Intention-driven control: the smart home case study

Avi Harel, Ergolight

ergolight@gmail.com

Abstract

The smart home case study exemplifies design mistakes typical of feature-driven control, such as in popular home systems, and proposes a methodology of intention-driven control, for enforcing seamless operation. The problem of feature-driven control is that it enables use errors. Intention-driven control enforces seamless operation by eliminating use errors. Generic rules, applicable to all interactive systems, are obtained by abstraction of specific rules examined for home control.

The smart home

The system examined is a home control unit enabling selecting a device and setting operational parameters, defining the way the selected device will function. The situations examined are instances of unexpected system behavior. For example, when the user's intention was to turn the lights off, but the embarrassing result of the operation was that the entertainment system unexpectedly turned on.

The design challenge is to eliminate the feasibility of this kind of mishap.

Case study: control confusion

The devices in the case study are a controller and units that the controller should control. The control unit comprises a menu of controlled units, each is associated with an On/Off toggle button and a submenu of interactions used when the unit is on. An interaction may include a trigger and a protocol. The trigger may be a user's selection or a unit message reporting about a significant situational change. The protocol may include feedback messages and interaction rules.

Operational confusion is associated with events of unintentional, unnoticed selection of the wrong device, and with diversion from the protocol.

Methodology

Topics that we need to concern in designing the control of smart homes

- Error prevention, to enable users to focus on their primary tasks
- Error root cause analysis (RCA), to identify design mistakes enabling the errors
- Integration engineering, to disable failure by design rather than detection in testing
- A model of operational risks, for comprehensive protection design

- Intention driven interaction
- Exception handling
- Rule-based implementation

Why focus on error prevention?

Errors consume the user's attention, which is limited, and shared with the primary user tasks. Error prevention is essential to facilitate the user's focus on the primary tasks.

Coping with the unexpected

Common system engineering methodologies and practices do not mitigate the risks of unexpected events, such as user errors or mode errors, typically attributed to 'force majeure'. The underlying premise is that systems engineers can mitigate such operational risks by considering the human limitations in assuring that the system and its operators are coordinated.

The user's tasks

Besides operating the home devices, the user has other concerns and activities not related with the house control.

- The primary user's tasks are not associated with home control. They are related to the user's primary jobs and roles in the community: at work, at home, with friends …
- The control task is secondary.

The main design goal is to optimize the performance of the primary user's tasks. The primary user's tasks are not in the scope of the control design.

The costs of operation errors

The performance of the primary tasks depends on the user's ability to allocate mental resources, namely attention, to the primary task. Errors in the secondary tasks result in hampering the performance of the primary tasks. The challenge is to optimize the allocation of mental resources. Seamless operation means allocating all the attention resources to the primary tasks. Seamless operation means carefree operation, namely, error free operation. The costs of operation errors are due to diverting mental resources, from the primary tasks to the control tasks.

Limited attention capacity

The term attention capacity refers to the brain's finite ability to process information; the capacity of user's attention is limited. According to the theory of mental resources, the human brain has a limited amount of cognitive resources to allocate to different tasks.

In the operation of home control, the primary and the secondary tasks are competing on maximizing the allocation of attention resources. The secondary tasks of controlling the devices compete with the primary tasks. When these resources are stretched too thin, human performance is hampered; eventually, the user might make the wrong decision, and take the wrong action.

The challenge of mental resource allocation

Errors in the secondary tasks result in hampering the performance of the primary tasks. The challenge is to optimize the allocation of mental resources. The challenge is to allocate as much attention to the primary tasks as possible. This implies that the mental resources allocated to the control task should be minimized. To this end, the time spent on error prevention during the operation, and on error correction, should be minimized.

Utility-oriented design

The goals of utility-oriented design are:

1. Maximize the primary user performance
2. Minimize the costs of control errors

Error RCA

Operational failure may be investigated based on RCA. Traditional RCA is reactive, namely, specific to an incident. In reactive RCA we look for a single main important cause of the accidents. Traditionally, these problems are attributed to use errors. Typically, developers do not admit their responsibility to eliminating use errors. Rather, according to theories of sociotechnical behavior, they blame the users instead of admitting their own role (Weinberg, 1971). Often, they quote Murphy's Law, attributing the failure to bad luck.

Data support for RCA

Operational errors are due to operating in exceptional situations. The engineering approach is that usability problems should be attributed to design mistakes and that they should be eliminated by design. However, developers often quote the theory of Black Swans (Taleb, 2008), asserting that instances of use errors are not predictable, because they do not have data about the sources for these errors.

The reason why we do not have data about the sources of use errors is that it is not cost effective. Developers do not install activity trackers, because nobody is going to pay for the expenses. Consequently, the developers do not have the data required to become aware of the errors and to identify their sources.

Traditional integration testing

Traditional integration testing is oops-driven, namely, reactive. To get the evidence of the sources of failure, we need to embed special probes and tools in the system design, to sense, trace, and analyze the system

activity (Harel et al., 2008), and to provide reports about exceptional activity (Harel, 1999, 2009; Kenett et al. 2009). These extra means are costly, because the system activity is complex, and are affordable only in special domains, such as aeronautics and high-risk process industries.

Human centered RCA

System operation might fail when the situation does not comply with the user's needs. According to Murphy's Law, failure is often attributed to bad luck. A human factors version of Murphy's Law is:

> *If the design enables the operators to fail, eventually they will!! (Zonnenshain & Harel, 2009).*

This paradigm emphasizes the system vulnerability to use errors, implying that it should be the developer's responsibility to eliminate the options of use errors.

In the investigation of this case study, it was found that the controller had a menu for device selection, and an On-Off toggle button for changing the On-Off state. Due to a slip, instead of the Lights control, the menu item selected was the Entertainment control. Unaware of the slip, the user pressed the On-Off button to shut the lights off, but since the device that was actually selected was the entertainment control, the action of pressing the On-Off button resulted in changing the state of the entertainment control.

The root cause of such errors is that the users are not aware of the intentions of the developers, and therefore they sometime fail to follow the operational procedure. Often, they do not notice the device selection, and consequently their commands are eventually applied to the wrong device (Harel, 2008).

Operational errors

The "What If" design technique is a powerful tool for generating fresh ideas and challenging assumptions. By asking "What if?" questions, designers and innovators can explore new possibilities, break free from conventional thinking, and develop creative solutions to problems.

Selection errors are often due to device confusion. In this case study, the questions were about:

- Device state confusion
- Device selection error
- Navigating the wrong device

Almost always, operational failure involves operating in an exceptional situation. Often, the cause is attributable to a use error, namely, the user's action that triggered the diversion from normal to the exceptional situation. In common reactive RCA the exceptions are not defined explicitly. They are implicit, and consequently the root cause is ambiguous.

Typically, as in this case study, the operational procedures are not defined explicitly: the developers assume that the users are rational, meaning that they know and recall the operational procedures, which the developers believe that are easy to understand and follow (Harel, 2020).

Understanding operational errors

Human-system interaction during home control is dominated by the users. In traditional interaction design we empower the human operators, enabling them to access rarely used error prone features, in terms of feature accessibility or availability, in order to cope with unexpected situations.

A naïve approach is that system engineers can mitigate operational risks by considering the human limitations. Usually, we assume that the users are rational.

The typical approach to home control relies on the user's ability to act as intended, and to find and correct potential diversions from operational rules. However, rationality relies on the information that the users perceive. However, the information that they receive is not stable and not objective. It is subjective and dynamic (Harel, 2020). Moreover, our knowledge of the operator's mental processing in decision making is subjective, based on our knowledge of which we are aware. This knowledge is different from that of the operators, and it develops gradually as we learn more about the operational requirements.

In utility-oriented engineering we assume that we cannot predict the failure of elementary units. We focus on the interactions, and we assume an OEM model of the elementary units (Harel, 2023). According to these models, we need to specify the functional and performance requirements, and the unit messages about both success and failure.

Proactive RCA

Reactive RCA does not provide answers to key questions: why did the user select the wrong button? Why was the user not aware of the error? Why did the system enable operating the wrong device? The proactive approach is that operational failure should be prevented by design. Because operation failure results from operating in exceptional situations, this implies that the system should be protected from operating in exceptional situations.

The failure demonstrated in this case study may be attributed to shortage of methods and practices for coping with errors by design. The goal of proactive RCA is to hint on ways to disable diversions. In this case study the operators were not aware of operating the wrong device. The RCA questions may be rephrased accordingly:

- Why did the design enable activating the wrong device?
- Why did the system not inform the operator about the activated device?

Rather than looking for a single trigger of the confusion the hazard, we may look for all possible triggers and conditions that enable such confusion, as demonstrated by Harel (2008, 2009). The root cause is human errors, and these may be avoided by automation.

Integration engineering

Traditional system integration

According to the System Engineering Body of Knowledge (SEBoK) “system integration consists of taking delivery of the implemented system elements which compose the system-of-interest (SoI), assembling these implemented elements together, and performing the verification and validation actions (V&V actions) in the course of the assembly”.

The ultimate goal of system integration is to ensure that the individual system elements function properly as a whole and satisfy the design properties or characteristics of the system.

Traditional discipline evolution

The analysis above demonstrates that failure is often due to operating in a situation not supported by the operation design. In many industries, such as consumer products, the evolution is Darwinian, namely, chaotic. Marketing authorities do not emphasize the need for safety and reliability, and developers do not volunteer to protect the operation from operational risks. The proof of the effectiveness is in hindsight, in the system survival in the market.

Learning from SW engineering

Integration design should develop from art to engineering (Harel & Zonnenshain, 2019). A model of this transition was proposed for SW engineering. Traditionally, the actual costs and time to market of software projects were 300% of the plans. Why?

“When a bridge falls down, it is investigated and a report is written on the cause of the failure. This is not so in the computer industry where failures are covered up, ignored, and/or rationalized. As a result, we keep making the same mistakes over and over again” (Standish, 1995).

The conclusion is that rather than focusing on testing, the integration problems should be prevented by design, and accordingly the concept of integration testing should be extended to integration engineering.

Cross-domain learning

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. we can think of problems in exploratory operation of various features in consumer products, such as traditional home appliances or new features integrated in computerized homes. The users of such products may realize that the device is stuck, which makes them conclude that they experience technical failure. This kind of operational problem was analyzed by Harel (2008). A solution to the problem of device confusion may apply also to other domains.

Proactive integration

Theoretically, exceptions could be detected in integration testing. The SEBoK approach to integration is reactive. The focus is on the testing stage. However, this is often too late, way beyond the time to market. It is better if the exceptions are eliminated by design. According to the proactive approach, the integration should begin early at the design, and the validation should start earlier, at the system definition.

HSI engineering

According to early studies, 63% of all software projects overran their estimates, with the top four reasons all related to usability (Lederer and Prassad, 1992).

Typically, the failure is associated with two types of human-system integration (HSI) problems:

- The user fails to set the states of all the device according to their intention
- The user is not aware of the state diversion.

The root cause of this problem is that the system situation does not comply with the user's intention. This implies that the discipline of integration engineering should be extended, to also tackle topics today handled under the title of HSI.

A model of operational risks

The engineering challenges of user interface (UI) control design include obtaining and employing models of normal operation and operational failure.

Hazards due to device confusion might be unexpected. However, the causes of device confusion are well understood; they are associated with triggers, situations, and activity. Therefore, the risks of selection errors may be regarded as expected.

The dual impact of errors

Errors occur when the integration testing is not systematic, namely, when it is not based on a model of the system operation. The impact of error on the operational performance is dual:

- Direct impact: spending the time required to recover from the error
- Indirect impact: slowing down to avoid repeating the error.

The operation model may and should be used for both elimination and detection of exceptions.

Exceptions

Typical exceptions demonstrated in this case study include:

- Error in device selection
  Due to a slip or a lapse, the user does not select the proper device, and consequently, the next action is applied to the wrong device.
- Wrong awareness of the device states
  the user is not aware of the current state or does not verify that it is as expected.

Often, the users are not aware of an exception, and continue operating as normally. For example, as demonstrated in this case study, the user might be unaware of a selection error, and subsequently might browse the wrong device, or might turn off one device instead of turning on an intended device. Not being aware of committing additional errors, they might end up escalating the system to an unpredictable situation.

Triggers

A trigger is an event that changes the operational situation from normal to exceptional. Triggers may be classified according to their actuators: human or technological. Technological triggers are rare because they are typically captured during the system verification process. Most exceptions are triggered by the users.

Human triggers may be either unintentional, or due to confusion. In this case study the critical triggers were device selection errors, namely, actual selection that did not comply with the user's intention.

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 user's intention, as perceived in the controller. Internal risks are due to diversion from the situations defined as normal. In this case study, the diversion was attributed to selection error.

Situational complexity

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.

In the case study the complexity is the multiplication of the ranks of the state sets of all the devices. The risk of setting any of these state sets erroneously is linearly correlated with the complexity, which is very high.

Situational ambiguity

Use errors such as in this case study are due to inconsistent assumptions about the system situation, because the operational rules are not defined explicitly. They are not specified in the requirements documents, and the system does not distinguish between proper and improper situations.

Incidents are often due to enabling operation in exceptional situations or in fuzzy scenarios.

When an operational mode is fuzzy, the user might not keep track of the history of the session activity. For example, the user may intend to browse one of the devices, but then might end up changing the value of the monitor setting, following distraction.

Use case conflict

A special pattern of situational ambiguity is when changing the controller, then resuming the original controller after changing the mode of the same device. This kind of problem is due to multiple use cases, resulting in inconsistent device state.

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 A). Activity risks demonstrated in this case study are incidents. An incident is an instance of operating in exceptional situation, namely, situation not supported by the operation design.

The barrier to using the smart home demonstrated here is mode errors, namely, activity error, due to mode mismatch, when the user's activity assumes a wrong mode.

Escalation risks

Examine a case when the user intends to turn on a device while it is already on. In this case, pressing the On-Off button results in turning the device off. The user might feel guilty for spoiling the device and enter a stressful situation. The user may want to try again, but being in a stressful situation, the user might press the button too hard, resulting in double click. In this case, device Off.

A special escalation risk is when a selection error results in changing the setting parameters of the selected device. Escalation problems were explained by Zonnenshain & Harel (2015).

Integration design

Normal operation is task driven, and subject to proper coordination between the human operator and the system.

Common design strategies

In traditional interaction design we focus on enabling the users to access rarely used error prone features, in terms of feature accessibility or availability. Common design strategies include:

- Feature-driven design: this is the basic approach, employed by engineers who focus on integrating features in the system design. In this case study, the features are those activated by the selection menu, the on-off control, and the pad used for parameter setting.
- Human-centered design: this is a methodology for developing systems that are usable, useful, and delightful to people, by focusing on the user's needs (Norman & Draper, 1986). The goal is to optimize the user's behavior, to match the intentions with the needs and to maximize the user's performance.
- HSI engineering: this is a new approach, focusing on eliminating operational confusion hampering seamless operation, by preventing error-prone activity.

A main barrier to reaching the user's needs is events attributed to user's errors, when the user's activity does not match the user's needs. The user's intention is a virtual variable, mediating the user's activity and the user's needs. In HCD we typically focus on ensuring that the user's intention complies with the user's needs. In HSI engineering we focus on ensuring that the user's activity complies with the intentions.

Minimizing the human control

Because the user is error prone, we need to minimize its part in the situation control, to reduce it to the bare necessity. The principle of minimal human involvement leads to the principle of direct mapping from intention to action, such that the impact of the action is done by automation (Harel, 2021 B).

Control automation

We may prevent human errors by automation, by mapping intentions to situational vectors, formulated as intention-driven situational rules. These mappings define normal situations. All other situations should be regarded as exceptional. The operator issues commands or requests to the system, and the system provides reports on its situation and activity.

Control automation is the application of technology to operate and manage processes, systems, or equipment automatically. It involves using sensors, actuators, controllers, and software to monitor and control various parameters to achieve desired outcomes. Core Components of Control Automation:

- Sensors: Gather data about the system's state.
- Actuators: Implement control actions based on sensor data.
- Controllers: Process sensor data and generate control signals.
- Software: Provides intelligence and algorithms for control.

The trigger of most failures of home control is errors in device selection. In an intention-driven design, the selection may be automated. In this case study, the user intentions may be turning the lights on or off, turning the entertainment system on or off, and more. The selection button is replaced by a software switch. The control unit should be provided with means for direct selection of these actions and activating them. These solutions have been demonstrated in the context of radiotherapy (Harel, 2024).

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 user and the home control we may apply the principle of multiple layer defense, as demonstrated using the Swiss Cheese illustration (Reason, 1997). In the context of home control, we may apply three layers, according to the following approaches:

1. Method: scenario-driven mode setting,
2. Implementation: automation
3. 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.

Behavior control

The goal of behavior control is to prevent errors. As the model of operational risks suggests, this implies that we have two primary design challenges:

1. Prevent diversion to exceptional situations
2. Facilitate detection and recovery from exceptions.

To enable exception handling the system should track the activity during the operation and identify exceptions.

Information design

The user and the underlying system interact by exchanging information between them:

- Notification: information that the system presents to the user about its situation
- Alarms: information that the system presents to the user about situation changes
- Command: a message from the user to the system, requesting an action
- Preview request: a message from the user asking for information required for decision making
- Command preview: the system response to a preview request
- Feedback: a message from the system in response to a command request

Human-centered design (HCD) is about the content in these information elements. Specifically, in order to enforce desired human behavior, the machine should provide information about the risks, and the information should be presented in forms considering the theory of human perception.

Activity tracking

Basic activity tracking comprises a means to record the activity during the operation, to use in case of a need to investigate a failure. A well-known example of tracking add-ons is the black box used in airplane accident investigation.

An advanced version of activity tracking is by using an add-on, which enables to accelerate learning, by capturing near misses, namely, capturing and reporting on exceptional situations before they turn to accidents. 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 tracker add-on may be integrated in the software design or added to an existing design.

Exception control

To detect exceptions, the design should be based on a definition of normal behavior. To facilitate the definition, we define normal behavior by protocols of normal activity. The challenge is that activity protocols could be complicated by the complexity of possible situations. This complexity may be resolved by defining scenarios as mediators between the intentions and the situations the comply with the intentions.

The scenario variable

Systems are designed to operate according to scenarios. This means that the response of any unit to any event is designed based on an assumption about the operating scenario. In normal operation, all system states assume the same scenario. This scenario is called the system context.

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. Human-system interaction may be 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 should represent both typical and exceptional intentions.

Scenario definition

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.

Device operational modes

Device operational modes are the system view of the scenario. Standard modes operational modes applicable to many systems are Operational, Testing, Training, Maintenance, Troubleshooting, and Safe-mode operation. Standard modes applicable to devices operated by home control are active, idle, and unavailable.

Mode errors

Mode errors are special, common sources of activity errors when the user is not aware of a change in the operational mode. They occur when a user misinterprets the current device mode, leading to actions that are appropriate for an incorrect mode. Examples:

- A critical feature is not available, because it was disabled earlier for maintenance
- Unnoticed mode set by default, when recovering from intermittent power failure
- Use cases competing on setting the mode of the same device.

An example, as demonstrated in this case study, is when a user is trying to control one device but is approaching a different device instead. The design challenge is to eliminate mode dependent activity.

Scenario-driven operation

Scenarios may be used as situation vectors, namely, pointers to the set of states of the system units, thus, reducing the situational complexity from exponential to linear. Situational consistency may be enforced by a mathematical function, defined by a mapping from scenarios to a vector in the space of the states of the system units. Accordingly, a situation is consistent with a scenario if it is identical to a projection of such mappings. In the home control example:

- The scenarios are lights on, lights off, entertainment on, entertainment off, etc.
- The situation space consists of the states of the home devices
- The situational mapping: Scenario ➔ (lights situation, entertainment situation, etc) where each device situation consists of the device states.

In this example, we have four consistent situations applicable to the case study, defined by the mappings:

- Scenario: Lights On ➔ (Lights: on)
- Scenario: Lights Off ➔ (Lights: off)
- Scenario: Entertainment On ➔ (Lights: on)
- Scenario: Entertainment Off ➔ (Lights: off)

Challenges of mode design

Norman (1981) highlighted the importance of clear and consistent feedback to prevent mode errors. Designers should:

- Indicate the current mode to the user.
- Ensure that controls for different modes are easily distinguishable.
- Require explicit user action to change modes.
- Simplify the system's interface whenever possible.

Norman argues that by implementing these design principles, developers can create systems that are more intuitive and less prone to user errors.

This approach is limited by the Human Factors version of Murphy's Law. It is difficult for the users to follow the mode changes, and they often fail. The result of mode errors is often an unexpected situation, which the users cannot handle easily.

Essentials of mode design

To prevent operating in the wrong mode, the mode transitions should be associated with scenario changes that represent changes in the user's intention. Design rules for preventing operation in the wrong mode:

- Avoid conflicting mode changes: if a feature is used by two or more use cases, then the feature operational mode should be set locally, subject to the use case that activates and manipulates the mode.
- Early mode validation in case of scenario change: avoid operating in the wrong mode by checking the mode compliance at the time of the scenario change.
- Enable control activation and mode changes only when they comply with the scenario.
- Notify the users on the modes and provide feedback when they try to activate a control when it does not comply with the scenario.

They occur when a user misinterprets the current device mode, leading to actions that are appropriate for an incorrect mode.

Situational rules

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

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

These are universal rules, which should be customized for particular systems. Other generic situational rules target specific scenarios:

- When in exceptional situation, notify and enable initiating a troubleshooting session
- Special notification when a critical feature, such as backup, is disabled
- Special notifications when in testing, maintenance, or training operating

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.

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. The generic rules are about informing the operators about situation changes:

- Feedback on the success and failure of scenario changes
- Warning on unexpected situation change.

These are universal rules, which should be customized for particular systems. Other generic activity rules target specific activity:

- On diversion from normal to exceptional situation, follow the diversion protocol below
- On disabling a critical feature, such as backup, alert the user about the risks
- On transition from functional to maintenance, or training, alert and disable risky features
- On transition from testing to functional, alert and disable test-only features

Diversion protection and rescue protocols

In case of diversion from normal situation, the system should assist the users with the recovery. The assistance may be based on generic protocols, comprising:

- Warning the user before execution of a risk action about the potential risks
- Control the user's acknowledgement of the warning message
- Supporting the users' decision about taking the risks
- Triggering protective procedures at the controlled devices
- Receiving and processing feedback messages sent from the devices
- Alert the users about risk reports
- Control the user's acknowledgement of the alert
- On-going verification that the warning system is effective, and the user can be alerted
- Proposing means to the users about possible ways to troubleshoot and rescue from the hazard.

Compliance verification

The system should continuously verify that the situation complies with the scenario, as set by the automation. In case of change, the system operation should freeze, to enable troubleshooting and the system should inform the user about the problem, suggesting resetting, or calling for technical assistance.

Conclusions

Home control incidents demonstrate the need for early detection of inconsistent situations. The article presents a rule-based model of the system operation, and a method for eliminating the risks of control errors. The method highlights the benefits of generic rules for associating the system situation with the scenarios.

The solution to the home control problem may apply also to the 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 home controls used for home. These rules may be specified in the unified architecture framework (UAF, Martin & O'Neil, 2021).

The vision

Home control may be implemented by customized models based on generic rules such as those discussed in this article.

Interaction designers should pay attention to the technical, operational, functional, environmental, cognitive, and blackout factors, looking for ways to improve the control usability to enable seamless operation.

The quality of operation relies on rule definition. The generic rules developed in this study may be customized and applied to various kinds of interactive systems, in various domains, thus enabling reducing the costs of eliminating operational risks.

References

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

Harel, A 2008. Standards for Defending Systems against Interaction Faults, *Incose International Symposium, Utrecht, The Netherlands*. DOI: 10.1002/j.2334-5837.2008.tb00908.x

Harel, A 2009- Statistical Analysis of the User Experience, Invited talk in *2nd Meeting of isENBIS*, Hertzelia, Israel

Harel, A 2020. System Thinking Begins with Human Factors: Challenges for the 4th Industrial Revolution. in R.S. Kenett, R.S. Swarz and A. Zonnenshain (Eds), Systems Engineering in the Fourth Industrial Revolution: Big Data, Novel Technologies, and Modern Systems Engineering, Wiley, DOI: 10.1002/9781119513957.ch15

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

Harel, A 2021 B. Model-based Human Interaction Design. DOI: 10.13140/RG.2.2.22631.16807

Harel, A 2023. Essentials of Integration Engineering, Preprint, DOI:10.13140/RG.2.2.19607.55201

Harel, A 2024. Synchronization control: the Therac 25 case study, DOI: 10.13140/RG.2.2.34099.64803

Harel, A & Weiss, M 2011. Mitigating the Risks of Unexpected Events by Systems Engineering (A. Harel, M. Weiss), The Sixth Conference of INCOSE-IL, Hertzelia, Israel.

Harel, A & Zonnenshain, A 2019. Towards HSI Engineering. *The Voice of the Systems: The Journal of the Israeli Systems Engineers, June issue*. DOI: 10.13140/RG.2.2.30285.77284

Kenett R, Harel A and Ruggeri F 2009- Controlling the usability of Web Services, *International Journal of Software Engineering and Knowledge Engineering (IJSEKE), Volume: 19*, Issue: 5(2009) pp. 627-651

Lederer, A.L. and Prasad, J., 1992. Nine management guidelines for better cost estimating. Communications of the ACM 35(2) 51-59.

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

Martin JN & O'Neil DP, 2021. Enterprise architecture guide for the unified architecture framework (UAF), *INCOSE International Symposium*, Wiley Online Library.

Norman, DA 1981. Categorization of action slips. *Psychological Review, 88*(1), 1–15. https://doi.org/10.1037/0033-295X.88.1.1

Standish Group, 1995, The Kompass Report, Forbes.

Norman, DA & Draper, SW 1986. User Centered System Design: New Perspectives on Human-Computer Interaction", in *D. A. Norman and S. W. Draper* (ed.), Hillsdale, New Jersey: Lawrence Erlbaum Associates.

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

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

Weinberg, G 1971. *The Psychology of Computer Programming*. Dorset House Books.

Zonnenshain A & Harel A 2009. Task-oriented System Engineering. *INCOSE Annual International Symposium, Singapore*. DOI: 10.1002/j.2334-5837.2009.tb00982.x

Zonnenshain A & Harel A 2015. "A practical guide to assuring the system resilience to operational errors". *INCOSE Annual International Symposium, Seattle*. DOI: 10.1002/j.2334-5837.2015.00080.x