Scenario-based modeling

Avi Harel

Ergolight, Haifa 3433506, Israel
ergolight@gmail.com

Abstract. A primary goal of software design is to maximize operational utility. The utility of software depends on the performance, constrained by the performance envelope, defined by operational rules, which set the contract between the software and the engineers. The performance envelope comprises rules for inter-element coordination, and for setting and controlling the activity. Rule-based scenarios enable simplifying the definition of the performance envelope, which is complicated otherwise. Scenario-based modeling (SBM) is a cross-domain abstraction of the operational rules. The objective of SBM is to support the design of seamless, robust, coordinated interaction by the rules. The goal is to define universal, generic models, which may be customized to a project through parameters. The models developed may comprise basic definitions of utility, performance envelope, interaction scope, scenario structures and transitions, situational rules, inter-element activity protocols, procedures for normal interaction, for adjusting the performance envelope, and for recovery from failure. The article presents basic principles, and a universal seven-layer model of normal and exceptional interaction. A way to validate a model-based interaction is by faking triggers and exceptions, and by comparing the actual interaction with that of a virtual twin. The article calls for developing and integrating tools for tracking, recording, and analysis of the interactions in upcoming projects, which may enable assessment of the actual costs of incidences, as well as validation of the universal models proposed here.

Keywords: Software Engineering, Systems Integration, Error Prevention, Rare Events. Safety, Usability, Formal Methods, Scenario-based Modeling.

1 Objectives of Scenario-based Design

Primary goals in software design are to maximize the operational value and to support agile development. This section describes a model of the operational value. The second section describes how scenario-based modeling may contribute to mitigating the costs of incidences. The third section discusses engineering considerations.

1.1 Operational Performance

A common practice for assessing the system value is in terms of utility. Operational utility is often defined in terms of performance, which depends on the system purpose and functions. Typically, performance is associated with metrics such as throughput, bandwidth, power consumption, etc. which are industry and domain specific. The problem with objective measures is that they do not represent implicit factors, which are not testable, affecting the system value to the stakeholders.

Often, the implicit factors are more important than those that are measureable and testable. Accordingly, the Cambridge dictionary defines performance using general terms, as “how well a person, machine, etc. does a piece of work or an activity”. Typically, the term utility refers to the perceived efficiency in achieving the system primary, goal, which is typically subjective, implicit, and biased.

1.2 The Performance Envelop

Systems are designed to optimize performance when operating in nominal conditions. Hence, by definition, when diverting from the optimal conditions, the performance should reduce. Moreover, in extreme conditions, the operation might fail, a situation called ‘incidence’ (Hollnagel, 2006) [13]. The term ‘performance envelope’ refers to the limits of the performance, namely, the situations in which the operation results in incidences. An incidence is an event of invading the performance envelope.

The concept of performance envelope extends that of flight protection envelope, imposed to protect aircrafts in extreme conditions. It is an expression of the performance reduction due to constraining the operation to avoid incidences. 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)[20].

1.3 Operational Utility

The operational utility is affected not only by the design features, but also by the constraints, defined by the performance envelope. The utility is an expression of the actual performance, when avoiding the risks of invading the performance envelope, as illustrated in the following figure:

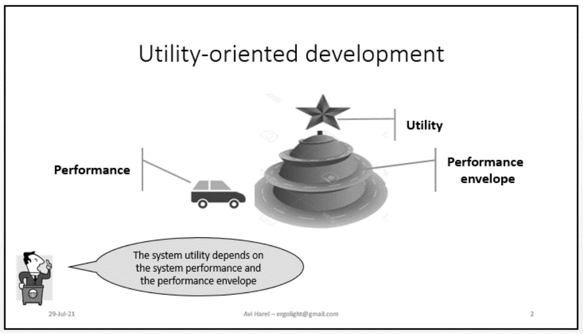

Utility-oriented development
Utility
Performance
Performance envelope
The system utility depends on the system performance and the performance envelope
29-Jul-21
Avi Harel – ergolight@gmail.com
2

Fig. 1. Utility-oriented development

In practice, the system utility varies during the system evolution. It grows during the initial learning phase, it fades out when the system is getting old. Accordingly, the system value may be defined as the accumulated utility over the life cycle.

1.4 The Costs of Incidences

Current design practices focus on functions, not on incidences. Therefore, most systems are not equipped with means to detect and trace incidences. The actual costs of incidence are known only when they are extremely high, namely, in accidents. In most cases, they remain unknown. However, informal studies, such as those reported by Harel (1999)[5] indicate that the accumulated costs of overlooked incidences are very high, as depicted in the following figure:

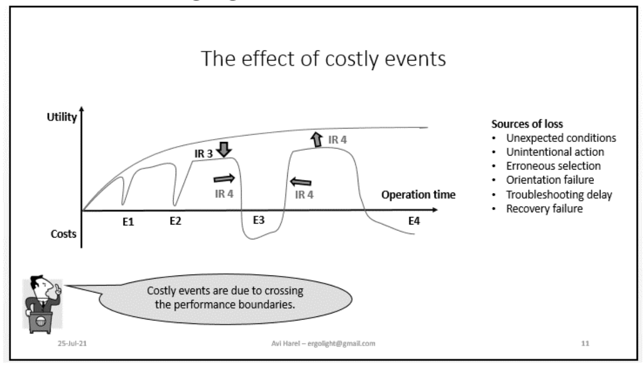
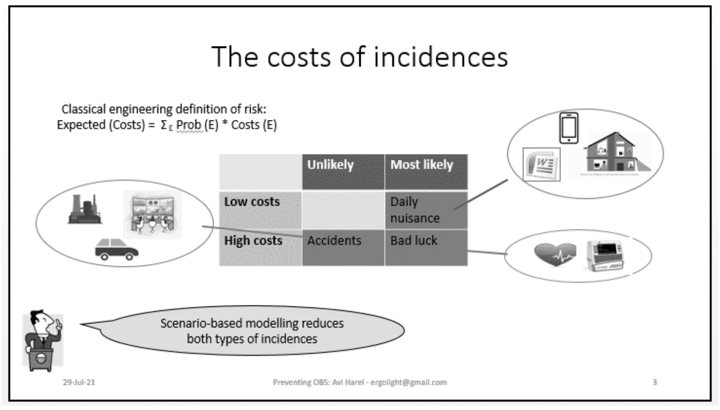


Fig. 2. The effect of costly events

We may distinguish between two types of incidences: those due to rare events and those due to daily, low-cost events.



The costs of incidences

Classical engineering definition of risk:
Expected (Costs) = Σ_E Prob (E) * Costs (E)

	Unlikely	Most likely
Low costs		Daily nuisance
High costs	Accidents	Bad luck

Scenario-based modelling reduces both types of incidences
29-Jul-21
Preventing OBS: Avi Harel - ergolight@gmail.com
3

Fig. 3. Risks of operational complexity

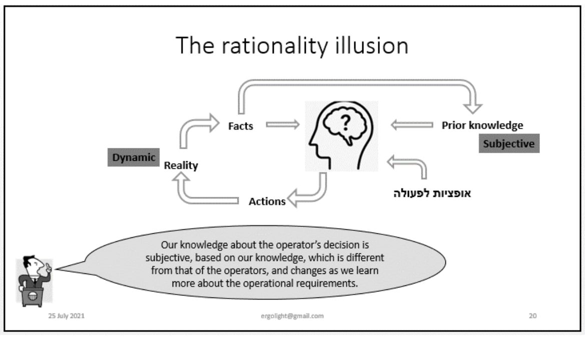
Taleb (2007)[21] argued that it is impossible to predict incidences due to Black Swans (rare events) because a-priori we do not have the data required for the prediction. Explicit incidences are those that in hindsight turn out to be costly.

1.5 Modeling the Human Operator

Human Performance. Early computer-embedded systems suffered from usability problems, resulting in productivity and safety loss (Landauer, 1995)[15]. The traditional view of performance is based on the wrong assumption that the operators do their job perfectly. The human factors view of performance is about the limitations of the human operators, such as attention deficit when in stress, when the attention demands are high, such as in uncertainty, or in multi-tasking (e.g. Wickens, 1992)[24].

Operational Complexity. Costly incidences are associated with operator errors in handling exceptional situations (Zonnenshain & Harel, 2015)[28]. The reason for this is that regular training practices assume operating in normal conditions. During normal operation, the operators encounter exceptional situations only occasionally, in a rate that 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 human productivity when doing text editing indicate that about half of the time is wasted on recovering from errors. The implication of this observation is that rather then investing on error analysis, the design should focus on preventing failure. According to the proactive approach to failure, the system design should support the operation also in extreme conditions.

Rationality. Another utility factor is about the operator's rationality. Usually, we assume that the operators are rational. 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 (Harel, 2020)[8] as illustrated in the following figure:



Our knowledge about the operator's decision is subjective, based on our knowledge, which is different from that of the operators, and changes as we learn more about the operational requirements.

Fig. 4. The rationality illusion

Errors. Since the 80th, cognitive psychologists study why and how people err (e.g. Norman, 1983) [16]. It turns out that errors emerge spontaneously, in various circumstances, and that it is hopeless to expect that people may ever stop making errors. The article advocates the human factors 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. Errors are essential for learning (Wiener, 1948)[25]. However, errors are costly:

- **High-cost 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)[17] sea (80%, Baker & Seah 2004)[2], driving (90%, Singh 2015)[19], and in the industry (60-80%, Kariuki & Löwe 2006)[14].
- **Minor errors**. Human errors are considered as the primary source of operational loss: by accidents, damage to property, low productivity, or user dissatisfaction (Landauer, 1995)[15]. Many of the usability issues in operating consumer products, such as home TV, are commonly attributed to use errors (Zonnenshain & Harel, 2009)[26].

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 malfunction, operators' errors. In many cases the loss is attributed ad hoc to the person who happened to be on duty at the time of the event (Dekker, 2007)[4].

Modeling Human Errors. Errors are incidental. Weinberg (1971)[23] reported on typical subconscious design mistakes, due to egocentric programming, hampering the productivity of the computer users. Shneiderman (1980)[18] promoted the concept of egoless programming suggested by Weinberg, and proposed few principles for avoiding such design mistakes. Norman (1983)[16] 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)[23] argue that judgment errors under stress are due to relying on irrelevant prior experience.

The Vendor's Responsibility. The article advocates a design goal of protecting the system from human errors. The article assumes a variant of Taleb's Black Swan theory (2007)[21], illustrated in the following figure:

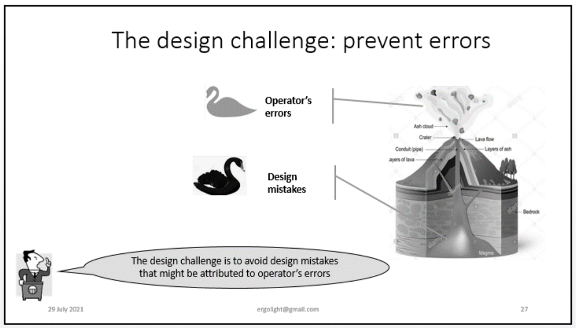


Fig. 5. The design challenge

The article assumes that most usability problems may be attributed to human limitations to perform perfectly in extreme conditions, such as exceptional situations due to design mistakes and bugs. The article assumes the proactive version of Murphy's Law, attributing operational problems to design defects, of enabling the operational problems. It is the design's responsibility to prevent situations in which the operators might fail.

Biasing. It is not in the scope of the system design to enforce proper operation, as expected or specified by the developers. Following Hollnagel (1983)[12] 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. 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 (Harel, 2010)[6], as demonstrated in the following figure:

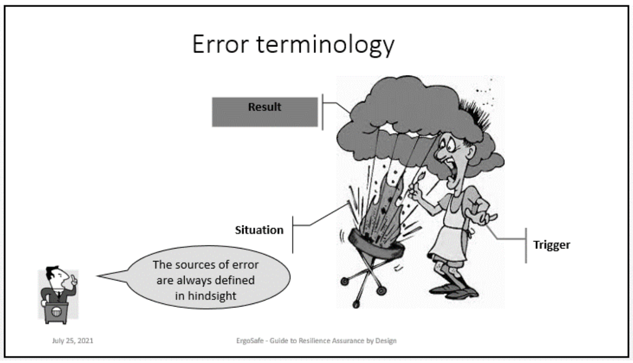

Error terminology
Result
Situation
Trigger
The sources of error are always defined in hindsight
July 25, 2021
ErgoSafe - Guide to Resilience Assurance by Design

Fig. 6. Error Terminology

According to Zonnenshain & Harel (2015)[28] 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)[7] analyzed various ways in which vendors of equipment for medical alarms infect the standards by diverting the accountability for failure to the operators.

1.6 Modelling the System Integration

Hollnagel (2006)[13] proposed two ways for modeling the system integration. The proactive approach is about how to describe normal behavior, and the reactive approach is about how to describe extreme events. The need to present formal solutions was elaborated by Harel & Zonnenshain (2019)[11]. The means to avoid exceptional situations and to support exception management may be integrated into the model used for the interaction design. A method and procedure for modeling the system integration was described by Harel (2021)[9]. The procedure has seven steps, according to a sever-layer model of the integration. Integration modeling is a hybrid approach, in which we define normal behavior proactively, and we apply learning from failure. The reactive part is by serendipitous learning from incidences (e.g. Copeland, 2020)[3].

2 Scenario-based Design

The dictionaries define “scenario” as “a sequence of events, especially when imagined”. In the context of system integration, the term refers to events that should be expected during the system operation. Scenario-based design facilitates the coordination between the system elements and enables enforcing operation by the rules. The focus is on generic models intended to optimize the system value. It is essential for enabling seamless, carefree operation.

2.1 Modeling the System Integration

Model-bases systems integration is critical to enabling agile development, essential for reducing the development costs. 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.

Barriers to seamless operation include instances of confusion and hesitation of the operators, due to anxiety about potential loss. Typically, the confusion is attributed to operator’s errors. Root-cause analysis of operator’s errors indicates that often they result from uncoordinated activity or from overriding implicit interaction rules. A key design goal is to enforce operating according to the rules. Scenario-based design facilitates rule-based coordination between the system elements.

2.2 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 of a scenario model 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. Developing scenario models may involve participation of customer, operator, and user representatives.

2.3 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.

2.4 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 (Zonnenshain & Harel, 2013)[27]. 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.

2.5 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 the situations 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 constrain the operation. A way to constrain the operation is based on situational models (Weiler & Harel, 2011)[22]. 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.

2.6 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 (Weiler & Harel, 2011)[22].
- 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 (Harel & Zonnenshain, 2019)[11].

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 system reaction to the faked situations.

2.7 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.

2.8 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 (Harel & Weiss, 2011)[10].

2.9 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.

2.10 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.

2.11 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.

2.12 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.

2.13 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.

2.14 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.

2.15 Recovery Models

A generic recovery model may be expressed using a standard protocol for notifying the operators about the transition failure, prompting to recover the situation prior to the transition request, notifying the operators about the recovery results, and entering special safe-mode operation session.

3 Discussion

3.1 Engineering

Engineering the scenario-based models may comprise the following topics:

- **Customizing**. The seven-layer model mentioned earlier (Harel, 2021)[9] is 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 layers; each layer depends on the previous one.

- **Parameter development**. 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.
- **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 informing the system administrators about the margins of alarms and safe-mode operation.
- **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 of the target system, using standard software packages that process the custom parameters.
- **Error-proofing**. Primary barriers to maximizing the utility are limitations of operating in exceptional situations, typically attributed to errors. These barriers result in hampering the system's usability. 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 (Weiler & Harel, 2011)[22].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.
- **Twin-based verification**. Digital twin are essential for controlling the system operation according to the STAMP paradigm: post-deployment emulation enables detection of incidences by comparing the output of the emulated unit with that of the real unit MBSE is essential for seamless change validation based on a digital twins.

3.2 Model Development

Following Harel & Zonnenshain (2019)[11] system integration may be based on models consisting of rules defining normal situations and activity, which may apply across various industries and domains. Typical activities in model development include:

- **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. Trackers should be developed and integrated in systems, to enable evaluating the effectiveness of this methodology.

- **Learning**. Harel (1999)[5] has demonstrated the potential benefits of tools for tracking and analysis of the system activity. Utility-oriented engineering may incorporate means, including sensors, trackers, recorders, and analyzers, for informing the operators and the developers about the time wasted in incidences.
- **Agile development**. 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.
- **Testability-oriented design**. 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. Utility-critical systems should incorporate means, including sensors and data analytics, for informing the operators and the developers about the time they could save.

3.3 Conclusions

A challenge for the 4th industrial revolution is to develop a methodology for cross-industry model-based integration. This study demonstrates that we can define universal rules, suggesting that this goal may be reached: principles of integration may be phrased as scenario-based rules, which may be transformed to protocols for risk detection, recognition, and identification. 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 in various domains. Validation of the models proposed here may be conducted by analysis of the system performance, obtained by activity trackers, using statistical metrics, followed by traditional usability testing in the corresponding scenarios.

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. 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*
4. Dekker, S 2007, Just culture: Balancing safety and accountability, Ashgate.
5. Harel, A 1999, Automatic Operation Logging and Usability Validation, *Proceedings of HCI International '99*, Munich, Germany, Vol. 1, pp. 1128-1133
6. Harel, A 2010, Whose Error is This? Standards for Preventing Use Errors, *The 16th Conference of Industrial and Management Engineering*, Tel-Aviv
7. Harel, A 2011, Comments on IEC 60601-1-8. Letter submitted to IEC/TC 62 working group
8. 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
9. Harel, A 2021, Towards model-based HSI engineering: a universal HSI model for utility optimization. *Accepted to the HSI conference, San Diego*.
10. Harel, A & Weiss, M 2011 Mitigating the Risks of Unexpected Events by Systems Engineering, *The Sixth Conference of INCOSE-IL*, Hertzelia, Israel.
11. Harel, A & Zonnenshain, A 2019, Engineering the HSI. *The first HSI conference,* Biarritz, France
12. Hollnagel, E 1983, Human Error. Position Paper for *NATO Conference on Human Error*. Bellagio, Italy.
13. 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.
14. 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
15. Landauer TK 1995, The Trouble with Computers, Usefulness, Usability, and Productivity. MIT Press
16. Norman, DA 1983, Design Rules Based on Analyses of Human Error. *Communications of the ACM 26(4)*:254-258
17. PlaneCrashInfo, 2014, *Causes of Fatal Accidents by Decade* http://planecrash-info.com/cause.htm

18. Shneiderman, B 1980, Software Psychology: Human Factors in Computer and Information Systems. Little, Brown
19. Singh, S 2015, *NHTSA CrashStat*, Critical Reasons for Crashes Investigated in the National Motor Vehicle Crash Causation Survey, DOT HS 812 115
20. Swatton, PJ 2011, "14.11", *Principles of Flight for Pilots*, Chichester, UK: Wiley & Sons Ltd
21. Taleb, NN 2007, The Black Swan: The Impact of the Highly Improbable. Random House Trade Paperbacks.
22. Weiler, M & Harel, A, 2011, Managing the Risks of Use Errors: The ITS Warning Systems Case Study
23. Weinberg, GM 1971, *The Psychology of Computer Programming*. Van Nostrand Reinhold.
24. Wickens, CD 1992, Engineering psychology and human performance (2nd ed.). Harper Collins Publishers.
25. Wiener, N 1948, Cybernetics; or, Control and communication in the animal and the machine. Technology Press, Cambridge.
26. Zonnenshain, A & Harel, A 2009, Task-oriented System Engineering, *INCOSE Annual International Symposium, Singapore*
27. Zonnenshain, A & Harel, A 2013, Resilience-oriented design. *The annual INCOSE-IL conference*, Hertzelia,
28. Zonnenshain, A & Harel, A 2015, A practical guide to assuring the system resilience to operational errors, *INCOSE Annual International Symposium, Seattle*