Control confusion: the appliance delay case study

Avi Harel, Ergolight

ergolight@gmail.com

Abstract

The appliance delay case study demonstrates the need for avoiding control confusion. Control confusion is due to enabling similar controls in the wrong scenarios. The article presents a model of scenario-driven user interface (UI) control, which may be used to prevent control confusion. The method highlights the benefits of generic rules for associating the UI control design with the scenarios.

The appliance delay case study

Timer confusion

In response to marketing requests, many systems, such as home appliances (washing machines, ovens, HVAC systems) or software applications, have a "delay start" or "timer" feature, enabling to postpone the device operation for few hours. In systems like computer software, smart home devices, or automation tools, a scheduled task or routine might be causing the delay. Also, if the system relies on a network (e.g., smart home devices, IoT systems), connectivity issues might introduce delays.

In this case study, the UI has two controls, with which the human operator may set the timers. The source of timer confusion is errors in the control selection in a human-system interaction. Typically, the delay feature is rarely used, however it is a source of confusion after the delay feature has been set unintentionally.

The costs of timer confusion

According to the theory of limited attention capacity, the user's attention should be optimized, by priority; the attention allocated to the control tasks should be minimized, to allow maximal allocation to the primary tasks, which are not related to the control tasks. Any distraction from seamless operation, such as timer confusion, involves attention consumption. Therefore, attention-optimized design requires that any distraction from seamless operation is removed.

Timer confusion might generate a negative user experience (UX). Often, the user's reaction to such situations is to call for technical assistance. Often, by the time the technician arrives the delay time is already over, and the system resumes normal operation. Typically, the users do not notice that the delay time is over, and they have no indication about the reason for the operational failure. They feel problems in controlling the device, but they cannot point at the source for these problems.

Investigation

Circumstantial investigation

According to Murphy's Law, failure is often attributed to bad luck. The root cause of failure may be classified according to the trigger: human or technological. Human operators are error prone. A human factors version of Murphy's Law is:

> *If the operators might err, eventually they will.*

Technological triggers are much rarer because they are typically captured during the system verification process.

Reactive root cause analysis (RCA)

Incidents may be prevented based on RCA. Traditionally, people do RCA of high-cost accidents, and disregard low-cost incidents. Traditional RCA is reactive, namely, specific to an incident. In reactive RCA we look for a single main important cause of the accidents.

System operation might fail when the situation does not comply with the controller's needs. In the investigation of this case study in was found that the UI had two similar controls:

- A timer, intended to set the operation time.
- A timer, intended to set the delay until starting the operation

Most of the time, the delay was set to zero, implying that the operation would start immediately. It was the job of the system administrator to reset the delay timer.

In the incident of the case study, it was found that in another operation session a week earlier, somebody set the delay time, instead of the operation time, and nobody noticed the error. The conclusion was that the root cause was occasional confusion between the two timers.

The investigation did not provide answers to key questions: why did the operator set the delay instead of the operation timer? why was the operator not aware of the erroneous setting?

Investigation biases

According to the theory of Black Swans, coincidence is not predictable (Taleb, 2008). According to theories of organizational accidents, organizations prefer to blame the operators instead of admitting the developer's role (Reason, 1997).

Proactive RCA

Traditional operation design relies on the operator's ability to act as intended, and to find and correct possible errors. The engineering approach is that incidents should be attributed to design mistakes, namely, lack of protection from the unexpected. Mishaps should be prevented by

design. The goal of proactive RCA is to hint on ways to disable errors. The proactive version of Murphy's Law is

> *If the system design enables operational failure, the operation will fail regularly.*

And the Proactive version of the Human Factors version of Murphy's Law is

> *If the system design enables user errors, the users will err regularly.*

In this incident the operator was not aware of the delay feature. Accordingly, the questions are rephrased:

- Why did the design enable setting the delay timer erroneously?
- Why did the system not inform the operator about the exceptional setting?

Cross-domain RCA

Often, we may come across incidents in other domains, with characteristics similar to those of our incidents. Sometimes, the incidents of the other domains have already been analyzed, which means that we can adapt conclusions from the investigations in the other domains.

In the context of the Delay case study, we can think of problems due to unintentional setting of the Delay timer in consumer products, such as traditional home appliances or new features integrated in computerized homes. The users of such products may realize that they cannot start the product operation, which makes them conclude that they experience technical failure. A solution to the Delay problem may apply also to the other domains.

Modeling the system coordination

The engineering challenges of UI control design include obtaining and employing models of normal operation and operational failure. In this case study, we have two timer controls: operation timer and delay.

Control confusion

Normal operation is task driven, and subject to proper coordination between the human operator and the system. The operator issues commands or requests to the system, and the system provides reports on its situation and activity.

Selection risks and hazards

Hazards due to selection errors might be unexpected. The sources of selection errors are well understood, and therefore the risks may be regarded as expected. Expected risks are associated with triggers, situations, and activity.

Triggers

The root cause of failure may be classified according to the trigger: human or technological. In the context of timer confusion, we deal with human triggers. Human triggers may be either unintentional, or due to confusion. In this case study the trigger was a lapse by the human administrator.

In traditional interaction design we empower the human operators, enabling them to access rarely used critical features, to cope with unexpected situations. These features are error prone in terms of feature accessibility or availability: when activated unintentionally, such as due to a lapse, the results are unexpected. For example, if a risky feature, such as a delay control, is enabled, then the operation might result in an incident due to a design mistake of under-constrained activity, resulting in the administrator's beta type error.

Situational risks

Situational risks may be classified as either external or internal:

- External risks are about uncontrollable variables, such as environmental conditions and other contextual parameters. Externally, normal operation must be in the performance envelope. External risks are due to approaching the performance boundaries, defined as limits of performance variables.
- Internal risks are about controllable variables, such as service performance and modes, which the operators may set and change. Internally, the situations must comply with the scenario, as perceived in the controller. Internal risks are due to diversion from the situations defined as normal. In this case study, the delay feature should be reset before the session, and the diversion was due to setting it to a non-zero value.

Internal situational risks may be attributed to difficulties in situation coordination, due to situation complexity: the number of possible situations grows exponentially with the number of state machines employed in the system operation.

Scenario-driven coordination

Human-system interaction should be dominated by the human operator. The interaction is regarded as coordinated if the situation complies with the operator's intentions. To enable automated verification of the coordination, the system must employ a means to capture the operator's intention. A generic means to set the operator's intention is by a system variable representing the scenario, set by the operator, as perceived by the operator. Scenarios represent the view of the human operator. A situation may be regarded as consistent if it complies with the scenario.

Scenarios may be used as situation vectors, namely, pointers to the set of states of UI controls, thus, reducing the situational complexity from exponential to linear. Situation consistency may be enforced by a mathematical function, defined by a mapping from scenarios to a vector in the

space of the UI control states. Accordingly, a situation is consistent with a scenario if it is identical to a projection of such mappings. In the Delay example:

- The scenarios are Immediate and Delayed operation
- The situation space consists of the Immediate and Delay timers
- The situational mapping: Intention ➔ (Session time, Delay)

In this case study, we have two consistent situations, defined by the mappings:

- Immediate operation ➔ (Session time, 0)
- Delayed operation ➔ (Session time, Delay time).

Activity risks

Almost all system units, and almost all system features, are prone to activity risks due to mistakes in constraining the system situation (cf. Harel, 2021). Activity risks demonstrated in this case study are incidents. An incident is an instance of operating in exceptional situation, not supported by the operation design. In the case study, the exceptional situation was (operation time, non-zero delay). Incidents are often due to enabling operation in inconsistent situations or in fuzzy scenarios.

Protection strategies

The protection framework

The protection plan proposed here is based on the cybernetics model of feedback control loops, as described in the STAMP paradigm, and developed in the STPA methodology (Leveson, 2004).

To design the coordination between the human operator and the UI controls we may apply the principle of multiple layer defense, as demonstrated using the Swiss Cheese illustration (Reason, 1997). In the context of the Delay feature, we may apply three layers, according to the following approaches:

1. Preventive: scenario-driven mode setting,
2. Proactive: impact validation, and
3. Reactive: compliance verification.

To be on the safe side, we should protect the system from all risky situations and from all risky activities, because we cannot tell when one of them might be critical.

Preventive: scenario-driven mode setting

In normal operation, the situation is scenario driven. The human operator and the UI controls are coordinated by the operational scenario.

The scenario should be implemented as a concrete system entity that the system should handle, ensuring that the situation is well defined.

In the Delay case study, the implementation of the operational scenario is the delay setting.

Proactive: session protocol

The incident could have been prevented had a session protocol managed the timers. According to this protocol, each session should begin by resetting the timers, followed by a prompt to enter the operation time. The protocol may enable exceptionally setting of a delay time. To prevent errors, the protocol should include displaying a warning to the operators about the effect of changing the default delay value.

Reactive: displaying a delay status

Soon after activating the operation, as long as the system updates the delay time, until it reaches zero, the system should display the remaining time to the operators, highlighting the exceptional situation.

Rule-based implementation

Operational reliability relies on rule definition. Many rules are generic, and customizable. Generic rules enable reducing the costs of eliminating operational risks.

Activity tracking

To enable learning from incidents, the system should include means for activity tracking, which is essential for detection of risky situations. Moreover, activity trackers enable to accelerate learning, by capturing near misses, namely, capturing and reporting on exceptional situations before they turn to incidents.

A well-known example of tracking add-ons is the black box used in airplane incident investigation. An example of an add-on designed for Windows applications was the Ergolight tracker, which was part of the usability testing suite, proposed by Harel (1999). This kind of activity trackers may be integrated in the software design or added to an existing design.

Scenario definition

The risk of design based on implicit scenarios is that the system situation is not consistent, in the sense that different system components might assume different scenarios.

To enable the situational coordination, the scenario should be defined explicitly, and implemented as a primary system variable. The scenarios should be defined such that a scenario change involves:

- Enforcing consistent change of the controllable variables

- Verification of the consistency of uncontrollable variables.

A scenario definition may be regarded as optimal if it is not redundant.

Situational rules

The basic rules about the compliance of the situation with the scenario:

- Consistency: The situation should be identical to the vector projected from the scenario
- Protection: While the situation does not comply with the scenario, all risky activities should be disabled.

In the Delay case study, the relevant scenarios were Immediate and Delayed operations. The customized rules for setting the controllable variables should be:

- While in the Immediate operation, the delay should be null.
- While in the Delayed operations, delay should be positive.

To enable the situation verification, the system requirement should specify the operational scenarios, the system design should manage the scenarios, and the software program should verify that the implementation complies with the design.

Situation verification

Situation verification may rely on the skills of the human operators. Generic rules about enforcing the situation awareness are:

- The UI should notify the operators about the compliance
- While the situation does not comply with the scenario, the UI should provide a salient notification about it.

These are generic rules that may be customized for specific systems.

Activity rules

The operational activity consists of events, each associated with a transition between system situations. Therefore, the activity rules are derived from the situational rules.

In this case study, prior to the transition, the active scenario was Immediate operation, and the Delay time was null. Then, after the operator changed the scenario to Delayed operation, the Delay timer was set to a positive number.

Activity verification

The UI should inform the operators about situation changes. The verification rules included are:

- UI feedback on receiving the command for changing their states
- UI notification on completion of the state transition.

These are generic rules, which should be customized for particular systems.

Conclusions

The Delay incident demonstrates the need for early detection of inconsistent situations.

The article presents a rule-based model of UI design, demonstrated with the Delay case study, and a method for eliminating the risks of positive delay based on this model. The method highlights the benefits of generic rules for associating the system situation with the scenarios.

The solution to the Delay problem may apply also to other domains, such as home computing and traditional appliances. The vision proposed here is that standards on integration engineering may include a chapter on when and how to apply the rules for setting and controlling the Delay feature.

References

Harel, A 1999. Automatic Operation Logging and Usability Validation. *Proceedings of HCI International '99*, Munich, Germany

Harel, A 2021. Scenario-based modelling DOI: 10.13140/RG.2.2.12834.35523

Leveson, N 2004. A new accident model for engineering safer systems. *Safety science 42* (4), 237-270, *2004*. 3052

Reason, J 1997. *Managing the Risks of Organizational Accidents*, Ashgate.

Taleb, NN 2008. *The black swan*. Penguin Books.