Towards Safety-oriented System Design: Preventing Operator Errors by Scenario-based Models

Avi Harel

***Abstract*—** Most accidents are commonly attributed in hindsight to human errors, yet most methodologies for safety focus on technical issues. According to the Black Swan theory, this paradox is due to insufficient data about the ways systems fail. The article presents a study of the sources of errors, and proposes a methodology for utility-oriented design, comprising methods for coping with each of the sources identified. Accident analysis indicates that errors typically result from difficulties of operating in exceptional conditions. Therefore, following STAMP, the focus should be on preventing exceptions. Exception analysis indicates that typically they involve improper account of the operational scenario, due to deficiencies in the system integration. The methodology proposes a model, which is a formal definition of the system operation, as well as principles and guidelines for safety-oriented system integration. The article calls to develop and integrate tools for recording and analysis of the system activity during the operation, required to implement and validate the model.

***Keywords*—** accidents, complexity, errors, exceptions, interaction, modeling, resilience, risks

I. Human Errors

Studies about the sources of accidents indicate that most of them are typically attributed to human errors, or improper usage. The factors mentioned by the reviewer are in the category of human errors. Human errors explain most accidents in the air (60%, PlaneCrashInfo 2014)[38] sea (80%, Baker & Seah 2004)[2], driving (90%, Singh 2015)[44], and in the industry (60-80%, Kariuki & Löwe 2006)[33].

Errors are incidental. Weinberg (1971)[50] reported on typical subconscious design mistakes, due to egocentric programming, hampering the productivity of the computer users. Shneiderman (1980)[42] promoted the concept of empathic programming suggested by Weinberg, and proposed few principles for avoiding such design mistakes. Norman (1983)[36] classified activity errors due to omission, or to taking the wrong action. A wrong action may be either a slip or a mistake. A mistake may be in situation perception or in deciding which action to take. However, following Bainbridge observation about ironies of automation (1983)[1], Weiler & Harel (2011)[49] argue that judgment errors under stress are due to relying on irrelevant prior experience.

The meaning of the term "human error" or “improper usage” is ambiguous. Accident analyses indicate the most of these instances involve several factors, most notable are component malfunctions. Often, the error is attributed in hindsight to the person who happened to be nearby, typically, the operator who was on duty (Dekker, 2007)[11]. Most accidents may be attributed to human limitations to perform perfectly in extreme conditions, such as exceptional situations due to design mistakes and bugs.

Following Hollnagel (1983)[28] the model presented here assumes that the term “error” is an engineering bias, diverting the accountability for design mistakes, resulting in failure to assist in the collaboration with the operators. Harel, (2010)[20] suggested that “in attributing the incident to the trigger, instead of the situation, the system stakeholders typically become sloppy and careless about the design features that could have prevented the incident”, as demonstrated in the following figure:

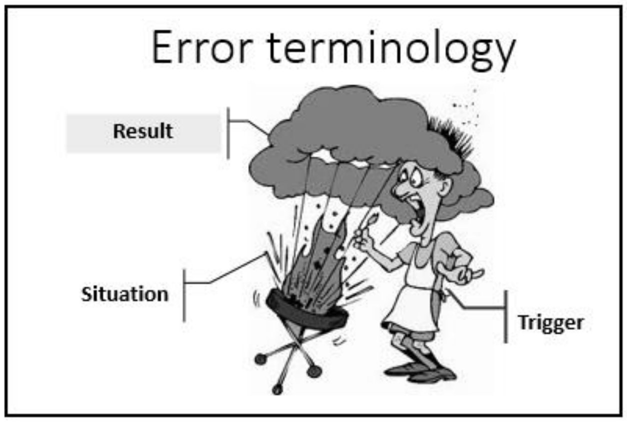

Fig. 1 Attributes of errors

According to Zonnenshain & Harel (2015)[56] the term refers to activities of the responsible organization intended to divert the focus of investigations from the management to the operators. For example, Harel (2011)[21] analyzed various ways in which vendors of equipment for medical alarms infect the standards by diverting the accountability for failure to the operators. The implication of this observation is that rather then investing on error analysis, the design should focus on preventing failure.

Barriers to seamless operation include instances of confusion and hesitation of the operators, due to anxiety about potential loss. Often, the confusion is attributed in hindsight to operator’s errors. Typically, we expect the operators to be rational. However, as prior studies demonstrated the meaning of the term rationality is vague (Harel, 2020)[22]. Rationality relies on the information that the operators perceive. However, the information that they receive is not stable and not objective. It is subjective and dynamic.

A. Harel is with Ergolight, Haifa 3433506. Israel (ergolight@gmail.com).

A. Human performance

A common measure of the system value is in terms of performance. Typically, the meaning of this term depends on the purpose and functions of the system. Ideally, it is associated with metrics such as throughput, bandwidth, power consumption, etc. However, the perceived performance is typically industry and domain specific. In practice, it depends also on implicit factors, which are not tangible or testable. Often, the implicit factors are more significant than the measureable and testable factors. Typically, the term refers to the perceived efficiency, namely, how well the system performs.

The system view of performance is based on the wrong assumption that the operators may do their job perfectly. The human factors view of performance focuses on bottleneck due to the limitations of the human operators, such as attention deficit, stress, when the attention demands are high, such as in uncertainty, or in multi-tasking (e.g. Wickens, 1992)[52].

B. Limited attention capacity

The human attention capacity is limited. When under stress the operators are liable to err, even when they pay their full attention to the operation (Clark and Dukas, 2003)[8]. For example, under stress, the operators may focus on solving a problem suggested by a particular alarm, and miss indications about other critical problems.

C. Situation awareness

This concept is about the operator's failure to perceive the system and environmental elements as expected, or to comprehend the significance of the situation perception. Situation awareness is critical for successful decision-making across a broad range of systems (Endsley, 1995)[13]. For example, Harel (2006)[18] explained that operational reliability and quality are critical for enforcing proper reaction to the alarms.

D. Operating in exceptional situations

A key hurdle to maximizing the system utility is the difficulties that the operators experience when the system is in exceptional situations (Zonnenshain & Harel, 2015)[56]. The reason for this is that regular training targets normal conditions. During normal operation, the operators encounter exceptional situations only occasionally, which is not sufficient for effective learning. Whenever they encounter an exceptional situation, they waste too much time trying to find their way around it. For example, informal studies on the productivity in text editing indicate that about half of the time is wasted in recovery from errors.

The means to avoid exceptional situations and to support exception management may be integrated into the model used in the system design.

E. Accountability

The article advocates the HF version of Murphy's Law: if the design enables the operators to fail, eventually they will. In particular, improper usage such as failure to handle situations with which the operators are not familiar, should be attributed to design mistakes. Therefore, the article advocates a design goal of protecting the system from human errors. According to the proactive version of Murphy's Law, it is the design's responsibility to prevent situations in which the operators might fail (Harel, 2011)[21].

F. Operational reliability

Operational reliability is the system's capability to minimize the costs of operating in exceptional conditions. Operational reliability may be defined as "The ability of an apparatus, machine, or system to consistently perform its intended or required function or mission, on demand and without degradation or failure" (Berard, 2013)[5].

G. The design challenge

The article assumes the proactive version of Murphy's Law, attributing operational problems to design defects, of enabling the operational problems. The article assumes a variant of Taleb's Black Swan theory (2007)[47], illustrated in the following figure:

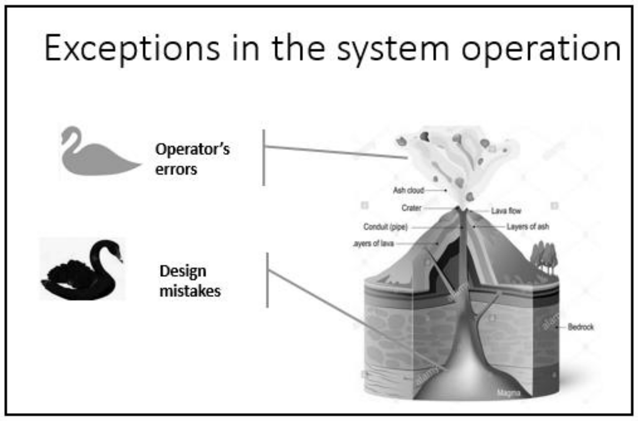

Fig. 2 Operator's errors are due to design mistakes

The figure illustrates that failure attributed to the operators should rather be attributed to design mistakes. Most design mistakes are latent, waiting for the opportunity to emerge. Only those with costly results are observed, and consequently attributed to operator's errors.

II. Design Challenges

A. Usability

Believing that it is the designer's responsibility to reduce the costs of operation, Norman and Draper (1986)[37] explained that to avoid the loss, the system design should be user-centered. Shneiderman (1986)[43] proposed eight golden rules for user interface design, based on Human-Centered Design (UCD) principles of usability assurance. The quality of the system usability affects the operator's productivity, system safety, and the experience of using consumer products.

B. System Integration

Prior studies indicate that HCD may prevent some of the errors, but not those due to flaws in the system integration. Indeed, many accidents are due to the operator's inability to detect, recognize, or identify situations in which not all units assume the same operational conditions. Examples of such accidents are the Therac 25, Torrey Canyon, TMI, Bhopal, and may friendly fire accidents. In hindsight, investigators attribute the coordination problem to operator's errors, assuming that the operators could have manage the exceptional situation. In reality, as Bainbridge (1983)[1] observed, operators are likely to fail in the task of coping with rare situations. Therefore, a key design challenge is to ensure coordination by design.

C. Human-System Integration

In many accidents, the coordination problem was between the operator and the technical system. The UCD view of these incidents is of the operator's situation awareness, attributing the failure to the human operators. Human-System Integration (HSI) is a special sub discipline of system integration, attributing coordination failure to the system design, rather than the operators. Accordingly, HSI engineering descends from systems engineering.

D. Operational complexity

A primary hurdle to operational reliability is operational complexity. Operational complexity is about possible confusion, and it applies also to very simple systems. Two common error modes attributed to operational complexity are physical confusion, such as control confusion, and logical confusion, such as mode confusion.

Control confusion is an instance of applying a wrong control due to similarity or proximity, as illustrated in the following figure:

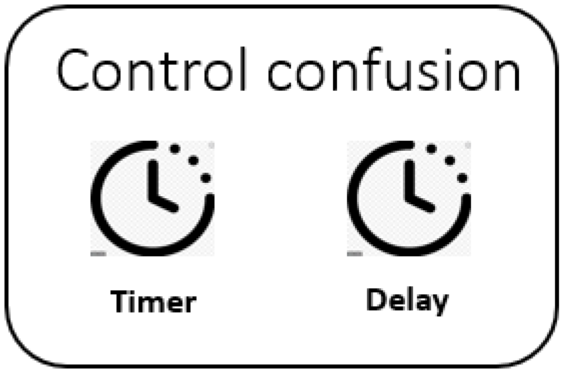

Fig. 3 Confusion delay timers

This type of complxity applies to many consumer systems, such as home appliances: Laundry, drier, air conditioner, furnace, and oven. It also applied the many B-17 accidents in WW II. Control confusion may be resolved within the discipline of HCD. Often, control confusion may be resolved by redundancy analysis, according to the principle of Occam's Razor.

Mode confusion is an instance of activating a control in the wrong mode, resulting in an unintentional effect. Examples of critical mode confusion are of activating setup or maintenance features during functional operation.

The number of situations grows exponentially with the number of states. Most of them are exceptional. Following Weaver (1948)[48] complexity may be defined as "the degree of difficulty in predicting the properties of a system if the properties of the system's parts are given". Sheard and Mostashari (2009)[40] categorized complexity as either structural, dynamic, or socio-political.

Many incidences of operational difficulties are due to inconsistent system response to the operator's commands. Accordingly, we may define operational complexity in terms of the amount and variety of condition-dependent activities. Operational complexity may be defined in terms of conditional activity, such as the conditions for human-machine interaction or inter-unit coordination, namely, the consistency of the reaction to events. If the design enables various reactions to a specific event, depending on the operational scenario, then this event is error-prone, contributing to the complexity. Reducing the operational complexity is critical for maximizing the HSI utility.

The article proposes to prevent unintentional mode setting by impeding the transition, as illustrated in the following figure:

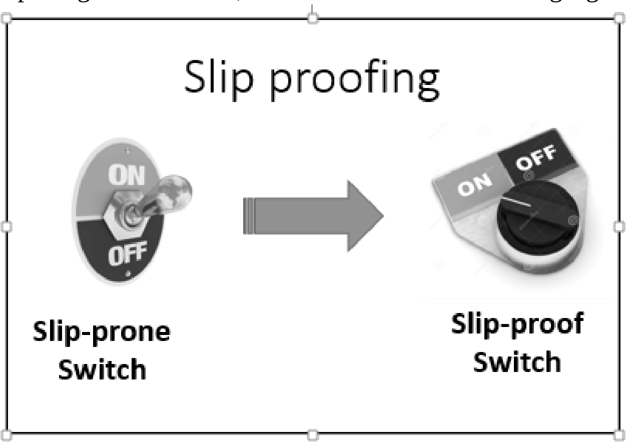

Fig. 4 Preventing slips

The article proposes to resolve this kind of problems by scenario-based design and testing.

E. The performance envelope

Hollnagel (2006)[29] suggested that system failure is often associated with operating in extreme conditions. The limits of performance may be defined by the performance envelope. The performance envelope is an extension of the concept of flight protection envelope. For example, the speed of an airplane is limited by the stall threat and the Mach number, and the altitude is limited by the Coffin Corner (Swatton, 2011)[46]. These conditions should be considered setting the performance goal. The performance envelope may be optimized by design supporting seamless operation.

F. Operational constraints

According to the principles of cybernetics, to avoid failure, the system should control its behavior, similarly to animals (Wiener, 1948)[53]. This principle is key to endorsing HSI reliability. HSI reliability relies on operating according to rules. In 1972 Alain Colmerauer and Philippe Roussel developed Prolog, a rule-based computer language (Cohen, 2001)[9]. Shapiro (1983)[39] studies the using Prolog for algorithmic program debugging. Leveson (2004)[34] adopted the principles of cybernetics and proposed the STAMP paradigm, applying the principle of self-control in a hierarchy of system views. The Prolog language demonstrates the feasibility of the STAMP paradigm. Operational constraints are operational rules constraining the system operational (Harel & Zonneshain, 2019)[25]. Typically, these constraints are scenario-dependent.

G. *Operational exceptions*

An exception is a situation intruding the performance envelope. HSI exception extends the concept of software exceptions, introduced in LISP (Gabriel & Steele, 2008)[15]. The extension is in the structures of static, dynamic, or behavioral exceptions. The original software exception has two components: a probe in the program, and an exception handler. The probe is actuated when the program reaches this probe. In contrast, operational exceptions reside in the system situations and events. The exceptional situations are handled by scanning the situational constraints, and the exceptional events are handled at the event handling. Applying system thinking (Leveson, 2004)[34], HSI focusses on rare situations, and the HSI models focus on operational rules (Harel & Zonnenshain, 2019)[25].

H. *Operational hazard control*

A hazard is a potential source of loss. Hazard control is used in industry to mitigate the risks of hazards. Operational hazard control is a method of hazard control focusing on HSI. It is inspired by methods of Statistical Process Control (SPC, Wheeler & Chambers, 1992)[51] and of Statistical Quality Control (SQC, Shewhart, 1931)[41]. Operational hazard control eliminates the risks of exceptional events and of operating in exceptional situations.

I. *Operational resilience*

According to the INCOSE Resilient Systems Working Group (RSWG), resilience is the ability to maintain capability in the face of adversity. Jackson and Ferris (2013)[31] presented principles for assessing and improving the resilience of engineered systems across their life cycle. Operational resilience is about HSI factors in resilience assurance (Zonnenshain & Harel, 2015)[56].

For example, we may explore various collaboration options in a minimal system, consisting of a simple engine with two states: On and Off, operated by a switch with states: On and Off. The functional option is complicated when the operator is required to support early detection and identification of malfunction. How will the operators know about instances of malfunction? How will they know if the problem is with the engine or with the switch? How will they identify problems in the connections? How will they know when the engine starts too slowly?

An error-proof design may include sensors of the engine and switch states, and an indication when the state are not compatible with each other. In addition, the design may include indication of these states, to facilitate the troubleshooting. The sensors may also be used to notify on problems of starting or stopping the engine too fast or too slowly. The following figure illustrates the inter-state transitions as system variables.

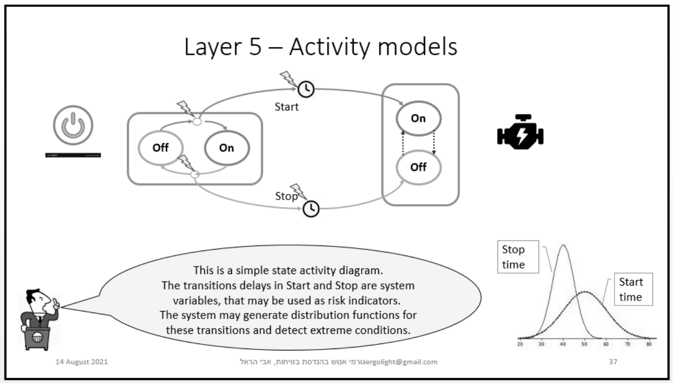
Layer 5 – Activity models
Start
Off
On
On
Off
Stop
This is a simple state activity diagram. The transitions delays in Start and Stop are system variables, that may be used as risk indicators. The system may generate distribution functions for these transitions and detect extreme conditions.
Stop time
Start time
14 August 2021
ergolight@gmail.com
37

Fig. 5 Inter-unit state transitions

The system may record the transition delays and generate distribution functions for these delays. The system design may make use of the distribution parameters, and include identification of extreme values, as well as extra means for alarming and emergency shut-down. The following figure illustrates how the design may define exceptional delays, and how the system may respond to exceptions:

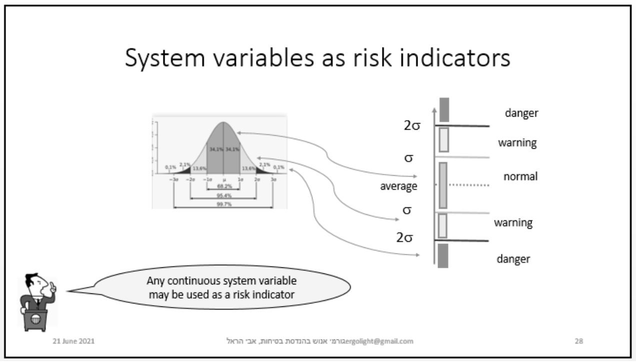
System variables as risk indicators
2σ
σ
average
σ
2σ
danger
warning
normal
warning
danger
Any continuous system variable may be used as a risk indicator
21 June 2021
ergolight@gmail.com
28

Fig. 6 Discretizing the system variables

In the example, the threshold of sigma may be used for alarming, and the threshold of two sigma may be used for safe-mode operation, such as emergency shut down.

III. Modeling

Scientific findings are documented in models, obtained in frameworks of meta-models of information behavior (Wilson, 1999)[54]. For example, Following Fuhs (2008)[14] description of hybrid vehicles, Boy (2012)[6] suggested modeling the system operation in the form of orchestrating human-centered design.

Models enable participation by diverse SMEs. Model-based engineering enables agile development of complicated systems. For example, Harel (1999)[17] demonstrated a model-based approach to usability testing, by capturing and analysis of instances of difficulties in using Windows applications. Also, Harel et al. (2008)[24] demonstrated a model-based method for automated analysis of website navigation based on usage statistics. Through simulation, models provide a gradual, seamless, reliable, modular transition from requirements to the implementation of digital twins and the final system.

A. *Model-based system integration (MBSI)*

The methodology of model-based engineering is inspired by a similar methodology of rapid prototyping, developed in the framework of software engineering in the 70s (Grimm, 1998)[16].

MBSI is the modern systems engineering version of software prototyping, a concept explored in the 80s Model-based system integration (MBSI) enables early integration by simulation, resulting in shortening the integration phase and reducing the development costs. (Luqi, 1989)[35]. MBSI may consist of project-specific, functional features, as well as universal features applicable to maintenance, resilience, training, etc. The universal features may apply to various domains and industries.

B. *The operator's view*

Jacobson (1987)[32] described a technique used at Ericson to capture and specify system requirements based on use cases. Today this technique is part of the Universal Meta Language (UML), commonly used in software design. The concept of use cases was migrated to systems engineering, in the framework of SysML. They are key to describing the designer's view of the system behavior, required to support Model-based Systems Engineering (MBSE). The operator's view of the use cases is called usage scenarios (Spool, 2014)[45]. HSI Meta Language (HSIML) is a meta-language used for the HSI design.

C. *Model-based HSI*

Failure is often attributed to latent defects, wear-out, unexpected environmental conditions, and improper usage (both accidental and malicious). Many developers are not aware of the risks of operating in exceptional situations. Therefore, they do not gain the education and resources required to mitigate these risks.

Hollnagel (2006)[29] proposed two ways for modeling the system operation. The proactive approach is about how to describe normal behavior, and the reactive approach is about how to describe extreme events. HSI modeling focuses on universal methodologies of rule-based design, for the sake of reducing the operational complexity. According to the proactive approach to failure, the system design should support the operation also in extreme conditions. HSI modeling is a hybrid approach, in which we define normal behavior proactively, and we apply learning from failure. The orchestra illustration is proactive-oriented. The reactive part is by serendipitous learning from incidences (e.g. Copeland, 2020)[10].

Model-based HSI (MBHSI) is part of model-based engineering, focusing on the integration between the system and its operators. Model-based design enables seamless adaptation to design changes. Rule-based models enforce mitigating the risk of operational complexity. Model-based HSI facilitates the implementation of the HSI part of the digital twin (Barricelli et al., 2019)[3].

D. *Learning from incidences*

The method applied in this study is actually a variant of the SSM methodology, which is critical system thinking, in generating patterns of system resilience (Checkland, 2001)[7], which is in the domain of Concept-Knowledge (C-K) theory employed in the design of social systems (Hatchuel et al., 2011)[27]. The goal is to identify patterns of failure, and to assign method employed in various industries. The methodology for pattern generation is based on abstraction of the system elements and activity, and matching abstracted elements of various incidences. This is illustrated in the following figure:

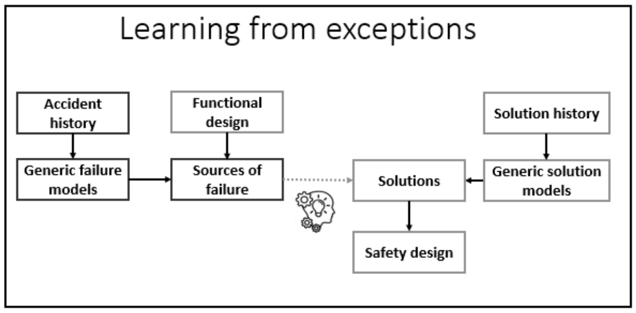

Fig. 7 Elements of modeling the system operation

Incidence modeling may be based on four types of evidence: anecdotal, statistical, causal, and expert evidence (Hornikx, 2018)[30]. By its nature, anecdotal evidence is subject to systematic deviation from the norm and/or rationality in judgment (Haselton et al. 2005)[26]. Expert investigation is also biased: Drury et al. (2002)[12] found that during the investigation stage the number of facts considered grows, but then decreases at the reporting stage. They conclude that the incident reports may not consider all causal factors. In HSI reliability studies, these biases need special attention.

Modeling may be based on the gradual abstraction of incidences and solutions. An incidence is an instance of crossing the limits of the performance envelope. The reactive part in HSI modeling is by cross-domain learning from incidences.

We may distinguish between two types of incidences: those due to rare events and those due to daily, low-cost events. Taleb (2007)[47] argued that it is impossible to predict incidences due to Black Swans (rare events) because a-priori we do not have the data required for the prediction.

The first stage in the methodology generation is to create a bank of generic failure modes, based on failure analysis. The following figure illustrates a way to document failure analysis when applied to the TMI accident:

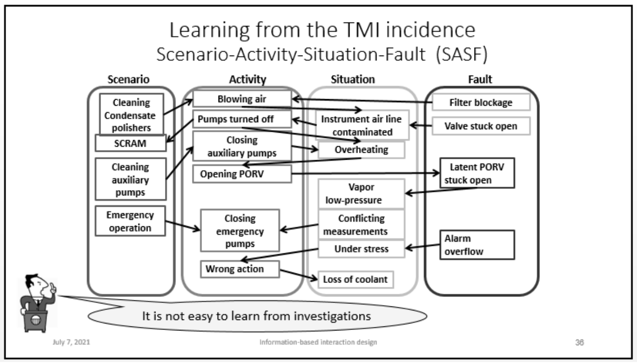

Fig. 8 Event flow in accident investigation

The second stage is abstraction. We accumulated evidence from other systems with similar problems. For example, there are several types of appliances that share the same redundant Delay feature

The methodology was developed gradually. An initial set of

11 patterns was proposed in 2008 in a working group on risk management of the Israeli chapter of INCOSE. After going through a bunch of failure modes, proposed by the workshop participants, we may come up with a simple model of system failure such as the one depicted in the following figure:

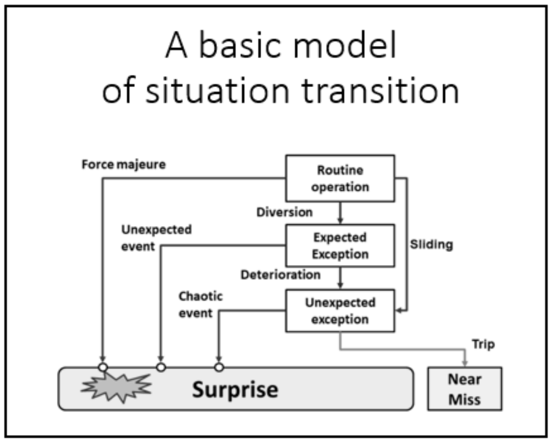

Fig. 9 Basic situation classification

Then we established a dedicated working group on system resilience, in which we examined various possibility of failure modes found in 67 incidences. The outcome of this examination was a comprehensive model of system resilience. The model was reported in 2015 INCOSE conference (Zonnenshain & Harel, 2015)[56]. An updated version of this model is demonstrated in the following figure.

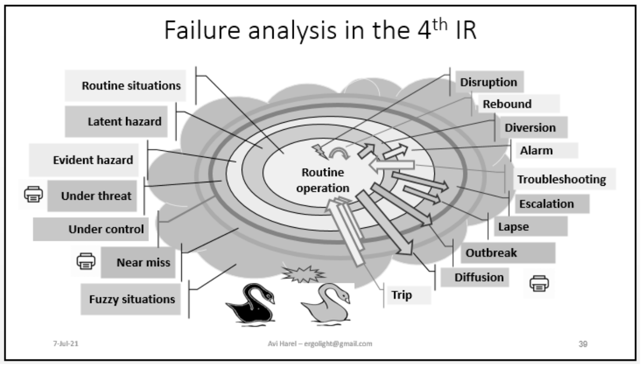

Fig. 10 A model of exception management

Based on this model, we examined a pattern of failure modes of activating a maintenance-only feature in functional operation, as described above. We looked at common methods for protecting from incidences employed in the industry and we came up with two approaches: a. disabling the activation of maintenance-only features during functional operation, and b. warning about such instances. This defines two patterns of preventing such mishaps, defined as operational rules. BTW, these rules apply also to simple systems, such as home appliances.

The next stage is inter-disciplinary task allocation. It is the responsibility of the systems engineer, and/or the safety engineer, to select the proper patterns, and it is the responsibility of the HCD practitioner to design the warning messages, the rebound feedback, and the notifications to the operators.

Finally, we had a validation test of the patterns. Matched the patterns with each of the incidences in our sample, and we obtained a pie chart representing the power of the test, as follows:

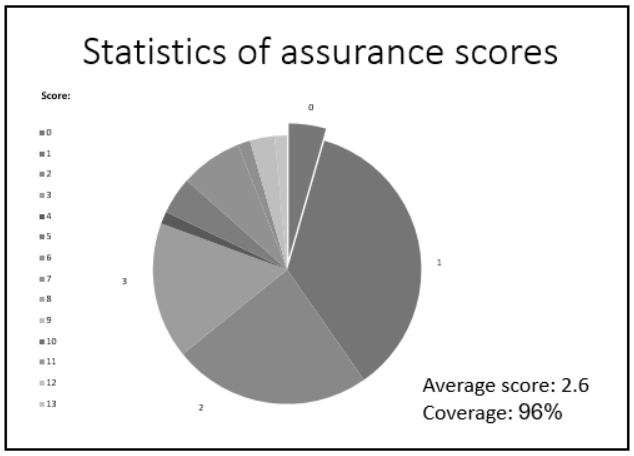

Fig. 11 Distribution of the effects of the 2015 model

For 96% of the incidences we matched at least one pattern from our collection. This was in 2015. Today our models are much more elaborated.

Theoretically, we need to quantify the value of this model, and to prove that the added-value of applying it is significant. Ideally, we may want to compare projects that employed this model with project that did not employ it yet. At this stage of the theory development this is impossible, because we cannot get the data that should be used for a comparative study, simply because there is no project yet that tried using the method. Even if we had such project, this method should be of low validity, because projects do not disclose their values.

A way to work around this limitation is by scoring made by intuition-based evaluation of experts in HSI design. Today, there are no experts yet that may evaluate the method, because it has not been published yet. The practice about this kind of preliminary situation is to get a consensus, similarly to that of the UML/SysML.

Most systems engineering conventions and practices are adopted by consensus, based on intuition, commonly called "common sense", not supported by research, for simple reason: it is impractical to design experiments on system development, in which the scoring is the real value, such that the conclusions will be of high face validity. It is impossible to apply proper experimental design to such paradigms.

This is the kind of support that we may expect when presenting new theories to practitioners. This discussion applies also to the other goal of the study, namely, demonstrating the feasibility of learning from rare events. A first step towards enabling learning from human errors and unexpected events, should be to develop tools for tracing and exploring the system activity.

The prototype of a universal HSI model was in two stages: first, defining the GMMs, and then organizing them in a structure.

E. Sampling

A preliminary version of the GMMs was developed earlier by Zonnenshain & Harel (2015)[56]. These GMMs were defined by analysis of 67 incidences, as patterns of typical system activity involved in the incidence. The present study repeated the analysis of these incidences, based on knowledge gained by analysis of additional case studies.

The study was based on 67 case studies reported elsewhere. The case studies are of three categories. Most of them are well-documented accidents. Others are anecdotal incidences due to minor flaws, reported by members of working groups on resilience assurance. Few case studies are of recurring, low-cost incidences.

An example of a case study is of the Three Miles Island accident, presenting two modes of failure of safety features:

- **The backup pump** was disabled during power generation
- **PORV** did not close after pressure release, backup pressure release was not provided.

F. A universal model of operation

Recently, Harel (2021)[23] has proposed a universal model of error-free system integration, consisting of seven layers of generic mini models (GMMs). The structure definition was based on analysis of the relationships between various entities that define the system behavior: functions, units, risks, states, events, reaction, and resilience. A prototype of a universal HSI model presented here consists of seven layers of GMMs as illustrated in the following figure:

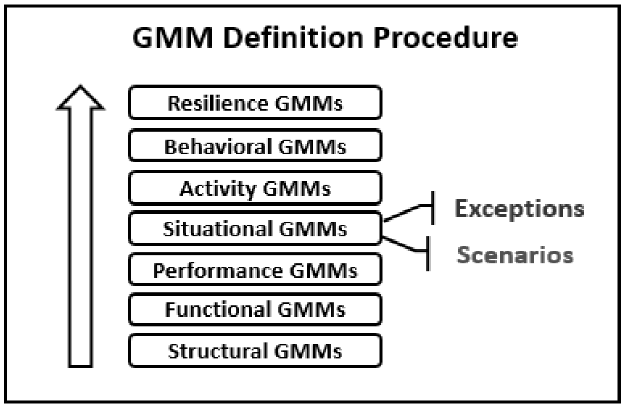

Fig. 12 Discretizing the system variables

The universal HSI model highlights the role of situational exceptions, as well as the role of scenarios, which should reduce operational complexity by information hiding, in support of direct mapping from intention to action.

A structural layer. This layer includes a break down of the system elements, including human agents (users, operators, artificial agents (processes, tools …) and sub systems. The system is a socio-technical, in which the operators are part of the system, but the users are external. The subsystems are coupled strongly with the operators, and loosely with the users.

A functional layer. This layer includes descriptions of the operational context and features of required to accomplishing an operator's task, intended to maximize the system utility.

A situational layer. This layer is contains representation of the operational situations. The design goal is to notify the operators about the system operating in exceptional situations. The core of the static model is an abstraction of the system situations, with a focus on exceptional situations.

An activity layer. This layer includes representations of the system dynamics. The design goal is to alert the operators about transitions from normal to exceptional situations. The core of the dynamic model is an abstraction of the system events, with a focus on unexpected events.

A behavioral layer. This layer includes definitions of the responses to events in various conditions. The design goal is to mitigate the risks of wrong responses to events. The core of the behavioral layer is an abstraction of typical system responses to exceptional events, with a focus on risk reduction. The focus is on assisting the operators in responding to rebound messages, in perceiving properly the risks associated with the alarms, and in troubleshooting.

A resilience layer. This layer includes representations of safety backups. The design goal is to mitigate the risks of operating with backup features missing or unavailable. The core of the resilience model is an abstraction of secondary risks due to the failure of safety features.

G. Model development

The GMMs may be developed gradually as incidences of new domains are added to the sample. Each cycle including the following activities:

- **Behavior abstraction**. This activity is the outcome of the incidence analysis. The goal of behavior abstraction is to transform domain-specific terms into universal, cross-domain terms. The abstract version of a specific incidence is an incidence model.
- **Model matching**. The objective of model matching is to identify common failure modes, namely, patterns of activities leading to incidences.
- **Protection evaluation**. The goal of protection evaluation is to detect and evaluate design features that may enhance reliability, namely, that may cope with the failure mode. Typically, this activity is serendipitous.

IV. Scenario-based Design

Root-cause analysis of operator's errors indicates that often they result from uncoordinated activity. Root-cause analysis of coordination failure indicates that often they are due to overriding interaction rules. Often, the reason for this is that the rules are not stated explicitly in the requirements documents.

A key design goal is to enforce operating according to the rules. Scenario-based design facilitates the coordination between the system elements and enables enforcing operation by the rules.

A. HSI scenarios

HSI scenarios are the HSI view of use cases/ usage scenarios. They are used for both design and testing. Operational complexity may be reduced by assigning the activity to scenarios.

Scenario-based design is essential to enabling seamless, carefree operation. Scenario-based modelling (SBM) is a procedure of activity design, in which the system activity is expressed in terms of operational scenarios. The objective of SBM is to support the design of seamless, robust, coordinated interaction by the rules. Trackers should be developed and integrated in systems, to enable evaluation of the effectiveness of this methodology.

B. Scenario models

A scenario model is a structure used to describe relationships, such as hierarchy and transitions between scenarios. It is the baseline for informal, normative, human-oriented, task-driven interaction design, as well as for disciplined system-oriented activity design. Often, it is a bundle of tree structures of scenarios associated with various system components.

The description may be similar to state-charts. Typical top-level scenarios of the system-level tree structure are generic, primary scenarios, such as: installation, initial setting, functional operation, initial training, advanced training, maintenance, testing, and problem solving. Typically, the problem solving scenario may break down to generic sub scenarios, such as: under hazard, under alarm, troubleshooting, safe-mode operation, resetting, recovery, and reporting. Further down, the “under alarm” scenario may be broken down to sub scenarios such as: low risk, high risk, and emergency.

Often, the lower levels are mostly domain specific. For example, the functional scenario of a commercial airplane may own three primary sub scenarios: takeoff, navigation, and landing. Further down the tree, the navigation scenario may own two sub-scenarios: manual navigation and automatic navigation. The bottom level may comprise project specific scenarios.

Component-level scenarios may be described by simple state trees, representing states about availability, reliability, activation, performance level, etc.

Scenario models may serve as a common vocabulary and a guide to system development. They simplify the definition of human-centered normative behavior, as well as features for enabling robust, carefree interaction.

The definition of scenario models may involve participation of customer, operator, and user representatives.

C. Normative interaction models

The goal of normative models is to envision how the system may be operated in normal scenarios. An interaction model is a presentation of the operation of primary tasks in terms of the scenario model. The modeling is based on participatory exploration by users and operators, by soliciting, analyzing, and elaborating stories about optional operational episodes and design alternatives.

The exploration may be employed using light, sketchy, agile simulation of the system operation. The simulation may have various forms, such as narratives, animation, role-play, board games, drama, or computer program. The simulation may employ various media, such as text, storyboards, video mockups, scripted, emulated, or real prototypes, or virtual reality.

D. Application to UI design

The root cause for many operator’s errors, such as in using consumer products, is due to erroneous activation of feature that should be available in different scenarios. For example, a most prominent problem in operating home appliance is the unintentional activation of setting features. This failure mode is the source of several famous accidents, such as the B-17 accidents due to control substitution in WW II, and the Torrey Canyon supertanker crash in 1967.

The scenario model may serve for designing the screens and panels, to prevent erroneous activation of features that do not comply with the active scenario.

E. Model realization

Coordination failure is often due to scenario ambiguity, in which different system elements assume different scenarios. For example, the friendly fire accident in Afghanistan (2001) is due to inconsistent assumptions about the operational scenario. Also, in other friendly fire accidents the fire support unit assumed a wrong phase of the fire plan. In order to enforce inter-element coordination, the design should include declaration and realization of the active scenario, to which all the relevant system elements should refer.

F. Situational models

A situational model is an expression of the system situation in the various scenarios. The system situation may be defined in terms of the states of system elements, such as units, agents, components, variables, procedures, and interaction options. In a situational model these are associated with scenarios. We may refer to these situations as the situational scope of the scenario.

A simple illustration of a situational model is an elementary system containing a device that may be On or Off, and a switch with two states, used to control the device. The functional scenario of the situational model may comprises two sub scenarios of normal operation:

- **Operative**: both the device and the switch are On
- **Idle**: both the device and the switch are Off.

Another example, illustrating the need for situational modelling, is demonstrated by the accident involved in operating Therac 25 radiotherapy equipment, which was operated in two normal functional scenarios:

- **X-ray testing**: obtained by high current, moderated electron beam
- **E-beam treatment**: obtained by low current, full electron beam

The accident was due to operating in an exceptional situation, of high current, full electron beam.

Other combinations of the device and switch states are out of the scope of the functional scenario, and are regarded as exceptional. The Torrey Canyon supertanker loss of control accident (LOCA) demonstrates the need to impose operation based on situational models. In this supertanker, the navigation control lever had three positions: manual, automated, and special position, disconnecting the rudder from the wheel. The special position was intended for use in maintenance only. The LOCA resulted from accidental selecting the special position while on board.

Continuous variables may be associated with scenarios by their distribution functions. For example, the available disk space of a computer may be either normal or critical. Accordingly, the situational model of the computer disk space may own two scenarios.

Thresholds of any continuous variable, such as container temperature, may define various performance scenarios, such as normal, low risk, and high risk. The Bhopal disaster demonstrates the need to impose operation based on situational models of continuous variables.

Continuous variables may also represent scenarios about external, contextual, or environmental situations, such as ambient humidity, as well as about time measurements of repeating activities.

G. Situational rules

Situational models enable structuring a framework of operational rules. According the principles of cybernetics, adopted for the STAMP paradigm, systems should operate according to rules. Many incidences may be attributed to ambiguous, implicit operational rule. For example, the rules defining the properness of the operation of the elementary system are derived from the situational models of the Operative and Idle scenarios. If these rules are implicit, then the system might not be detect exceptional situations, such as when the switch is Off and the device is On.

Situational rules may consist of conditions and reaction. The conditions may be expressed as boolean expressions of states. The reaction may be preventive, by enforce a proper operation, or defensive, for example, by rebounding or notifying the operators about the rule violation. The reaction part may reflect our prediction of the costs of the reaction options.

Situational rules are attributes of scenarios. Examples of situational rules are:

- In functional computer operation, when the available disk space is critically low, the system should advice the operator to clean it.
- In the production of dangerous materials, when the container temperature is higher than a safety threshold, the system should notify the operators and enforce safe-mode operation.

Examples of generic rules:

- When in a functional scenario, risky features should be disabled. The need for imposing this rule is demonstrated by the Torrey Canyon and the Afghanistan friendly fire accident, and many other.
- During the operation of safety-critical scenarios, safety backup features should always be available and enabled. The TMI accident (1979) demonstrates the risks of erroneous disabling of the backup pump.

Typically, the definition of situational rules is in the scope of systems engineering. The validation of the situational rules may be based on faking exceptional situations, and evaluating the HSI reaction to the faked situations.

H. Rule based exceptional handling

A situation is regarded as exceptional if it does not comply with the rules applicable to the active scenario. The best design strategy to enforce compliance with the rules is by disabling or avoiding exceptional situations. Method for avoiding exceptions include rebounding from errors, or providing the operator with a forecast of the effect of optional events.

Exception handling is required when we cannot prevent the exception, in cases when the exception is due to an external hazard, a hardware fault, a power failure, or a communication interrupt, or a design or implementation mistake. The design should provide means to accommodate them, by notifying the operators about operating in high-risk situations, by prompting the operators to take actions, and by guiding them in the recovery procedure.

I. Unexpected situations

The situational model includes only part of the situations, those included in the situation scope of the scenarios. Most of the situations are not included in the scope of any of the scenarios. For example, in the elementary system describe earlier, only two of the four combinations are expected. Similarly, in the Therac 25 example, only two of the four combinations of current- electron beam are expected. In hindsight we know that the Therac 25 accidents are due to operating the system in a mixed mode of high current and full electron beam, which is not in the situational scope of X-ray testing scenario, nor of the E-beam treatment scenario. These situation are unexpected, and their root may be in mistakes in the definition of the situational rules, or in bugs.

The challenge is of handling unexpected situations: the system design should prevent them, and notify the operators about operating in such situations. Special safe-mode procedures may be designed to handle them.

J. Activity models

The system activity may be defined in terms of the system reaction to events. Typically, the reaction depends on the operational conditions, which are defined by the system situation and by external conditions. An activity model is a description of the activities constrained by scenarios. It may be expressed in terms of activity rules.

K. Activity rules

The activity rules define the reaction to events in terms of scenarios. An activity rule may describe normal interaction, or ways to prevent diversion from normal to exceptional situation. Interaction rules define optional responses to an event, in a particular situation, depending on the scenario. Examples of preventive rules are.

- Safety features should not be disabled while in high-risk scenario.
- Transition to a functional scenario should be avoided when any of the safety features is disabled.

Typically, the definition of activity rules is in the scope of systems engineering.

L. Protective rules

Protective rules may be derived from situational rules by examination of the possible transitions from normal situations to exceptional situations.

For example, examine the situational rule about the availability of safety features during safety-critical functional scenarios. Depending on the costs of automated suspension of the functional operation, the system may either suspend the functional operation, or notify the operators about the risks of operating without the safety feature. Protective rules derived from this situational rule are:

- The system should prevent or warn the operators about disabling the safety feature while in a functional scenario
- The system should prevent scenario transition from maintenance to functional when the safety feature is disabled.

The validation of the protection rules may be based on faking exceptional situations or events.

M. Activity protocols

The activity rules may be formalized in terms of protocols of event-response. The responses to events may include changing of the operational scenario. The activity model may include special protocols for handling the operator's control. For example, a protocol for responding to disabling a safety feature in a functional scenario may consist of two steps:

1. Rebounding: prompting the operators to regret or to confirm their intention
2. Switching to a safe scenario, such as maintenance, idle, safe-mode or shutting down.

N. Transition synchronization

Following a request to change the active scenario, the system needs to activate the situational rules that apply to the new scenario. By definition, changing a situational rule of a scenario involves changing of a state of at least one system element. Changing the state of a system element may be time consuming. The Therac 25 accident demonstrates a challenge of responding gracefully to synchronization delay, and to suspending the operation until the scenario transition is complete.

Transient scenarios define the system response to events during the transition. During a transient scenario, the system may operate in a special sync mode. The design should include special features for enforcing graceful synchronization, such as disabling risky activity, notifying the operators while in synchronization, warning the operators in case of failure, and handling the recovery.

While in a transient scenario, the system may operate in a special transition mode. The operation in the transition mode may be initially automated, by default. If applicable, the operators may have an option to override the automated behavior.

O. Transition models

A transition model is a description of the procedure for changing the situational rules during the scenario transition. Transition models may describe ways to capture and notify on exceptions, and escape procedures, in response to exceptions.

The transition model may include a special transient scenario, representing the operation until the new scenario is synchronized, and a special escape scenario, representing the case of transition failure. The operators need to know about such cases, and the system should provide an exception warning when the situation does not comply with the new constraints. The transition model may include special features for enforcing graceful delay or failure, such as disabling risky activity and notifying the operators while in the transient scenario.

A generic synchronization model may be expressed using a standard protocol, including:

- A transition request, pointing at the target scenario, and setting a sync time out limit
- Activating processes aimed at applying the rules associated with the target scenario
- Waiting until the situation complies with the rules of the target scenario. While waiting, the system should indicate that the system is in a transient scenario
- After complying with the rules of the target scenario, it becomes the active scenario
- In case of reaching the timeout limit, provide a warning message and initiate a recovery procedure.

P. Transient timeout adjustment

An initial value of the sync timeout may be defined in the transition specification, but this value might not fit all circumstance. The design may provide means for measuring the actual transition time, and for adjusting the timeout for each of the transitions, based on statistics of the measurements. The adjustment may be automated or manual.

Q. Recovery models

A generic recovery model may be expressed using a standard protocol, including:

- Notifying the operators about the transition failure, prompting to recover the situation prior to the transition request
- Notifying the operators about the recovery results
- Enter a special safe-mode operation.

V. ENGINEERING

As discussed by Harel & Zonnenshain (2019)[25] the engineering of HSI is based on defining operational rules, which define exceptions by exclusion from normal behavior.

A. HSI statecharts

SysML offers a simplified version of UML statecharts for graphical representation of state transitions. This kind of representation is not adequate for modeling the interaction between state machines. The problem is that events designed using SysML statecharts are error-prone. The HSI version of statecharts supports describing various attributes of mutual effects between state machines, as well as enforcing error-free state transitions.

B. Evaluation

For evaluating the model, we may employ the Layer Of Protection Analysis (LOPA) technique, commonly used in the process industry for assessing the protection needs. The evaluation is based on testing the effects of protection layers and calculating the potential risks (Baybutt, 2002)[4].

C. Infrastructure

Utility-critical systems should incorporate means, including sensors, trackers, recorders, and analyzers, for informing the operators and the developers about the time they could save.

The infrastructure for model-based HSI may include special means intended to save the time wasted in handling exceptional situations. The means to avoid exceptional situations and to support exception management may be integrated into the model used to design the HSI. For example, they may include model interpreters that enable customizing the model transition to software units.

D. *Data analytics*

Tracking tools enable capturing and measuring the costs of daily, low-cost events (Harel, 1999)[17]. Harel et al. (2008)[24] demonstrated a way to apply data analytics in automated usability testing, and Harel (2009)[19] demonstrated that data analytics may be used to identify problem indicators. Universal tracking is crucial also for enabling learning from rare events.

E. *Digital twins*

A digital twin is an executable virtual model of a physical thing or system (Wright & Davidson, 2020)[55]. The concept of digital twins is based on the concept of virtual prototyping, dated in the 80s, in which a model was used to replace system units by emulation. This features enables early integration, by using virtual units instead of the real components that are not ready yet for the integration. This feature was recently adopted for systems engineering in the form of digital twins.

Digital twins enable to control the system operation according to the STAMP paradigm: the post-deployment emulation enables detection of incidences by comparing the output of the emulated unit with that of the real unit, as depicted in the following figure:

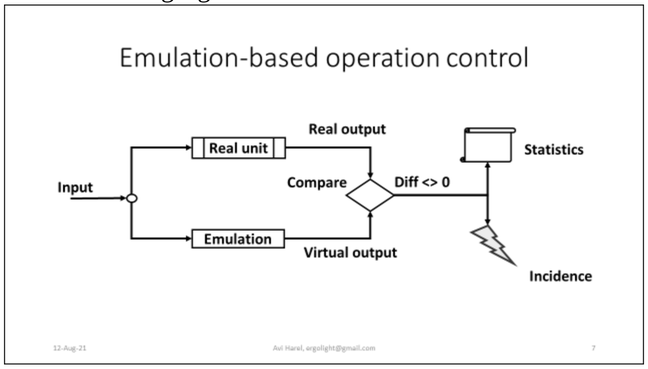

Fig. 13 Digital twins used for incidence detection

Digital twins may be integrated in MBHSI, for seamless change validation according to the following figure:

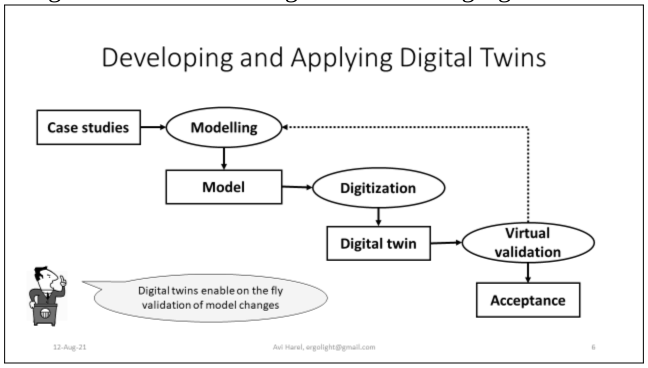
Developing and Applying Digital Twins
Case studies
Modelling
Model
Digitization
Digital twin
Virtual validation
Digital twins enable on the fly validation of model changes
Acceptance
12-Aug-21
Avi Harel, ergolight@gmail.com
6

Fig. 14 The role of digital twins in model based HSI

F. *Customizing*

The seven-layer models are generic, applicable to various domains and industries. To adapt it to a particular project these models need customization. The customization process is according to the order above, as the definition of each model depends on that of the previous one.

G. *Simulation*

The transition from the customized model to a prototype and/or digital twin should be automated. The automation may be based on simulation of the orchestrated version of the system, using standard software packages that process the custom parameters.

H. *Model development*

Models enable saving development costs by enforcing seamless adaptation to design changes. The models should be defined iteratively, each cycle is followed by evaluation. Typically, the evaluation ends up with a list of requirements for design changes, intended to reduce the operational complexity. The development might end when it is obvious that all known significant risks are removed. Criteria form ending the development may be based on the Service Integrity Level (SIL) evaluation method commonly applied in the process industry (Redmill, 2000)[40].

I. *Testability*

Testing rare events is challenging. To enable testing exceptions the system should incorporate a special tester unit that fakes various kinds of faults, in various conditions, that the testing team can customize. A special scenario should be defined, which is part of the operational conditions.

J. *Adjustability*

The setting of the alarm and safety thresholds of the various risk indicators is a delicate design goal, aiming to balance properly the rate of nuisance of the alarms. A special utility may enable inform the system administrators about the margins of alarms and safe-mode operation.

VI. CONCLUSIONS

Primary barriers to maximizing the utility are limitations of operating in exceptional situations, typically attributed to errors, hampering the system's usability.

The conclusions from this study are that we can learn from case studies drawn from various domains and industries, and formulate a universal HSI model. This model may consist of layers of GMMs, expressed in terms of rules for system definition.

Principles of HSI reliability may be phrased as scenario-based rules and protocols for risk detection, recognition, and identification. A challenge for the 4th industrial revolution is to develop a methodology for cross-industry model-based integration design. This study demonstrates that we can define universal rules, suggesting that this goal is achievable.

The article explores various protection patterns, but certainly not for all possible design challenges. It may be interesting to explore operational rules for various tasks, such as unsupervised learning or for adaptive systems.

Some rules for adaptive systems were proposed elsewhere, based on the experience in using them, and by modeling the behavior in various situations. These rules rely on scenario definition, based on the skill level of the users. The measures examined may be obtained by statistics of measurements of the system performance. Validation of the rules may be conducted by analysis of the activity obtained by trackers of the system performance, using statistical metrics, followed by traditional usability testing in the corresponding scenarios.

ACKNOWLEDGMENT

I thank Avigdor Zonnenshain, Uzi Orion, Moshe Weiler, Sharon Shoshani, Ami Harel and other members of INCOSE-IL, Gordon Center for Systems Engineering, Iltam, Israel Resilience working group, Israeli HSI working group, and HSI international working group for their support and for providing helpful comments on previous works on the subject, and also the HSI2021 reviewers who provided helpful comments on the previous version of this article.

REFERENCES

[1] Bainbridge, L 1983, Ironies of automation. *Automatica. 19 (6)*: 775–779. doi:10.1016/0005-1098(83)90046-8. ISSN 0005-1098

[2] Baker, CC & Seah, AK 2004, Maritime Accidents and Human Performance: the Statistical Trail Paper presented at *MARTECH 2004, Singapore*, September 22-24

[3] Barricelli, BR, Casiraghi, E and Fogli, D 2019, 'A Survey on Digital Twin: Definitions, Characteristics, Applications, and Design Implications', *IEEE Access*, November, PP(99):1-1

[4] Baybutt, P 2002, Layers of Protection Analysis for human factors (LOPA-HF), *Process Safety Progress* 21(2):119 – 129, DOI:10.1002/prs.680210208

[5] Berard, J 2013, *Accelerating Leadership Development: Practical Solutions for Building Your Organization's Potential*, John Wiley & Sons, 25 Jul 2013.

[6] Boy, GA, 2013, *Orchestrating Human-Centered Design.* New York: Springer. ISBN 978-1-4471-4338-3

[7] Checkland, PB 2001, Soft Systems Methodology, in J. Rosenhead and J. Mingers (eds), *Rational Analysis for a Problematic World Revisited*. Chichester: Wiley

[8] Clark CW, & Dukas R 2003, The behavioral ecology of a cognitive constraint: limited attention. *Behav Ecol* 14:151–156.

[9] Cohen, J 2001, *A Tribute to Alain Colmerauer. Theory and Practice of Logic Programming. 1 (6): 637–646.*

[10] Copeland, S 2020, On serendipity in science: discovery at the intersection of chance and wisdom, *Synthese: an international journal for epistemology, methodology and philosophy of science*

[11] Dekker, S 2012, *Just culture: Balancing safety and accountability*, Ashgate.

[12] Drury CG, Woodcock K, Richards I, Sarac A, Shyhalla K. A New Model of how People Investigate Incidents. Proceedings of the Human Factors and Ergonomics Society Annual Meeting. 2002;46(13):1210-1214. doi:10.1177/154193120204601343

[13] Endsley, MR 1995, Toward a theory of situation awareness in dynamic systems. *Human Factors. 37 (1)*: 32–64. doi:10.1518/001872095779049543

[14] Fuhs, A 2008. Hybrid vehicles: and the future of personal transportation. *CRC press*.

[15] Gabriel, RP & Steele, GL 2008, A Pattern of Language Evolution. *LISP50: Celebrating the 50th Anniversary of Lisp*. pp. 1–10.

[16] Grimm, T 1998, The Human Condition: A Justification for Rapid Prototyping. *Time Compression Technologies, vol. 3 no. 3*. Accelerated Technologies, Inc. May 1998

[17] Harel, A 1999, Automatic Operation Logging and Usability Validation, *Proceedings of HCI International '99*, Munich, Germany, Vol. 1, pp. 1128-1133

[18] Harel, A 2006, Alarm Reliability, *User Experience Magazine*, Vol 5., Issue 3.

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

[20] Harel, A 2010, Whose Error is This? Standards for Preventing Use Errors, *The 16th Conference of Industrial and Management Engineering*, Tel-Aviv

[21] Harel, A 2011, Comments on IEC 60601-1-8. *Letter submitted to IEC/TC 62 working group*.

[22] 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

[23] Hare, A 2021. Towards Model-based HSI Engineering: A Universal HSI Model for Utility Optimization, to be published in *Proceeding of the second HSI conference,* San Diego, US.

[24] Harel, A, Kenett, R & Ruggeri, F 2008, - Modeling Web Usability Diagnostics on the basis of Usage Statistics. in: *Statistical Methods in eCommerce Research*, W. Jank and G. Shmueli editors, Wiley.

[25] Harel, A & Zonnenshain, A 2019, Engineering the HSI. *Proceedings of the first HSI conference*, Biarritz, France

[26] Haselton MG, Nettle D, Andrews PW 2005. 'The evolution of cognitive bias.' (PDF). In Buss DM (ed.). *The Handbook of Evolutionary Psychology*. Hoboken, NJ, US: John Wiley & Sons Inc. pp. 724–746.

[27] Hatchuel, A, Le Masson, P & Weil, B 2011, Teaching innovative design reasoning: How concept–knowledge theory can help overcome fixation effects. Published online by Cambridge University Press

[28] Hollnagel, E 1983, Human Error. Position Paper for *NATO Conference on Human Error*. Bellagio, Italy.

[29] Hollnagel, E 2006, Resilience: The challenge of the unstable. In: Hollnagel, E., Woods, D. D. & Leveson, N. C. (Eds.), *Resilience engineering: Concepts and precepts* (p. 9-18). Aldershot, UK: Ashgate.

[30] Hornikx, J 2018, Combining Anecdotal and Statistical Evidence in Real-Life Discourse: Comprehension and Persuasiveness. *Discourse Processes Vol 55*, Issue 3.

[31] Jackson, S & Ferris, T 2013, Resilience Principles for Engineered System. *Systems Engineering, 16(2)*, 152-164. doi:10.1002/sys.21228.

[32] Jacobson, I 1987, Object-oriented development in an industrial environment. *ACM SIGPLAN Notices*. 22 (12): 183–191.

[33] Kariuki, SG & Loewe, K 2006 Increasing Human Reliability in the Chemical Process Industry Using Human Factors Techniques, *Process Safety and Environmental Protection* 84(3):200-207

[34] Leveson, N 2004, A New Accident Model for Engineering Safer Systems. *Safety Science 42(4)*:237-270

[35] Luqi 1989, Software Evolution through Rapid Prototyping. *IEEE Computer. 22 (5):* 13–25. doi:10.1109/2.27953. hdl:10945/43610

[36] Norman, DA 1983, Design Rules Based on Analyses of Human Error. *Communications of the ACM 26(4)*:254-258

[37] Norman, DA and Draper, S 1986, *User Centered System Design: New Perspectives on Human-Computer Interaction* Lawrence Erlbaum Associates.

[38] PlaneCrashInfo, 2014, *Causes of Fatal Accidents by Decade* http://planecrashinfo.com/cause.htm

[39] Shapiro, E. 1983, *Algorithmic program debugging.* Cambridge, Mass: MIT Press. ISBN 0-262-19218-7

[40] Sheard, SA and Mostashari. A 2009, Principles of complex systems for systems engineering. *Systems Engineering, vol. 12, no. 4*, pp. 295-311.

[41] Shewhart, WA 1931, *Economic Control of Quality of Manufactured Product* ISBN 0-87389-076-0

[42] Shneiderman, B 1980, *Software Psychology: Human Factors in Computer and Information Systems*. Little, Brown

[43] Shneiderman, B 1986, *Designing the User Interface: Strategies for Effective Human–Computer Interaction*, 1st edition. Addison-Wesley

[44] Singh, S 2015, *NHTSA CrashStat*, Critical Reasons for Crashes Investigated in the National Motor Vehicle Crash Causation Survey, DOT HS 812 115

[45] Spool, JM. 2014, Scenarios and Journey Maps Help Designers Become Storytellers. *User Interface Engineering*, May 7.

[46] Swatton, PJ 2011, "14.11", *Principles of Flight for Pilots*, Chichester, UK: Wiley & Sons Ltd

[47] Taleb, NN 2007, *The Black Swan: The Impact of the Highly Improbable*. Random House Trade Paperbacks.

[48] Weaver, W 1948. Science and complexity. *American Science, vol. 36*, pp. 536-544.

[49] Weiler, M & Harel, A 2011, Managing the Risks of Use Errors: The ITS Warning Systems Case Study, *The Sixth Conference of INCOSE-IL*, Hertzelia, Israel.

[50] Weinberg, GM 1971, *The Psychology of Computer Programming*. Van Nostrand Reinhold.

[51] Wheeler, DJ & Chambers, DS 1992, *Understanding Statistical Process Control* ISBN 0-945320-13-2

[52] Wickens, CD 1992, *Engineering psychology and human performance (2nd ed.*). Harper Collins Publishers.

[53] Wiener, N 1948, *Cybernetics; or, Control and communication in the animal and the machine*. Technology Press, Cambridge.

[54] Wilson, TD 1999, Models in information behaviour research, *Journal of Documentation*, *Vol. 55 Iss 3* pp. 249 – 270

[55] Wright, L., Davidson, S. How to tell the difference between a model and a digital twin. *Adv. Model. and Simul. in Eng. Sci*. 7, 13 (2020). https://doi.org/10.1186/s40323-020-00147-4

[56] Zonnenshain, A & Harel, A 2015, A practical guide to assuring the system resilience to operational errors, *INCOSE. Annual International Symposium, Seattle.*

Avi Harel received his B.Sc. (1970) and M.Sc. (1972) degrees in mathematics from the Technion, the Israeli Institute of Technology, in Haifa, Israel. For his M.Sc. degree he received an outstanding grade. For his master's thesis he was granted the Landau's award. Between the years 1985 and 1989 Avi studied Behavioral and Management Sciences at the faculty of Industrial Engineering of the Technion**.**

Between 1975-1992 he worked for Rafael, the Armament Development Authority of Israel, during which he gained experience in working with a wide range of applications, platforms, operating systems, programming languages and development environments. Between 1977-1980 he was the manager of 30 people in the Software Department of Rafael's Division of Electronics. Between 1980-1983 he was the manager of the leading project of the Electronics Division of Rafael. Between 1983-1985 he designed the software for a touch operated telephone set for the Design Interpretive department of BNR, Canada. Between 1985-1987 he developed a generator of user interfaces, for use by frequent users. Between 1988-91 he conducted various projects in Human Factors engineering in Rafael. His work experience includes software engineering, system engineering and ergonomics in Rafael, Nortel, IBM, Attunity and Ergolight. Since 2007 he focuses on developing and publishing methods for preventing human errors in system operation. This activity is conducted in collaboration with the Israeli branch of INCOSE and with the Gordon Center for Systems Engineering of the Technion, Haifa.