches that have been proposed for mitigating errors in these systems.

Errors can occur in human-AI systems when there is a mismatch between the expectations and capabilities of humans and AI. Errors can arise for various reasons, such as data bias, lack of transparency in decision-making, or miscommunication between humans and AI systems. For example, AI systems may make errors in image recognition tasks when they encounter images that are different from the ones they were trained on or when used in contexts that were not anticipated during their development. Human users may also make errors when interacting with AI systems, such as misinterpreting the system's output or failing to provide the system with the necessary inputs.

To minimize errors in human-AI systems, it is important to design AI systems that are transparent, explainable, and robust to variations in data and context. It is also important to ensure that humans are properly trained to interact with AI systems and understand the limitations and capabilities of these systems. Ongoing monitoring and evaluation of AI systems can help identify and address errors as they arise and improve their overall performance and reliability.

There has also been research regarding Second Language Learning [11] and Error Remediation in those systems using Artificial Intelligence. AI systems can be trained to analyze learner performance and identify patterns of errors that are common among learners. This analysis can be used to develop targeted interventions to help learners improve their language skills. For example, an AI system may identify that a group of learners is struggling with a particular grammar rule and provide them with additional exercises or feedback to help them master that rule.

To be effective, AI systems used for second language learning must be able to identify errors accurately and provide appropriate feedback to learners. This requires the system to be trained on large learner performance data datasets and recognize subtle variations in language use that may indicate errors.

Errors can occur in human-AI systems used for second language learning if the system is not adequately calibrated or cannot recognize the full range of errors that learners may make. For example, an AI system may fail to recognize errors unique to certain dialects or resulting from interference from a learner's first language.

It is important to continually evaluate the AI system's performance and make adjustments as needed to minimize errors in human-AI systems used for second language learning. This may involve retraining the AI system on new data or adjusting the AI system's algorithms to recognize certain errors better than others. Additionally, it is important to provide learners with opportunities to interact with human teachers or tutors who can provide additional feedback and support to help them overcome errors and improve their language skills.

2.4 Human and system error and repair

Preventing and dealing with user errors has always been a central part of the HCI literature. For example, two of Nielsen's heuristics are "Error prevention" and "Help users recognize, diagnose, and recover from errors" [13]. The first concerns scaffolding, such as using fixed sets of options rather than free typing or 'recognition rather than recall' [21], whereas the second concerns what happens after an error occurs.

The latter is closer to the main focus of this paper, and the importance of being to tell *that* something has gone wrong is central in Norman's influential seven-stage model [14]. Three stages are about the user working out what to do and doing it, but three are about assessing the outcomes of their action on the system, that is *feedback*. The model makes distinctions about problems at different levels, most importantly between slips and mistakes. The former, a slip, is when the user's intended action is correct, but there is a problem in executing it, for example, pressing the wrong key. The second, a mistake, is when the users' fundamental model of the system state or behavior is incorrect. So they formulate an incorrect action, for example, that they think ctrl-U means "undo".

Crucially, the ability to perform this assessment and evaluation depends on the system giving timely and informative feedback. Not surprisingly, this is a key part of classic user interface design, for example, "Visibility of system status" in Nielsen's heuristics [13] and "Offer informative feedback" in Shneiderman's Eight Golden Rules [21]. These do not prevent things from going wrong but mean that errors are noticed and thus can be fixed. Shneiderman's "permit easy reversal of actions" [21] is precisely addressing this ability to recover.

Human communication is rarely problem free; we mishear and misunderstand one another and yet manage. This is in part due to processes of *repair*, where we realize problems have occurred and deal with them, but most often in ways that do not interrupt the overall flow of the conversation. Frohlich used conversational analysis of human-human repair to inform the design of human-machine dialogues [12].

In conversation and other aspects of life, timeliness is critical for repair; it is usually far easier to correct errors as soon as they happen than later when there may be further knock-on effects of the error. Some years ago, Stephen Brewster noted an expert slip with on-screen buttons. The expert user would occasionally not properly press the button. Still, precisely because they were experts, they did not pay attention to the semantic feedback and only noticed too late that the error had occurred. Adding appropriate sound did not prevent these errors from occurring, but it did mean that they were immediately noticed and hence could be repaired [5].

The middle stage of Norman's seven-stage model is system execution, which, as noted previously, is often assumed to be flawless, at least in execution, if not in design. Some design approaches to "recognize, diagnose, and recover from errors" apply equally well to AI and user errors. However, there may be additional problems as the AI errors may not be immediately apparent to the users.

Seamful design [6] addresses this by embracing the deficiencies in sensing and system behavior and bringing them to the surface as an explicit part of the interaction. This is often used in an entertainment context, for example, the early work using gaps in WiFi as part of gameplay. However, it also has more prosaic applications, for example, the fact that a mobile phone shows signal-strength bars allows various ameliorative actions, such as standing closer to a window.

Appropriate intelligence [10] suggests that limitations in AI performance should be managed by embedding the intelligent algorithms within interactive contexts that make the errors in the AI less damaging, for example, using easy-to-accept suggestions rather than full automation. Formal techniques for the design of sensor-rich IoT systems [9], expand on this by modeling both human activity and sensor limitations and then attempting to match the certainty of sensing and the varying consequences of errors in different situations to create rules and set thresholds, which are highly likely to be correct in the most critical situations, whilst giving 'best efforts' where it is less critical.

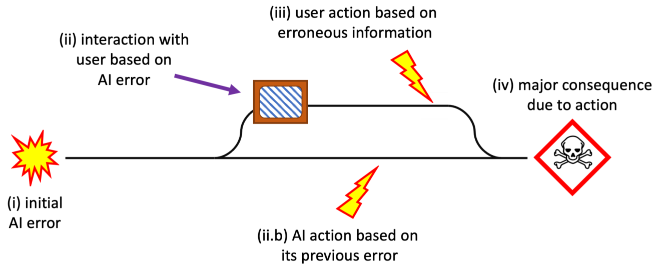
3 DEFINE ERRORS AND FAILURES

Accuracy and other metrics are crucial in designing ML/DL models, but models usually have less than 100% accuracy. Indeed, AI can make a wrong assumption about the context or user behavior due to bad sensing or deductions.

This section will look at different kinds of AI errors that may occur, and different dimensions in which to classify them. In fact, an error in the strict sense of the term is to be considered a malfunction of the system with respect to the design expectations, but within a complex system, errors can be defined from several points of view. In this section, we distinguish between observable and unobservable errors (3.1) and look at the differences between errors of omission and commission (3.2). Finally, we provide our own classification based on users' perception of errors and their changing expectations and behaviors (3.3).

3.1 Observable errors

When the AI gets something wrong, that mistake may not cause a problem for the user, let alone a system failure. Based on the deductions and calculations of the AI, subsequent actions are usually programmed for the system's functioning. However, not all of them directly affect the user and their system experience. Based on this



opportunities for correction		
	(i)–(ii) and (i)-(iib)	– no externally observable impact
	(ii)–(iii)	– prevent problematic user action
	(iii)–(iv) and (iib)–(iv)	– relatively easy repair
	after (iv)	– increasingly hard repair

Figure 2: Error Timeline

consideration, we distinguish between observable and unobservable errors on the user's part. Figure 2 schematizes the possible flow of actions that derive from a system's decisions made based on an AI error.

The system's functions can be divided into background and foreground.

Background actions happen when the system seems to be doing nothing, such as when the smartphone is in the user's pocket. These functions can be activated because they are programmed, but the context and the environment can also trigger them. On the other hand, when the system does something and exhibits observable behavior, such as updating the user interface, these results fall under the category of foreground actions.

When an AI makes a computation, such as a context assessment, whether the result is right or wrong, background and foreground actions can be triggered. If the AI makes a mistake and the consequent action is in the foreground, this will result in an error observable to the user. As suggested in the upper path of Figure 2, an observable error can also induce the user to propagate the error with one or more new actions. Similarly, a background action triggered by an error condition is also an error, but in this case, it is not user-observable and will not initially generate additional error conditions.

Please note that an unobservable AI error can become observable later, impacting the consequences, as discussed in the following section 4. Moreover, in this phase, we distinguish between the possibility for the user to detect and intercept the error, and not between detected and undetected errors.

Also, note that some AI errors could generate erroneous actions within the same system that are observable in some cases and for some users but unobservable for others. Consider the example presented in 1.3. Assume that Emanuele parks his car, but the AI wrongly register its location due to some sensing errors or uncertainty. This error is a background error for Emanuele, who can proceed and get home without perceiving any errors in his user experience. But later, Emanuele's son needs to get the car, so he walks to the wrong car location to discover it is not there. In this second case, the action resulting from the AI error becomes observable and actively damaging.

This dimension of distinction, observable or unobservable, is intuitive but fundamental for introducing an error's criticality level. Closely linked to the observability of an error is the possibility of intercepting and repairing the error.

3.2 AI errors of commission and omission

For human errors, one often distinguishes errors of commission: things done wrongly, and errors of omission: things that are not done when they should have been. Both can have deleterious consequences, but typically errors of commission are regarded as more severe or blameworthy. Similarly, an AI can act in error by doing something wrong or doing nothing (when supposed to do something). For example, the AI inaccurately estimates Emanuele's car parking location and wrongly modifies the UI to inform his son about it (commission error). On the other hand, the AI may incorrectly assume that Emanuele is still traveling, not informing his son that the car is free to use (omission error). Table 1 describes the four possibilities of an AI taking actions that could be either correct or wrong based on the design expectations.

	Expect something	Expect nothing
Does something	correct/wrong	wrong
Does nothing	wrong	correct

Table 1: AI actions based on design expectations

Note we are referring here to design expectations. Ideally, the user will understand the system; hence, design and user expectations are the same. Still, in practice, user expectations could be misplaced: for example, if the user believes the system can do something it cannot.

Note, too, that this will interact with whether the AI errors are observable. Typically, wrong actions (commission error) will be observable, but erroneous inaction (omission error) may not be noticed, even if the user expects the outcome. For example, the user may expect the system to identify free parking spots vacated by other app users; however, if the system fails to notice these parking spots, the user would hardly tell if there are no parking spots or if the AI did not notice them.

3.3 Users perception of errors, changing behavior and expectations

It is interesting to highlight the many degrees of perception the user may have toward AI errors before discussing their likelihood of occurrence and the value of error recognition in preventing more severe failures. Below, we discuss the concepts of detection, perception, and understanding.

- Detection is the ability of a human to notice when an AI system makes an error. This may involve recognizing a discrepancy between the expected and actual output of the system or identifying patterns or trends that suggest a problem. For example, Emanuele notices that the app registered their last parking spot in the wrong location.

- Perception is the ability of a human to interpret and make sense of the error detected in the AI system. This may involve having an idea of the context and predicting the consequences of the error. For example, Emanuele predicts that if the app shows the parked car in the wrong location, it may fail to track when the car's location changes again.
- Understanding is the ability of a human to comprehend the underlying causes and factors that may have contributed to the occurrence of the error in the AI system. This may involve knowledge of the technical aspects of the system, as well as the broader social, ethical, and legal implications of the error. Understanding an AI error may also involve identifying potential solutions or strategies for preventing similar errors from occurring in the future. For example, suppose Emanuele understands that the AI system fails to correctly locate the parked car when there is a poor connection, like in the underground parking lot of his home. In that case, he may manually register the car's position every time he gets home.

Based on that and what was introduced in the previous sections, we propose an additional error classification based on user perception, understanding, and changing expectations and behavior. We, therefore, distinguish errors into three types:

- Type I – The AI does something that causes problems in 'normal' (pre-intervention) behavior, but the user still does not understand the system enough. This type of error is unobservable until very late, when consequences may be costly. For example, the car unlocks as Emanuele passes it (maybe simple proximity-based switching), but he is rushing to a shop, then a thief notices, opens the door and steals something. Emanuele did not understand that proximity would unlock the car, so he could not detect the error. Hence it leads to failure.
- Type II - The user has begun to build a perception of the AI and has expectations about what it will do, but it does something different. For example, Emanuele reaches for the car door handle, expecting it to be unlocked, but it is still locked, and he breaks his fingernails. The user has a level of understanding of the system, but it is not enough to prevent failures.
- Type III – The user's fundamental model or knowledge of the world has changed due to a smart app/environment, and they are less able to do something—for example, not being able to find their way in a well-known city without a navigation app because they have become used to simply following directions. In Type III, the user's expectation is that the AI does not make errors, but there is no AI. In this case, the user can have a complete understanding of the system or not, but the accent is on the complete trust that the user has in the system.

In the next section (4), we will consider how an AI initial error may or may not lead to critical consequences, i.e., whether *AI error* leads to *system failure.* A failure impacts the human-AI system and possibly the user's life, and a non-negligible cost may be required to repair it. Typically, a failure comes from the user's or AI's wrong actions misdirected by an initial AI error that cannot be undone.

4 ERROR DETECTION AND REPAIR TO PREVENT FAILURES

We have outlined a wide range of different types of AI errors. We saw in Section 2.4, when discussing human error, that early detection and repair are at the heart of preventing minor slips from becoming major problems. The same is true when we look at AI errors. No matter the course of the error, the earlier problems are found, the more likely they are to be fixable.

4.1 Importance of detection

Detection, by the human or the AI, is the ability to notice when an AI system makes an error. This may involve recognizing a discrepancy between the expected and actual output of the system or identifying patterns or trends that suggest a problem. As anticipated, the importance of detecting an *error* lies in the possibility of repairing it and taking an alternative action. Otherwise, an undetected error is likely to become a *system failure* - a significant error that impacts the user experience and may require a non-negligible cost to be repaired. Referring to the above example in section 1.3, suppose Emanuele's son can detect the error. In that case, the failure may be prevented: for example, if the AI notified him when the parking location was not detected, or its estimation is not accurate enough.

On the other hand, the repair is impossible if the error is undetected. Eventually, the user will acknowledge it, but it will have already become a failure. In our example, Emanuele's son would walk to the wrong parking location to discover that the car was not there.

Of course, the repairing action itself may be costly, and the timing of the detection is relevant: an early detected error is usually reparable with lower costs than a late detected one. For example, if Emanuele's son somehow realizes that an error occurred and is in a hurry to get the car, he can wake up his father to know where the car is actually parked.

Effective repair makes an AI system robust, and detection is a necessary condition for repair.

4.2 Detection and Repair - who does what?

We have seen that errors or inaccuracies are often inevitable; detecting them and repairing any consequences before they become severe is essential.

In a robust human-AI system, two main aspects must be addressed: who *detects* the problem and who will *repair* it. Indeed, both humans and AI can weigh in to detect and repair errors; it is left to the designer to choose if the agent detecting the error is the same as the one that repairs it or not.

When the user is in charge of detecting and repairing the error, a robust design should aid human detection by explicitly displaying the system state. In our example, the AI may notify Emanuele about the registered parking location, so that he could check it and fix it before going to sleep.

If the AI should detect and repair this error, it may take into account additional information gathered after the event generation. For example, if the AI is wrongly assuming that Emanuele is still driving, but then detects that he is actually connected to his home

WiFi - hence detecting the error, it can autonomously make a new estimation about the car's parking location.

When the agent detecting the error is not the same as the one that needs to repair, there must be some communication between the two.

In our example, two possible design solutions are:

- Human detects, AI repairs – Emanuele notices that the car parking position on the app is wrong and give this negative feedback to the AI, which will include it in a new estimation of the car parking location.
- AI detects, Human repairs – The AI might see that its estimation accuracy is under a certain threshold, and alert Emanuele, who selects the correct car parking position on a map.

Understanding these different options allows us to consider different potential paths to error detection and recovery; for example, the user detects – user tells the AI – AI fixes; or AI detects – AI tells user – user fixes. There needs to be AI-state visualization or various forms of user interaction within these places. We will return to several of these when we discuss design implications in Section 6. The AI system is intelligent and changes potential user interactions with it. Several of these interactions are similar to the corresponding cases of detection and repair when the primary error was due to a human mistake or misunderstanding (often itself due to a design error).

Note that, in this section, we have lightly made the assumption that AI can detect errors. In reality, designing and developing the AI detection process can be much more complicated than human detection, which can also be based on perception and understanding. For this reason, we have dedicated the following section to the particular case of how AI can detect errors.

5 SENSING ERRORS

It is reasonable that a computer system that is or isn't equipped with AI can detect certain forms of inconsistency or problems with user input as it can take in different factors; for example, a spelling checker may not know what the user meant to write but can tell that the word typed is not in the dictionary. In contrast, if the computer system can detect its own error, why couldn't it simply do it right before making the error? Typically the answer to this lies in the changing information available to the system both from the user and the environment. In this section, we look at how this can be achieved by combining knowledge of behavior and the varying degrees of certainty of sensor data and inferences.

5.1 Times and information

In most AI inference/estimation tasks, we are considering two main time points:

- the time when the inference/estimation is made
- the time the inference/estimation is about

Figure 3 shows the time of estimation on the horizontal axis and the time it is about on the vertical axis. Points along the diagonal represent point estimates. When the time of inference is the same as the time it is about, the AI system is using its sensing data to make an inference about the current state of the world. For example, if the AI detects that Emanuele has got in his car, it may turn on the radio.

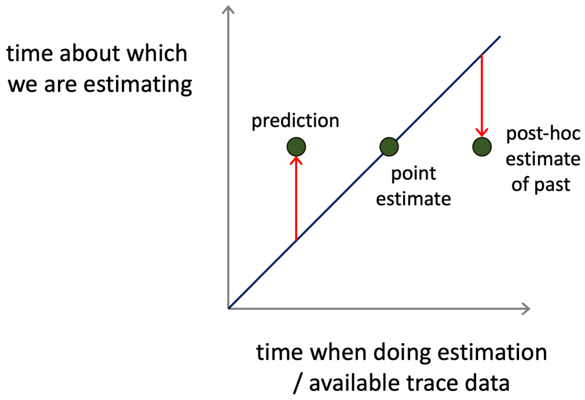

time about which we are estimating
prediction
point estimate
post-hoc estimate of past
time when doing estimation / available trace data

Figure 3: Two times: time of estimation vs. time about which we are estimating

In contrast, points above the line represent situations when the thing being estimated is in the future; that is a prediction. In the parking context, this may be when Emanuele walks out of his office and heads toward his car. The system may predict that he will drive away and therefore alert another driver that the parking space will become available.

Finally, points below the time represent post-hoc estimates about the past. These do not immediately sound useful, but consider the example above. Even if the system does not detect the correct moment when Emanuele parks his car at night, the important thing is that the correct location is given to his son in the morning.

Crucially, more information may be available when the system makes inferences later. In general, the later an estimation is made, the greater its accuracy. This can happen for a prediction; for example, if Emanuele walks towards his car and turns into a shop, the prediction of him freeing his parking spot will change. It can also happen retrospectively, if, for example, the car parking estimation can correct itself overnight before Emanuele's son looks for the car.

In the former case, it is clear that new information is available (the user walking into the shop). Still, it is unclear what additional information would be available for post-hoc correction of the vehicle position. Sometimes, there can be delays in sensor information or other data becoming available; this is common in some application areas; for example, if some sensors are not attached to permanent networks, or there is a need for raw sensor data to be processed, such as the parking application.

For other applications, no new sensing about the time of interest is available after the event. Happily, knowledge of future states can be combined with models of human activity to improve post-hoc estimates.

5.2 Post-hoc estimation – traces and inconsistency

In some cases, we can create state models of normal behavior with valid traces, such as:

… driving <car parked> walking …

These might come from ad hoc modeling or machine learning inference from user traces.

This model can be used for prediction, especially in cases with additional probabilistic knowledge. For example, if Emanuele is known to be at home (based on WiFi reception) at 2 am, they may be assumed to be sleeping and unlikely to drive again before the morning. However, these models can also be used to 'play backward', inferring past states, especially when sensing data is inaccurate or incomplete.

If the system comes into a certain state, it may be possible to identify strong inconsistencies. For example, if the state was 'driving' followed by an uncertain state and then definite 'walking', then the system knows that a parking event should have happened in between, even though it is unclear where. That is, the system has detected a certain failure, albeit potentially a considerable time after the event.

This does not help correct any past actions such as, but it can help later. For example, this could be conveyed via a 'best guess' estimate, such as the last point when the system was deemed to be driving, or by one that explicitly conveys uncertainty, such as showing the set of recorded positions between the last definite driving location and the first definite walking one.

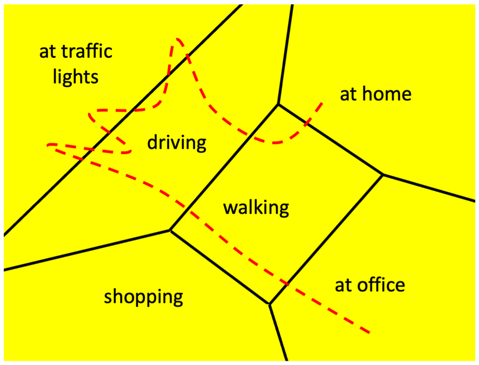


Figure 4: Unambiguous sensing

5.3 Levels of ambiguity

Figure 4 shows the ideal situation where sensing is unambiguous. The overall area represents the set of all possible sensing values. For each state of the real world, sets of unambiguous sensor readings correspond to the state, and each trace of user activity moves ambiguously through a consistent path of states. The dashed line shows an example trace, where Emanuele is at his office one evening, walks to the car, drives, periodically stops at traffic lights, and then walks home.

However, sensing is rarely as perfect or accurate as this. In practice, there are levels of ambiguity as shown in Figure 5:

certain This is a state in which the AI is confident about detecting the context. Considering smart parking, we can be fairly certain that Emanuele is driving if there are engine-like vibrations, the GPS shows rapid movement, and there is relatively little body movement. Alternatively, if there is minimal engine noise, slow GPS movement (1–3 km/h), and periodic (approximately 1 Hz) body movement, the AI can safely conclude that Emanuele is walking. Every hint that can help the system stay in or return to this state, is welcome.

uncertain This is when there are some possible unusual behaviors and the system is not completely confident but within the range of possible variation of the previous certain state. The default behavior assumes the system is still within the previous state. An example would be if Emanuele is walking and stops momentarily to look in a shop window or chat with a friend – for a short while, there is no GPS movement, low body movement, and no engine noise.

incoherence Here the AI is unsure as the sensor readings do not match the typical variation of any usual behavior. An example of this will be if there is both engine noise and periodic leg movement. If these states are transient, they would be filtered out, but if they persist for any length of time, they suggest a state that is totally unknown to the system; maybe Emanuele is dancing in the car seat while waiting in a long road queue. This may simply be recorded internally as an unknown state, but if any critical action is to be taken, this may be a time to warn the user explicitly before it's too late.

For obvious reasons, we informally refer to these as the egg yolk, egg white, and frying pan, respectively.

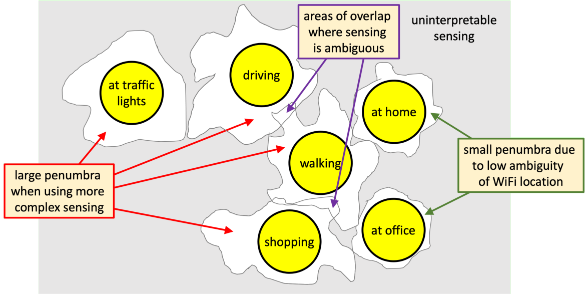

areas of overlap where sensing is ambiguous
uninterpretable sensing
at traffic lights
driving
at home
walking
small penumbra due to low ambiguity of WiFi location
large penumbra when using more complex sensing
shopping
at office

Figure 5: Ambiguous and incoherent sensing

Note, we may also have continuous levels of certainty rather than three fixed levels, but using distinct regions can help us think about detection and intervention strategies. Also, we can think of these instantaneously, but they have a temporal nature; for example, their 'stationary in a car' sensor readings that last a long time might be regarded as unusual or incoherent, even if short stationary periods are common.

5.4 Using traces and ambiguity for post-hoc detection

We return now to the parking at night example in section 1.3 and see how the knowledge of common traces and ambiguity levels can help the AI system to better deal with errors and offer the potential for AI or human repair. Figure 6 shows the scenario with the period of uncertainty and the normal variation levels (the egg white region) for the driving and walking states.

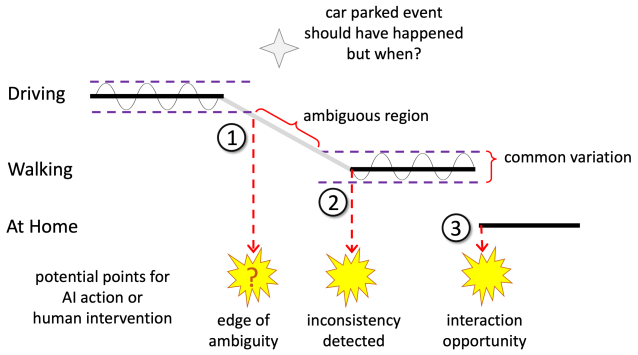


Figure 6: Difficult case

In the ideal situation (figure 1), the parking event was generated when the state changed from driving to walking, but in this case, there is no clear point of change.

In Figure 6, point (1) is the first time the sensor readings fall outside the common variation of driving and, thus, the first point at which the AI system's model of the world is incoherent. AI's information is not enough to generate a parking event, and potentially it is missing to detect a parking location; the consequences of this error are quite important in our example. This could be a point at which the AI system requests an explicit confirmation. However, we normally wish to avoid this in implicit interaction.

Point (2) is the first time the system can inform Emanuele that something has gone wrong and that there is an inconsistency in its past sensing. There has been a transition from driving to walking, but no parking event has been generated. This could again be a chance for subtle and ignorable user interaction for clarification. Alternatively, the AI system could take corrective action itself as suggested in Section 5.2, but using the knowledge of normal levels of variation to make estimates more accurate. As noted, up until point (1), it is fairly safe to assume that Emanuele was still driving. Similarly, the time at which Emanuele can be confidently inferred to be walking can be pushed back to the first point the sensors enter its range of variation. The period of uncertainty and hence the geographic inaccuracy has been reduced.

Finally, at point (3), Emanuele is at home. Even though this is somewhat removed from the point of ambiguity, it is the first time he is in a context that makes explicit interaction more acceptable. Of course, he may have forgotten the parking location by this time, but it is better to clarify this at point (3) than when in a hurry in the morning.

5.5 Dealing with sensor-layer failure

In a safety-critical situation, we might want to report every low-level sensor failure and perhaps get human confirmation or overrides. Typically we do not want this for most applications. Furthermore, the knowledge of the right level of warning/request for user intervention needs application knowledge. Architecturally there needs to be rich enough API between layers to enable higher levels of application software to decide which lower-level failures matter and how to weigh up the severity of the problem (for the user) with the certainty that there is a problem.

In the case of an incoherent trace, we have a high certainty that a problem has occurred, but it may be some time after the actual important event. In the car parking example, Emanuele might get a prompt from his parking app when it is certain he is walking, or if not before, when he is at home (as sensed by GPS, WiFi signal). When asked at this point, he is more likely to remember than the next morning. Also, repairing at that point might avoid wasting time for his son, who may be in a hurry the next morning.

The ambiguity region is a little more complicated. The ambiguity means we are uncertain of the state, and it is possible that interaction at this point would actually cause an interruption, maybe while Emanuele is close maneuvering while parking. However, this is also the ideal point to request user information as it is closer to the real change point.

6 DESIGN IMPLICATIONS

Understanding this understanding of AI errors leads to a high-level design strategy. In general, it poses questions to ask, given a potential error or inaccuracy with the system; more detailed heuristics will depend on particular situations.

6.1 Helping early detection

As we have seen, early detection makes effective repair more likely. We have two design options:

- help the user – Provide appropriate visualization, audio, or other feedback or status to the user so that they are more likely to detect a problem.
- help the AI – Create mechanisms so that the AI system can detect its errors. This will typically happen after some time when more information becomes available.

Sometimes one or other design options may be easier to implement or more effective. However, note they are not mutually exclusive, and both can be employed. For example, in the case of the parking app unlocking the car as the user approaches, this could immediately produce a small notification sound. However, if this is missed, the system might still notice if the user has walked away again.

6.2 Helping communicate

In the case when detection and repair are performed by different agents, we need to seek appropriate communication options:

- user to the AI – When the user has detected a problem, but the AI needs to correct it, we need to find easy ways to let the system know. Ideally, this can be designed to make use of the fact that the detection will be due to a recent notification or status change so that it can be highly contextual and not require extensive interaction.
- AI to the user – When the AI has detected its own error but needs to inform a user, this needs to balance the need to inform the user as soon as possible to enable repair before problems get worse while being subtle enough to avoid distracting the user, or being tedious.

6.3 System Reliability - when accuracy may become a problem

Another aspect that designers should be concerned with is the level of reliability the user expects from the AI. This expectation is usually accumulated by the active experience of the system, as after a certain period of use, the user will begin to accumulate knowledge of the system, experiencing situations in which the AI will work more or less accurately. Of course, one of the goals is for the user to trust the AI - and we certainly want to avoid them discovering situations of unreliability in critical contexts. In our car example, it would be better for the user not to discover that the car unlocks itself as they walk past it, only when the thief has already stolen it. A possible design solution would be an AI that advertises its real level of accuracy, to avoid deluding the user, but manifesting a certain level of uncertainty can be highly detrimental to building user trust, leading them to abandon the system quickly. In short, the designer has to deal with the honest reliability/user trust trade-off. In any case, we can distinguish two situations:

6.3.1 Low reliability. When an AI has relatively low reliability, the users expect this unreliability.

Here the rules of appropriate intelligence come in an ‘active’ way. The unreliable AI should not actively do something hard to repair – for example, interrupting important work or rewriting text without asking. However, the user monitors things, so situation errors that go undetected are rare – the human ensures detection. In the car door example, if the system opens the door 70% of the time, but the other 30% fails to detect, the driver may gently try the door handle before pulling hard.

6.3.2 High reliability. Sometimes the worst problems occur if the system is very reliable, perhaps correct 99% of the time, or 99.9% of the time, but still occasionally gets things wrong.

In these cases, the human comes to expect the AI to behave correctly and therefore is unprepared for failure. For example, if the AI detects paid parking areas with 99.9% accuracy and pays automatically, the driver will get used to not checking if the payment occurred and may be fined when the AI fails.

6.3.3 Responsibility for detection. In the low-reliability situation, the human effectively takes responsibility for monitoring the system’s behavior (without necessarily even being aware that is what they are doing).

In a high-reliability situation, the human will be unlikely to notice, and therefore we need to design strategies that help alert the user to unexpected situations.

For example, suppose the user is walking very quickly along the pavement towards their car and don’t appear to be slowing; the parking detection system might assume they will not enter the car and so not unlock it (precautionary principle for avoiding theft). However, if the user stops suddenly (perhaps was just in a hurry or almost missed noticing which of the line of cars was theirs) and reaches for the car door, we will likely get the broken finger-nail situation! To avoid this, the car app could detect that while its action is NOT to unlock, it is a potentially ambiguous situation and so do something to warn the user, perhaps vibrate their phone (a sort of ‘hello’ from the car as they walk past) or make the car handle glow red.

7 CONCLUSIONS

This paper has highlighted the importance of considering errors and failures in human-AI systems and the challenges involved in detecting and repairing them. It is clear that AI systems are not infallible and can make errors of commission or omission that can have significant consequences for users. The perception of errors can change over time, and users may develop new expectations and behaviors in response to errors.

To prevent failures, it is essential to detect errors as early as possible and have clear mechanisms to repair them. This requires collaboration between human and AI and effective communication and feedback mechanisms to ensure that users are aware of errors and understand how they can be repaired.

The research on human-AI interaction has shown that detecting errors in AI systems can be challenging, and there is no one-size-fits-all solution. Therefore, it is essential to have a range of approaches available, including post-hoc estimation, the use of traces and ambiguity, and multiple sensor layers to detect and repair errors in AI systems.

Designers of human-AI systems must consider the challenges involved in error detection and repair and design systems that are resilient to failures and can adapt to changing user expectations and behaviors. This may require new system design approaches, such as using multiple sensors, or the development of more sophisticated algorithms for error detection.

Finally, it is essential to understand how errors and failures in AI systems affect user trust and confidence. Rebuilding user trust after errors or failures occur can be challenging. Research into how users perceive AI errors can help inform strategies for rebuilding user trust.

REFERENCES

[1] Enrico Bassetti, Andrea Berti, Alba Bisante, Andrea Magnante, and Emanuele Panizzi. 2022. Exploiting User Behavior to Predict Parking Availability through Machine Learning. *Smart Cities* 5, 4 (2022), 1243–1266. https://doi.org/10.3390/smartcities5040064

[2] Enrico Bassetti, Alessio Luciani, and Emanuele Panizzi. 2021. *ML Classification of Car Parking with Implicit Interaction on the Driver’s Smartphone.* 291–299. https://doi.org/10.1007/978-3-030-85613-7_21

[3] Alba Bisante, Venkata Srikanth Varma Datla, Stefano Zeppieri, and Emanuele Panizzi. 2022. Implicit Interaction Approach for Car-Related Tasks On Smartphone Applications - A Demo. In *Proceedings of the 2022 International Conference on Advanced Visual Interfaces* (Frascati, Rome, Italy) *(AVI 2022)*. Association for Computing Machinery, New York, NY, USA, Article 78, 3 pages. https://doi.org/10.1145/3531073.3534465

[4] Alba Bisante, Emanuele Panizzi, and Stefano Zeppieri. 2022. Implicit Interaction Approach for Car-Related Tasks On Smartphone Applications. In *Proceedings of the 2022 International Conference on Advanced Visual Interfaces* (Frascati, Rome, Italy) *(AVI 2022)*. Association for Computing Machinery, New York, NY, USA, Article 39, 5 pages. https://doi.org/10.1145/3531073.3531173

[5] Stephen A Brewster, Peter C Wright, Alan J Dix, and Alistair DN Edwards. 1995. The sonic enhancement of graphical buttons. *Human—Computer Interaction: Interact’95* (1995), 43–48.

[6] M. Chalmers, I. MacColl, and M. Bell. 2003. Seamful design: showing the seams in wearable computing. In *2003 IEE Eurowearable.* 11–16. https://doi.org/10.1049/ic:20030140

[7] Allen Cypher. 2000. Eager: Programming Repetitive Tasks By Example. *Proceedings of CHI’91* (04 2000). https://doi.org/10.1145/108844.108850

[8] Jeffrey Dean and Sanjay Ghemawat. 2008. MapReduce: Simplified Data Processing on Large Clusters. *Commun. ACM* 51, 1 (jan 2008), 107–113. https://doi.org/10.1145/1327452.1327492

[9] Alan Dix. 2017. Activity modelling for low-intention interaction. In *The Handbook of Formal Methods in Human-Computer Interaction.* Springer, 183–210.

[10] Alan Dix, Russell Beale, and Andy Wood. 2000. Architectures to make simple visualisations using simple systems. In *Proceedings of the working conference on*

Advanced visual interfaces. 51–60.

[11] Marina Dodigovic. 2007. Artificial Intelligence and Second Language Learning: An Efficient Approach to Error Remediation. *Language Awareness - LANG AWARE* 16 (05 2007), 99–113. https://doi.org/10.2167/la416.0

[12] David Frohlich, Paul Drew, and Andrew Monk. 1994. Management of repair in human-computer interaction. *Human–Computer Interaction* 9, 3-4 (1994), 385–425.

[13] Jakob Nielsen. 1994. Heuristic Evaluation. In *Usability Inspection Methods*, J. Nielsen and RL Mack (Eds.). John Wiley & Sons, New York, Chapter 2.

[14] Donald A Norman. 1988. *The psychology of everyday things.* Basic books.

[15] Emanuele Panizzi and Alba Bisante. 2022. Private or Public Parking Type Classifier on the Driver's Smartphone. *Procedia Computer Science* 198 (2022), 231–236. https://doi.org/10.1016/j.procs.2021.12.233 12th International Conference on Emerging Ubiquitous Systems and Pervasive Networks / 11th International Conference on Current and Future Trends of Information and Communication Technologies in Healthcare.

[16] R. Parasuraman, T.B. Sheridan, and C.D. Wickens. 2000. A model for types and levels of human interaction with automation. *IEEE Transactions on Systems, Man, and Cybernetics - Part A: Systems and Humans* 30, 3 (2000), 286–297. https://doi.org/10.1109/3468.844354

[17] Stephen Russell, Ira S. Moskowitz, and Adrienne Raglin. 2017. *Human Information Interaction, Artificial Intelligence, and Errors.* Springer International Publishing, Cham, 71–101. https://doi.org/10.1007/978-3-319-59719-5_4

[18] SAE International. 2018. Taxonomy and definitions for terms related to driving automation systems for on-road motor vehicles. *SAE international* 4970, 724 (2018), 1–5.

[19] SAE International. 2021. SAE Levels of Driving Automation^{TM} Refined for Clarity and International Audience. SAE Blog, posted: Monday, May 3, 2021, https://www.sae.org/blog/sae-j3016-update.

[20] Ben Shneiderman. 2022. *Human-centered AI.* Oxford University Press.

[21] Ben Shneiderman, Catherine Plaisant, Maxine S Cohen, Steven Jacobs, Niklas Elmqvist, and Nicholas Diakopoulos. 2016. *Designing the user interface: strategies for effective human-computer interaction.* Pearson.