Towards Model-based HSI Engineering: A Universal HSI Model for Utility Optimization

Avi Harel
Ergolight
6 Givon Str.
Haifa 3433506 Israel
+972 54 453 4501
ergolight@gmail.com

ABSTRACT
This article targets designers of tools for developing utility-critical systems, including safety-critical, usability-critical, and productivity-critical systems, as well as consumer products. Analyses of operational failure case studies indicate that many of them are due to insufficient support for coping with operational complexity. Operational complexity may be defined in terms of exceptional situations. Prior studies indicate the feasibility of reducing the operational complexity by merging theories from various domains with practices employed in various industries. The article presents a semi-quantitative study of a universal HSI model, consisting of seven layers of generic mini models (GMM) used to cope with exceptions. The article assumes that all engineered systems are error-prone, and that even systems need to protect all system agents from making errors. The paper structure is adapted for people who are used to study presentation according to APA standards, with four sections: introduction, method, results, and discussion.

Keywords
Models, modeling, HSI, utility, risks, exceptions, resilience

INTRODUCTION

Early computer-embedded systems suffered from usability problems, resulting in productivity and safety loss (Landauer, 1995)[32]. Analyses of operational failure case studies indicate that often they are due to insufficient support for preventing and operating in exceptional situations (Zonnenshain & Harel, 2015)[56].

Foundations

Some of the concepts employed in this study, such as performance envelope, exceptions, and model-bases system integration, are built on the basic concepts. These concepts were described in (Harel & Zonnenshain, 2019)[23].

HSI2021, November 17-19, 2021, San Diego, CA, USA

System performance

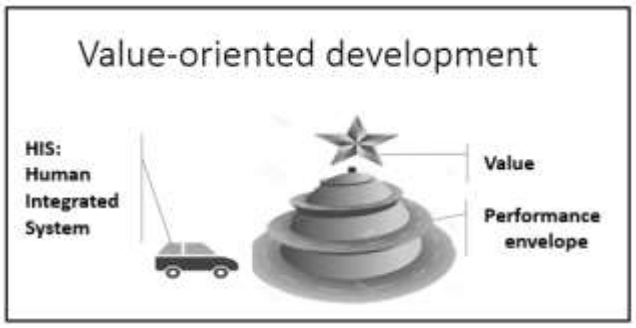
A common measure of the system value is in terms of performance. SEBoK refers to the System performance as a basic term. Typically, the meaning of this term depends on the purpose and functions of the system. Ideally, it is associated with metrics such as throughput, bandwidth, power consumption, etc. However, the perceived performance is typically industry and domain specific. In practice, it depends also on implicit factors, which are not testable. Often, the implicit factors are more significant than the measureable and testable factors. Therefore Cambridge dictionary defines performance as “how well a person, machine, etc. does a piece of work or an activity”. Typically, the term refers to the perceived efficiency, namely, how well the system performs.

The performance envelop

Hollnagel (2006)[26] 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.

The system value

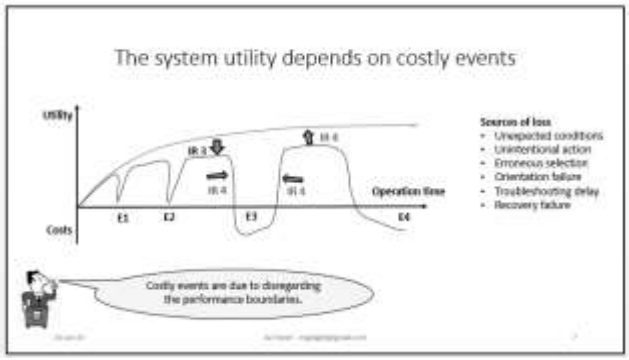
A common practice for assessing the system value is in terms of performance in routine operation in optimal conditions. For example, a typical goal of people as human beings is to stay healthy as long as possible. The problem with this kind of assessment is that the performance in optimal conditions does not represent properly the actual system value. In practice, as human beings who take care of their health, we take care of illness: prevent illness, and recover easily. The value of systems may be expressed in terms of the life-span performance bounded by the performance envelope. The value is affected not only by the design features, but also by the constraints, defined by the performance envelope, as illustrated in the following figure:



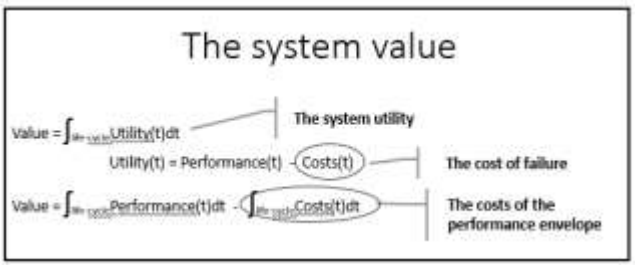
The actual system value is affected by the need to avoid the risks of crossing the performance envelope.

The effect of incidences

An incidence is an event of crossing the performance envelope. Costly incidences are those that cross the protection envelope. The effect of costly incidences on the system utility is depicted in the following figure:



Theoretically, the system value may be calculated by accumulating the utility over the life cycle, as in the following box:



The focus of MB HSI is on generic models enabling optimizing the system value, which is the life-cycle accumulated utility. The system utility is based on the performance, reduced by the operational costs.

Operational utility

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.

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 return on investing in exception management

Many developers are not aware of the risks involved in operating in exceptional situations. The means to avoid exceptional situations and to support exception management may be integrated into the model used to design the HSI. Models enable saving development costs by enforcing seamless adaptation to design changes.

Utility-critical systems should incorporate means, including sensors and data analytics, for informing the operators and the developers about the time they could save. The infrastructure for exception management may include special routines for saving the time that operators typically waste in handling exceptional situations.

Modeling

Scientific findings are documented in models, obtained in frameworks of meta-models of information behavior (Wilson, 1999)[54]. For example, Following Fuhs (2008)[13] description of hybrid vehicles, Boy (2012)[6] suggested modeling the system operation in the form of orchestrating human-centered design.

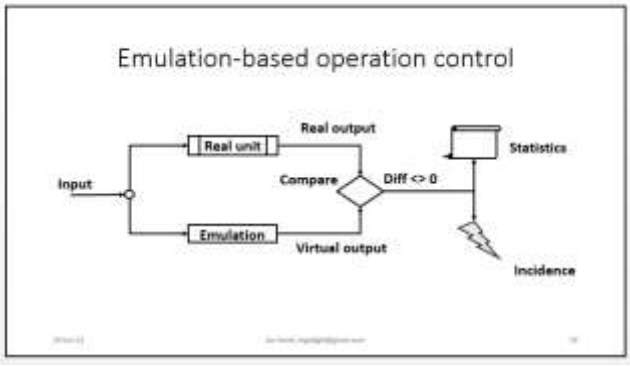
The goal of model-based engineering is to enable the development of more complicated systems than is possible with document-centric approaches.. Models enable participation by diverse SMEs in ways previously either not possible or less efficient. For example, Harel (1999) [16] demonstrated a model-based approach to usability testing, by capturing and analysis of instances of difficulties in using Windows applications. Also, Harel et al. (2008)[21] 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.

Digita twins

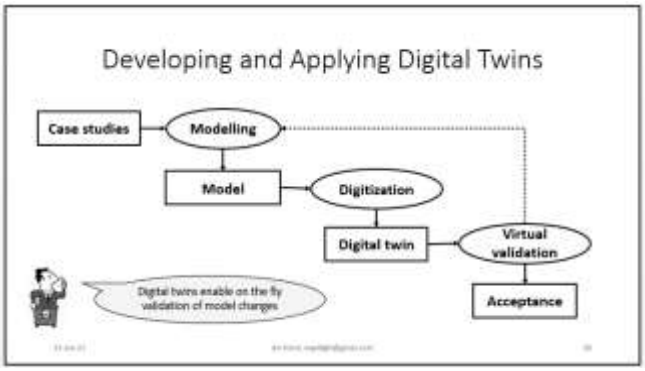
A definition of a digital twin adopted by SEBoK is "a high-fidelity model of the system which can be used to emulate the actual system". The concept of digital twins is based on the concept of virtual prototyping, dated in the 80s, in which a model was used to replace system units by emulation. This 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.

The article assumes that the digital twin is essential for controlling 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:



The article assumes that digital twins may be integrated in MBSE according to the following figure:



Therefore, MBSE is essential for seamless change validation based on a digital twins

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)[15]. .

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)[34]. 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.

Human factors

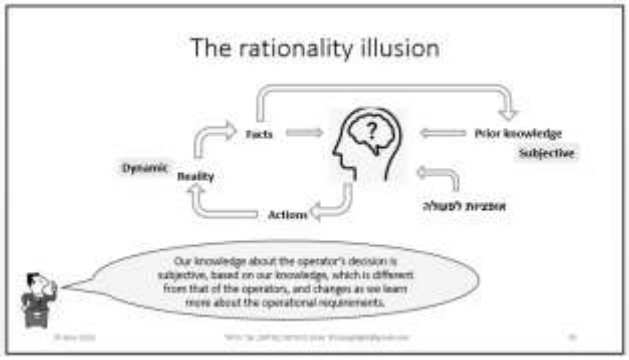
Human factors may apply to any human agent within or outside the system. However, in this study the focus on the effect of the operators on the system utility

Human performance

The system view of performance is based on the wrong assumption that the operators 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].

Rationality

Another topic is that of proper operation. This term implies that we expect the operators to be rational. The article assumes that rationality is vague. 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 as illustrated in the following figure:



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.

Errors

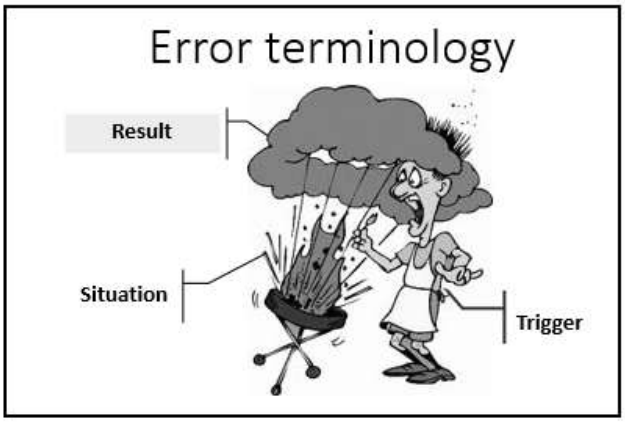
Errors are essential for learning (Wiener, 1948)[53]. However, errors are costly. Studies about the sources of accidents indicate that most of them are typically attributed to human errors, or improper usage. The factors mentioned by the reviewer are in the category of human errors. Human errors explain most accidents in the air (60%, PlaneCrashInfo 2014)[37] sea (80%, Baker & Seah 2004)[2], driving (90%, Singh 2015)[44], and in the industry (60-80%, Kariuki & Löwe 2006)[31]. Human errors are considered as the primary source of operational loss: by accidents, damage to property, low productivity, or user dissatisfaction (Landauer, 1995)[32]. Many of the usability issues in operating consumer products, such as home TV, are commonly attributed to use errors (Zonnenshain & Harel, 2009)[55]. 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.

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 egoless programming suggested by Weinberg, and proposed few principles for avoiding such design mistakes.

Norman (1983)[35] 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], Weiller & 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 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)[11].

It is not in the scope of the system design to enforce proper operation, as expected or specified by the developers. Following Hollnagel (1983)[25] 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)[19], as demonstrated in the following figure:



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)[20] 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.

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, during surgery, the surgeon may remove a tumor very carefully, but might miss the fact that the patient is bleeding from another part of the body.

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)[12]. For example, Harel (2006)[17] explained that alarm reliability and quality are critical for enforcing proper reaction to the alarms.

Usability

Believing that it is the designer's responsibility to reduce the costs of operation, Norman and Draper (1986)[36] explained that to avoid the loss, the system design should be user-centered. Shneiderman (1986)[43] proposed eight golden rules for user interface design. The quality of usability affects the operator's productivity, system safety, and the experience of using consumer products.

HSI essentials

The article assumes that HSI is a special case of system integration, in which we regard the human operators as a system unit, with special features, defined externally, by God. A primary challenge of system integration is the integration between engineered and human components. This section presents various definitions from various scientific and industrial sources used for modeling the HSI.

HSI engineering

HSI is a special focus of system integration. As such, HSI engineering should descend from systems engineering. Applying system thinking (Leveson, 2004)[33], HSI should focus on rare situations, and the HSI models should focus on operational rules (Harel & Zonnenshain, 2019)[23].

HSI reliability

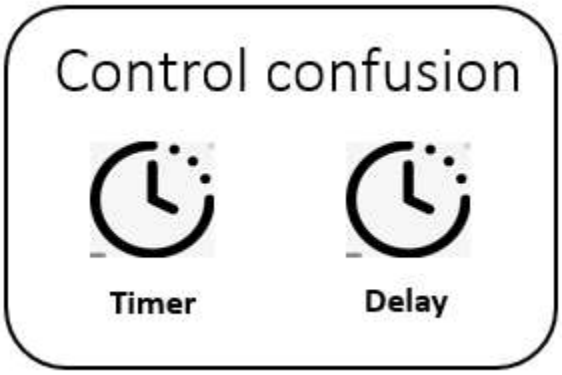
HSI reliability is a term used to describe an ideal virtue, of maximizing the system utility. HSI reliability is a descendent of operational reliability, which is a sibling of software reliability. Operational reliability may be defined as "The ability of an apparatus, machine, or system to consistently perform its intended or required function or mission, on demand and without degradation or failure" (Berard, 2013)[5].

The term HSI reliability refers to the system's capability to minimize the costs of operating in extreme conditions. 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)[20].

HSI complexity

A primary hurdle to HSI reliability is operational complexity. HSI 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:



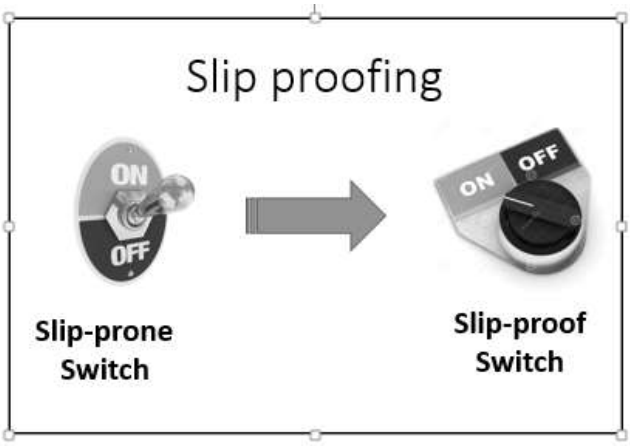
This type of complxity applies to many consumer systems, such as home appliances: Laundry, drier, air conditioner, furnace and oven. It also applied the many B-17 accidents in WW II. Control confusion may be resolved within the discipline of HCD. Often, control confusion may be resolved by redundancy 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)[43] 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 (2011)[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 HSI complexity in terms of the amount and variety of condition-dependent activities. HSI 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 HSI 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:



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

HSI hazard control

A hazard is a potential source of loss. Hazard control is used in industry to mitigate the risks of hazards. HSI 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]. HSI hazard control eliminates the risks of exceptional events and of operating in exceptional situations.

Prior studies

The prior art is an ongoing study on protecting systems from operational surprise, with focus on human errors. Most of it is documented in an article by Zonnenshain & Harel (2015)[56], which was revised by Harel & Zonnenshain (2019)[23].

Model-based HSI

Model-based HSI is part of model-based engineering, focusing on the integration between the system and its operators.

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. 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].

HSI modeling

Hollnagel (2006)[26] 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. Most failures are often attributed to latent defects, wear-out, unexpected environmental conditions, and improper usage (both accidental and malicious) 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].

Learning from incidences

Incidence modeling may be based on four types of evidence: anecdotal, statistical, causal, and expert evidence (Hornikx, 2018)[27]. By its nature, anecdotal evidence is subject to systematic deviation from the norm and/or rationality in judgment (Haselton et al. 2005)[24]. 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 o 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.

Data analytics

Tracking tools enable capturing and measuring the costs of daily, low-cost events (Harel, 1999)[16]. Harel et al. (2008)[21] demonstrated a way to apply data analytics in automated usability testing, and Harel (2009)[18] demonstrated that data analytics may be used to identify problem indicators. Universal tracking is crucial also for enabling learning from rare events.

HSI meta language

Jacobson (1987)[30] 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.

HSI scenarios

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

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.

HSI 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)[33] 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. HSI modeling focuses on universal methodologies of rule-based design, for the sake of reducing the HSI complexity. HSI constraints are operational rules constraining the HSI (Harel & Zonneshain, 2019)[23]. Typically, these constraints are scenario-dependent.

HSI exceptions

An exception is a situation beyond the performance envelope. HSI exception extends the concept of software exceptions, introduced in LISP **(**Gabriel & Steele, 2008)[14]. 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, HSI 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.

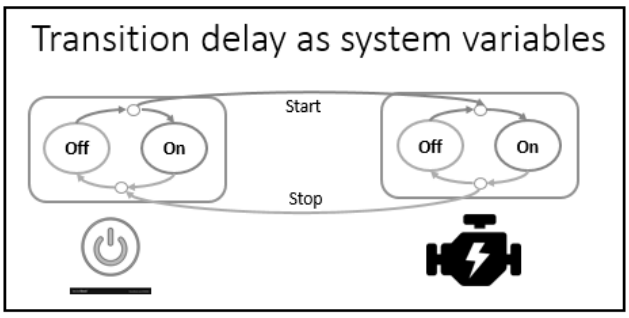
HSI resilience

According to SEBoK, resilience is the ability to maintain capability in the face of adversity. Jackson and Ferris (2013)[29] presented principles for assessing and improving the resilience of engineered systems across their life cycle. HSI resilience is about HSI factors in resilience assurance (Zonnenshain & Harel, 2015)[56].

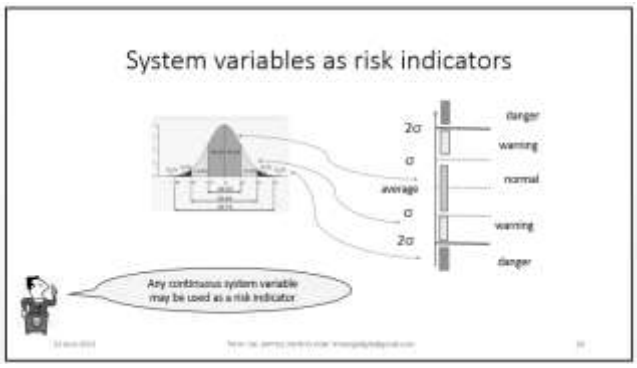
For example, we may explore various collaboration options in a minimal system, consisting of a simple engine with two states: On and Off, operated by a switch with states: On and Off. The functional option is complicated when the operator is required to support early detection and identification of malfunction. 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.



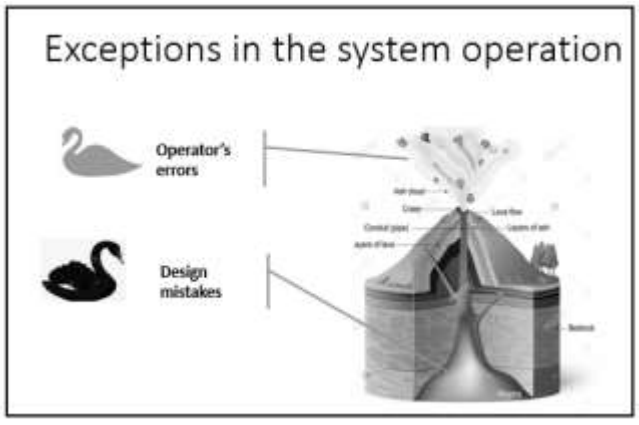
The system may record the transition delays and generate distribution functions for these delays. The system design may make use of the distribution parameters, and include identification of extreme values, as well as extra means for alarming and emergency shut-down. The following figure illustrates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



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.

HSI accountability

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:



Objective of the present study

The puzzle of HSI consists of various disciplines, such as:
Task-oriented disciplines:

- SCD – System Centered Design
- SoS – System of Systems
- HITL – Humans In The Loop
- HCD – Human Centered Design
- UID – User Interface Design
- HCI – Human Computer Interaction
- TDD – Task Driven Design

Risk-oriented disciplines:

- HF – Human Factors
- Cognitive Engineering
- Resilience Engineering

This study focuses on the a discipline not elaborated yet, which is Activity-oriented Design (AoD)

The existing knowledge assumes that the system design should focus on performance, and assign low priority to failure prevention. MBSE is based on project-specific models. The practices for preventing failure and for safe-mode operation are mostly undocumented industry and domain specific. The Black Swan theory about learning from rare events implies that we cannot foresee the surprise, and the common design practices do not include means to capture incidences.

The article proposes extending the existing knowledge:

- A way to capture incidences
- A way to develop models of the incidences captured
- A general, seven-layer model of system operation
- Sets of general mini-models (GMM) enabling to prevent errors.

This methodology has been developed gradually since 2008. Parts of it were presented in the INCOSE 2015 conference and in the HSI2019 conference.

The primary objective of this study was to develop a prototype that demonstrates the feasibility of a universal HSI model, which may be applied and customized for the design of utility-critical systems.

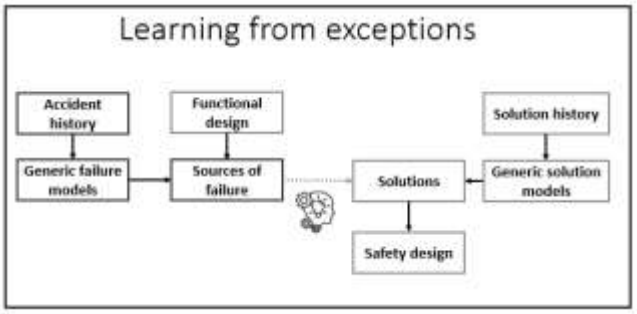
METHOD

Method overview

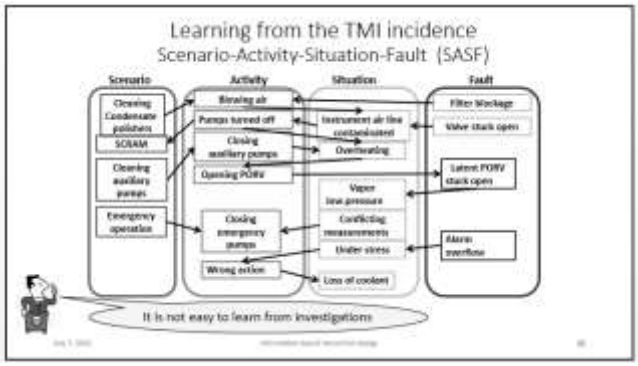
The article presents a model of HSI focusing on the utility. It shows the feasibility of modelling the system operation, highlighting key critical failure modes. Also, the article presents a method used for developing this model, of learning from rare events.

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)[28]. 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:

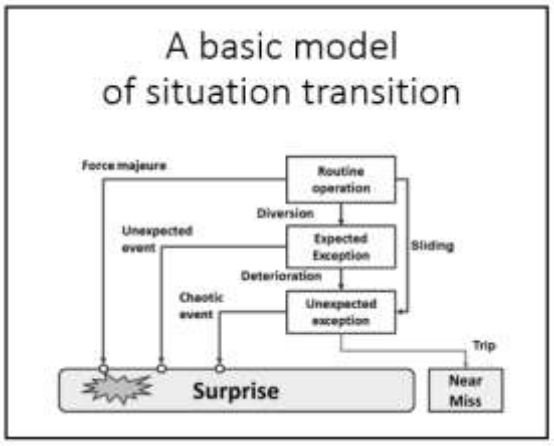


The first stage in the methodology generation is to create a bank of generic failure modes, based on failure analysis. The following figure illustrates a way to document failure analysis when applied to the TMI accident:



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:

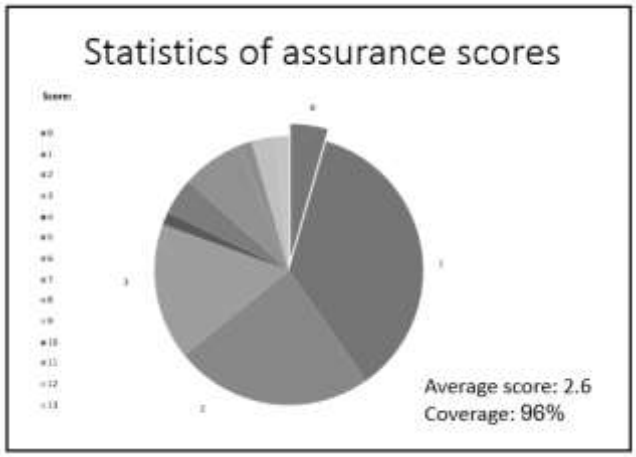


Then we established a dedicated working group on system resilience, in which we examined various possibility of failure modes found in 67 incidences. The outcome of this examination was a comprehensive model of system resilience. The model was reported in 2015 INCOSE conference (Zonnenshain & Harel,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

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:



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 developed in two stages:

Defining the GMMs

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.

Sampling

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.

Model development

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

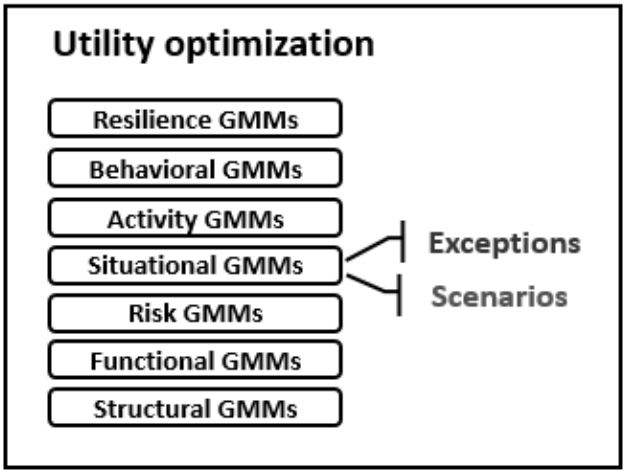
- **Behavior abstraction**. This activity is the outcome of the incidence analysis. The goal of behavior abstraction is to transform domain-specific terms into universal, cross-domain terms. The abstract version of a specific incidence is an incidence model.
- **Model matching**. The objective of model 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.

Defining the structure of the universal HSI model

The structure definition was based on analysis of the relationships between various entities that play ops in the system operation: functions, units, risks, states, events, reaction, resilience.

RESULTS

This section presents a prototype of a universal HSI model developed in this study, consisting of seven layers of GMMs; structural, operational risks, functional, static, dynamic behavioral, and resilience models.



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 model describes a system recursively, in terms of 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 subsystem is linked strongly with the

operators, and loosely with the users. The focus here is on the operators, which may typically affect the activity more than the other human participants. Each of the following models may apply to each subsystem defined by this recursion.

Entities of the structure layer

Common assumptions about the machine and the operator's roles are:

- **The machine's role**. The machine needs to trace situation changes, and to inform about them to the operators. The information should include an indication of risky situations and critical events, and instructions for mitigating the risks.
- **The operator's role**. The operators need to select an optional response according to the operational scenario.
- **The HSI**. The integration's role is to constrain the activity to the design scope, and to prioritize the operational options according to risk considerations.

A functional layer

A functional model describes the operational context and features of the human-machine interface (HMI) required to accomplishing an operator's task. These include activating the system functions and maximizing the performance, as well as practices for risk reduction.

Entities of the functional layer

In MBSE, the following entities are part of the data structure:

- **Tasks.** These are the operators' view of the system functions
- **System state**. A value of a discrete state variable such as power on/off.
- **Unit state**. The sensory or virtual operational state of subunits, such as enabled/disabled.
- **Scenario**. Any system state significant to the operators' view of the operational situation. Examples are operational modes (routine, testing, maintenance…)
- **System variables**. A system variable may be any performance variable, any risk variable, or any state of any system component.
- **Performance variables**. Measures of the system performance, such as vehicle speed

Functional rules

These include the following types of rules:

- **Universal rules**. These apply to all human-operated systems, such as about the matching between the operator's intention, represented by scenarios, and the values and changes of system variables.
- **Interaction rules**. These apply to the human-machine coordination.
- **Industry-specific rules**, such as those derived from physical laws.

Common design practices

The risk is about the operators not noticing the exception. The design principle is that the system should inform the operators about exceptions and provide explanations about the risks of optional possible responses.

A risk layer

This model describes various ways in which the operation might fail. These risks are often attributed to operator errors (Zonnenshain & Harel, 2015)[56].

Entities of the risk layer

The risks involved in the system operation include:

- **Omission** of a necessary action
- **Mistake.** Selecting an improper control
- **Operator's slip**. Unintentional activating the wrong control
- **Mode error**. Error in selecting a control while assuming the wrong conditions
- **Risk variable**. Measure of the operational risks, such as operational temperature
- **Safety limits**. High and low thresholds of safe values of system variables
- **Normal limits**. High and low thresholds of normal values of system variables

Common design practices

Protection features including:

- **Risk preview**. Display potential risks based on trend analysis. Prediction by statistical analysis of measurements of system variables
- **Scenario-based interaction.** Avoiding critical errors in an emergency, due to the limited attention capacity of the human operators
- **Unconditional action selection**. Seamless operation obtained by enforcing direct mapping from intention to action through scenarios, enabled by the consistent effect of control activation.

A static layer

A static model is a 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.

Entities of the static layer

Normal behavior should be defined in the operational specification. Abnormal behavior should be defined by instances of diversion from the normal behavior.

In MBSE, the following entities are part of the data structure:

- **Situations**. The system situation may be represented by system variables. A situation is defined as a set of values of a selection of the system variables.
- **Normal situations**. A situation is normal if it complies with any of the rules describing normal situations. For example, a simple system consisting of a device and a power switch may have two rules

defining normal situations: Operative scenario, when the switch is On and the device is operating, and Non-operative scenario, when the switch is Off and the device is not operating.

- **Exceptional situations**. Any situation that does not comply with any of the rules defining normal situations is exceptional. In the example of the simple system above, the exception may be **expected**, when the switch is On and the device is not operating, or **unexpected**, when the switch is Off and the device is operating.
- **Risk indicators**. The values of continuous performance or risk variable may be compressed to five states: normal, risky low, risky high, forbidden low, and forbidden high.
- **Static design scope**. The situations (system states) that the design supports.

Abnormal situations are defined as the complement of normal situation: a situation is classified as abnormal if it is not a member of the set of normal situations, in the context of the active scenario. Similarly, an event is classified as abnormal if it is not a member of the optional events specified for the active operational stage (such as procedure step).

Situational rules

These rules enable the handling of exceptional situations.

- **Basic rules**. Inter-state constraints, inter-unit coordination, scenario-unit coordination, expected exceptions, enable the identification of expected exceptions.
- **Derived rules**. Enable detection of unexpected exceptions, excluded from the basic rules

Common design practices

Common practices are based on integrating human factors in process hazard analysis (PHA), commonly classified as hazard and risk assessment techniques, such as FTA, FMEA, and HAZOP (Baybutt, 2002)[4].

A dynamic layer

A dynamic model is a representation of the operational activities. 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 (Harel & Weiss, 2011)[22].

Entities of the dynamic layer

In MBSE, the following entities are part of the data structure:

- **Events**. An event is an instance of a situation change.
- **Normal events.** An event is normal if it complies with any of the rules describing that it is normal before the situation change. For example, the simple system above may have two rules defining normal activity: Turning the switch On when the situation is the Operative scenario, and turning the switch Off when the situation is the Non-operative scenario.
- **Expected diversion.** This applies to any event complying with a rule about transitions from normal to an exceptional situation. An example from the simple system above is the device being stopped due to malfunction.
- **Dynamic design scope**. These are the system events that the design supports, including normal events and expected diversion.
- **Unexpected diversion.** This is an event of transition from normal to an exceptional situation that does not comply with any of the rules. In the example of the simple system above, the event of turning the switch On while in the Operative scenario, or turning the switch Off while in the Non-operative scenario, are exceptional.
- **Slip.** This term refers to unintended diversion.
- **Surprise / Diffusion / chaotic event**. Any events resulting in a transition to a situation out of the dynamic design scope.

Activity rules

These are protocols governing

- **Operational procedures.** Defining operational procedures and constrained control.
- **Exception management.** Controlling the transition to exceptional situations

Common design practices

The risks are about the operators not noticing the diversion. The design principle is that the system should warn the operators and enable graceful rebounding from unintended events.

A behavioral layer

A behavioral model is a representation of the responses to events. The risks are that the operators might fail to respond properly to the rebound message, or that they do not understand the risks associated with the alarms, or they fail with the troubleshooting. The design goal is to mitigate the risks of wrong responses to events. The core of the behavioral model is an abstraction of typical system responses to exceptional events, with a focus on risk reduction.

Entities of the behavioral layer

In MBSE, the following entities are part of the data structure:

- **Rebounding**. Automated rejection of a risky event, such as an operator's slip.
- **Hazard**. An event with evidence about risk, such as resulting in an exception.
- **Trip**. An automated interruption in the system operation, aiming to avoid failure.
- **Alarm**. An automated alerting message to the operators about a hazard.
- **Troubleshooting**. An indication of a possible source for an alarming situation.

Behavioral rules

Rules applicable to responding to events include:

- **Scenario compliance**. The automated setting of states assigned with the scenarios.
- **Alarm**. The sound attributes should signify the risk level.
- **Protocols**. The response to exceptional events may be phrased as protocols for risk detection, recognition, and identification.

Common design practices

The following practices are commonly employed to mitigate the risks are about the unexpected response to events when in an exceptional situation:

- **Rebound**. The rebounding from an operator's slip is by enforcing the operators' understanding of the rebound message.
- **Trip**. The information in the trip message is displayed gradually, supporting both common and rare trips.
- **Troubleshooting**. Based on information from sensors, enabling seamless identification of the source for the alarm.

A resilience layer

A resilience model is a representation 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.

Secondary risks

Rule-based design may facilitate the interaction and coordination in normal operation, but might also hamper the control in an emergency. Weak constraints may facilitate emergency control, but the result of operating in exceptional situations might be unexpected.

Entities of the resilience layer

In MBSE, the following entities are part of the data structure:

- **Safety feature.** A feature intended to secure the system's safety.
- **Secondary risk**. Risk of potential failure of a safety feature.
- **Safety backup**. A safety feature intended to use as a backup in case of a secondary failure.

Resilience rules

Rules applicable to resilience include:

- **Situational rules**. Constraining the availability of safety features
- **Activity rules**. Protocols governing the transitions to exceptional states of safety features.

Common design practices

The following practices are commonly employed:

- **Alerting**. Alarming when the backup feature stops being available in functional scenarios
- **Reminder**. Continuous notification on unavailable backup while in functional scenarios
- **Rebounding**. Prevent going functional when the backup feature is not available.

DISCUSSION

Engineering

As discussed by Harel & Zonnenshain (2019)[23] the engineering of HSI is based on defining operational rules, which define exceptions by exclusion from normal behavior.

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].

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.

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.

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.

Model 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 (Redmill, 2000)[38].

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.

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.

Conclusions

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 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 operational rules. An example of a GMM is a set of rules for disabling maintenance activity during functional operation by assigning such activities to maintenance scenarios.

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. However, this is not in the scope of this article.

ACKNOWLEDGMENTS

I thank Avigdor Zonnenshain, Uzi Orion, Moshe Weiller, 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. Endsley, MR 1995, Toward a theory of situation awareness in dynamic systems. *Human Factors. 37 (1)*: 32–64. doi:10.1518/001872095779049543
13. Fuhs, A 2008. Hybrid vehicles: and the future of personal transportation. *CRC press*.
14. Gabriel, RP & Steele, GL 2008, A Pattern of Language Evolution. *LISP50: Celebrating the 50th Anniversary of Lisp*. pp. 1–10.
15. Grimm, T 1998, The Human Condition: A Justification for Rapid Prototyping. *Time Compression Technologies, vol. 3 no. 3*. Accelerated Technologies, Inc. May 1998
16. Harel, A 1999, Automatic Operation Logging and Usability Validation, *Proceedings of HCI International '99*, Munich, Germany, Vol. 1, pp. 1128-1133
17. Harel, A 2006, Alarm Reliability, *User Experience Magazine*, Vol 5., Issue 3.
18. Harel, A 2009, Statistical Analysis of the User Experience, Invited talk - *2nd Meeting of isENBIS*, Hertzelia, Israel
19. Harel, A 2010, Whose Error is This? Standards for Preventing Use Errors, *The 16th Conference of Industrial and Management Engineering*, Tel-Aviv
20. Harel, A 2011, Comments on IEC 60601-1-8. *Letter submitted to IEC/TC 62 working group*.
21. 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.

22. Harel, A & Weiss, M 2011 Mitigating the Risks of Unexpected Events by Systems Engineering, *The Sixth Conference of INCOSE-IL*, Hertzelia, Israel.
23. Harel, A & Zonnenshain, A 2019, The first HSI conference, Biarritz, France
24. 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.
25. Hollnagel, E 1983, Human Error. Position Paper for *NATO Conference on Human Error*. Bellagio, Italy.
26. 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.
27. Hornikx, J 2018, Combining Anecdotal and Statistical Evidence in Real-Life Discourse: Comprehension and Persuasiveness. *Discourse Processes Vol 55*, Issue 3.
28. 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
29. Jackson, S & Ferris, T 2013, Resilience Principles for Engineered System. *Systems Engineering, 16(2),* 152-164. doi:10.1002/sys.21228.
30. Jacobson, I 1987, Object-oriented development in an industrial environment. *ACM SIGPLAN Notices*. 22 (12): 183–191.
31. 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
32. Landauer TK 1995, *The Trouble with Computers, Usefulness, Usability, and Productivity*. MIT Press
33. Leveson, N 2004, A New Accident Model for Engineering Safer Systems. *Safety Science 42(4)*:237-270
34. Luqi 1989, Software Evolution through Rapid Prototyping. *IEEE Computer. 22 (5):* 13–25. doi:10.1109/2.27953. hdl:10945/43610
35. Norman, DA 1983, Design Rules Based on Analyses of Human Error. *Communications of the ACM 26(4)*:254-258
36. Norman, DA and Draper, S 1986, *User Centered System Design: New Perspectives on Human-Computer Interaction* Lawrence Erlbaum Associates.
37. PlaneCrashInfo, 2014, *Causes of Fatal Accidents by Decade* http://planecrashinfo.com/cause.htm
38. Redmill, F 2000, Understanding the use, misuse and abuse of safety integrity levels, *Proceedings of the Eighth Safety-critical Systems*
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. Weiller, 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. Zonnenshain, A & Harel, A 2009, Task-oriented System Engineering, *INCOSE Annual International Symposium, Singapore*
56. Zonnenshain, A & Harel, A 2015, A practical guide to assuring the system resilience to operational errors, *INCOSE Annual International Symposium, Seattle.*