Chapter 15

System Thinking Begins with Human Factors: Challenges for the 4th Industrial Revolution

Avi Harel

Synopsis

The 3rd industrial revolution (IR) brought us two new disciplines critical for system design: systems engineering (SE) and human factors (HF). In the 3rd IR, the two disciplines did not integrate very well, because both disciplines were system-centric. This is changing in the 4th IR, with the emergence of system thinking. System thinking has two aspects. The *internal* aspect is about the collaboration between system components, and the *contextual* aspect is about the interaction with the real world, namely, the customers and stakeholders, as well as the operational constraints. System thinking is a two-stage process, beginning with the contextual aspect, followed by the internal aspect. The role of HF is in the domain of contextual system thinking, namely, of integrating people in complex systems. The framework recently established for relating the systems to the real world is human-machine interaction (HMI). This new transdisciplinary framework enables us to bridge the chasm between the two disciplines. A model of HMI design proposed here regards two distinct views of the user, corresponding to the two aspects of system thinking: as a system operator, the user corresponds to the contextual aspect, and as a system component, the user corresponds to the internal aspect. As a system operator, in contextual system thinking, we are concerned about the HMI. As a system component, in internal system thinking, we are concerned about the human capabilities, as well as about safety issues. In the 4th IR, we need to reengineer the HF, in order to integrate it better with the SE. HMI engineering (HMIE) is a new discipline, intended to implement ideas and tenets of HMI, for the sake of enabling effective system thinking. A new model of human machine collaboration (HMC) enables representing formal definition of contextual information, including the operational scenario, and the operator's goal. The model enables representing critical design dilemma, such as automation control and task management, in normal and exceptional operational conditions. New architectures, based on the new model, may result in better gain, in terms of productivity and safety.

15.1 Introduction

A case study

A few years ago, I visited a friend of mine at his home in Toronto. This guy boasted to me about a new system that he bought, enabling him to control his house using several small, mobile devices. He could control the lights, the TV systems, the music systems, the windows, etc., all by these small devices. When he finished describing to me the features he had there, his wife mentioned him politely that he was the only person in the house who could actually operate the system. She commented that she did not dare to touch the remote controls, because often, when she tried to close the lights in her bedroom, she ended up activating the TV or the stereo system in the living room, which was unpleasant and embarrassing.

The legacy of the 3rd industrial revolution

Traditionally, systems engineers and software engineers disregard the ways people might operate the system. Typically, they believe that their intuition is good enough to design operational procedures that the operators may follow easily and reliably. Typically, they are not aware of the risks of letting the operators deviate from the intended operational procedures. Often, they overlook the possibility of choosing the wrong option from among the various features incorporated in the interface, as was the case of the computerized house in Toronto, described earlier. They might overlook the fact that more may be less. If the operators have many options to choose from, the might not find the one they need. Typically, they do not bother to document the operational procedures, and to prevent diversions from the procedures by design. They are not ordinarily aware of the special education required for ensuring that the interaction is efficient and reliable.

The challenge of the 4th industrial revolution

The 4th industrial revolution (IR) is about a shift in our view of the effect of technology on our experience of using systems. In his keynote article on the significance of the fourth industrial revolution, Klaus Schwab (2016) sets the goal:

> "All of these new technologies are first and foremost tools made by people for people".

He then explains that,

> "[The] inexorable shift from simple digitization (the Third Industrial Revolution) to innovation based on combinations of technologies (the Fourth Industrial Revolution) is forcing companies to reexamine the way they do business."

Kenett et al. (2018) discuss different aspects of systems engineering in the context of the 4th IR, and concluded that in the near future:

- Virtually all systems will have porous and ill-defined boundaries
- Virtually all systems will have ill-defined requirements which are changed frequently

According to Cooper (1998), technology consists largely of pieces that work, but not at all well. This is often the fault of poorly designed user-interfaces. Too many devices ask too much of their users. Too many systems make their users feel stupid when they cannot get the job done. In the 4th IR, everything we regularly use in our home, work, transportation, is being equipped with new technology. Boy and Narkevicius, (2013) observed that "requirements, solutions and the world constantly evolve, and are very difficult to keep current." Can we use systems of the 4th IR safely?

Putting people first

A primary challenge of system design in the 4th IR is about what people experience in going through this change. In the words of Schwab: "In the end, it all comes down to people and values. We need to shape a future that works for all of us by putting people first and empowering them and constantly reminding ourselves that all of these new technologies are first and foremost tools made by people for people."

Crossing the human boundaries

In the first three industrial revolutions, the boundaries between technology and people were clear. Technology was a means to design products and systems. The effect of technology on the human behavior was indirect, through these deliverables. The new technologies imply changes in the ways people interact with the systems. In the 4th IR, technology is gradually crossing the boundaries, affecting directly the body and soul of people.

15.2 Systems

Definitions of systems

According to ISO/IEC/IEEE 2015, following Bertalanffy (1968), the term "system" is a short name for "system-of-interest." The general term 'system' does not require any purpose or interest, and may refer to natural systems as well. According to the INCOSE System Engineering Body of Knowledge (SEBoK), the interest of systems engineering is in engineered systems. According to Bartolomei el al., (2006) an engineering system is

"An engineering system is a complex socio-technical system that is designed, developed, and actively managed by humans in order to deliver value to stakeholders."

The SEBoK definition of '*socio-technical system*' is *"an engineered system which includes a combination of technical and human or natural elements."*

In this chapter, the system is a socio-technical system, which means that it includes technical and human elements. The technical elements are called here "functional units." The human elements may be insiders, namely, operators, or outsiders, namely, stakeholders, such as users.

System thinking

Systems thinking is widely believed to be critical in handling the complexity facing the world in the coming decades. Richmond (1994), the originator of the systems thinking term, defines systems thinking as "the art and science of making reliable inferences about behavior by developing an increasingly deep understanding of underlying structure." With systems thinking, a systems engineer "can see both the forest and the trees; one eye on each." Senge (1990) defines systems thinking as a discipline for seeing wholes and a framework for seeing interrelationships rather than things, for seeing patterns of change rather than static snapshots. Sweeney and Sterman (2000) noticed that systems thinking involves the ability to represent and assess dynamic complexity (e.g., behavior that arises from the interaction of a system's agents over time), both textually and graphically.

Stave and Hopper (2007) observed that the term systems thinking is used in a variety of sometimes conflicting ways. Kopainsky, Alessi, and Davidsen (2011) assert that systems thinking should include appreciation for long term planning, feedback loops, non-linear relationships between variables, and collaborative planning across areas of an organization.

Arnold and Wade (2015) compared the various definitions and came out with the definition that

> *"Systems thinking is a set of synergistic analytic skills used to improve the capability of identifying and understanding systems, predicting their behaviors, and devising modifications to them in order to produce desired effects. These skills work together as a system."*

The authors conclude that "the use of systems thinking transcends many disciplines, supporting and connecting them in unintuitive but highly impactful ways."

Layers of system thinking

Following the discussion above, we may consider two aspects of system thinking:

- The internal aspect is about the functional units integrated with the operators, collaboration between components of the engineered system, and

- The contextual aspect is about the interaction of the engineered system with the real world, namely, the customers and stakeholders, as well as the operational constraints.

Figure 1 illustrated the two layers of system thinking:

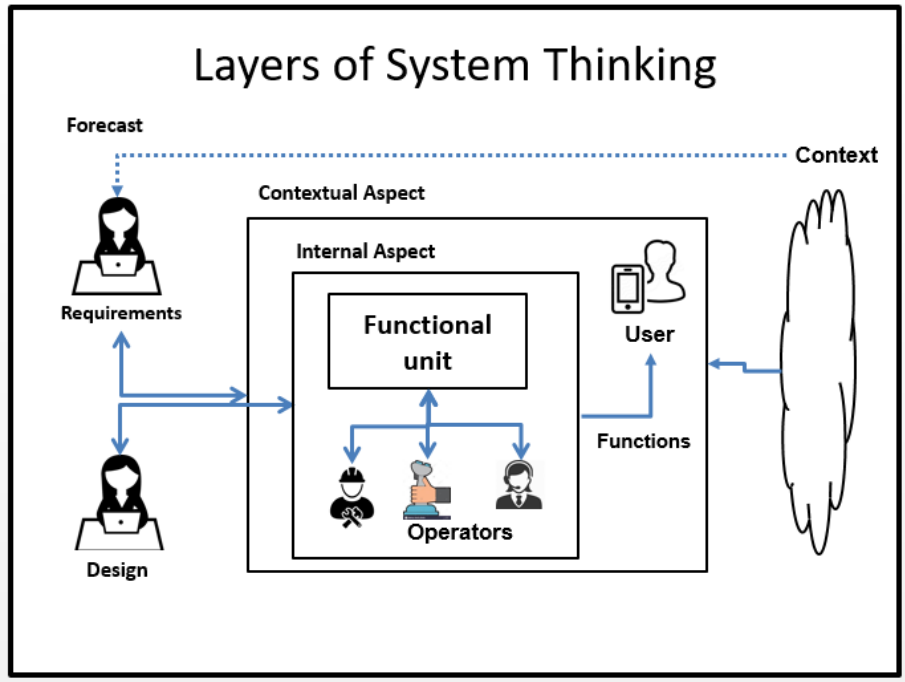

Figure 1 – Layers of System Thinking

According to this model, the contextual aspect is defined based on requirements specifications, with respect to the user's tasks and capability, and considering forecast of the context. Also, the internal aspect is defined design considerations about the various roles of the operators, and their collaboration with the functional units.

This chapter is about ways people may integrate with systems, about necessary changes in the way engineers may implement the concept of system thinking, and about engineering activities enabling successful integration with its operators.

Agile system thinking

In the early days of the 3rd IR, system development followed the waterfall model. According to this model, the system design is based on the requirement specifications, which remained unchanged until the version release. This model did not work very well, because during the system development new requirements emerge. Therefore, the waterfall model was replaced by other models, such as iterative development or agile development, which facilitated changing the requirements during the system development.

System thinking is a continuous process, integrated with agile development. The contextual aspect is the one that senses the need to change the requirements, and triggers the change. The internal aspect is the traditional response to changes, typical of agile development.

15.3 Human factors

Traditional interaction definition

Traditionally, the engineers who define the interaction with the operators are systems engineers or software engineers. Typically, they are technology-oriented, which means that they try their best to integrate state-of-the-art technological feature. Often, they are feature-oriented, which means that they include in the design as many features as the technology allows them to include, regardless of whether or how the operators will use them. Also, often, they are designer-centric, which means that optimize the interaction according to their knowledge about the operational procedures, and their own preferences.

In the 3rd IR, we were concerned of the effect of the human-machine interaction on functional attribute: performance, production etc. In the 4th IR we are also concerned about subjective attributes, such as customer satisfaction and operator's experience of making the system work as intended.

Technology-driven interaction design

Technology-oriented engineers love to offer as many features as they can, and they try to assist the operators in as many ways as they can. In the early days of computing, when people just started to write software on the first mainframe computers, Weinberg (1971) demonstrated and analyzed the trouble with technology-driven software programming. Technology developed much since 1971, but the troubles remain the same.

Typically, software engineers focus on providing user interfaces that support the features specified in the requirement document. Often, they do not think how the operators will access these features, or if they use them correctly, in the proper situations (Cooper, 1998). They feel responsible for preventing bugs, but not for preventing operator's errors (Harel, 2010). Infrequently used units may contain bugs that go unnoticed in normal operation. These bugs might hamper the interaction in exceptional conditions.

Standish (1995) analyzed reasons for the failure of software projects. The conclusion of this analysis was that there is a huge gap between software development practices and engineering disciplines. They demonstrated their finding by comparing the attitude to failure of software projects to those of building bridges. They concluded that beside 3,000 years of experience, there is another difference between software failures and bridge failures. "When a bridge falls down, it is investigated and a report is written on the cause of the failure. This is not so in the computer industry where failures are covered up, ignored, and/or rationalized. As a result, we keep making the same mistakes over and over again" (Standish, 1995). Figure 2 illustrates the effect of software bugs on the user experience.

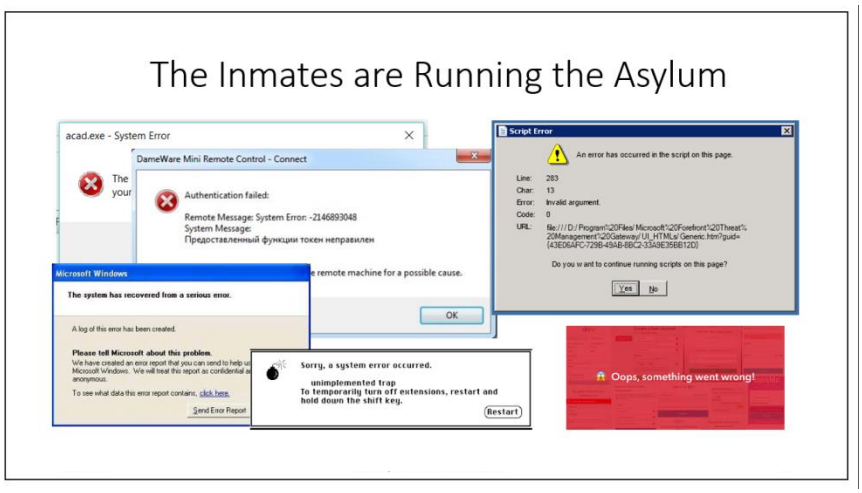

Figure 2 – The Inmates are Running the Asylum

This kind of messages is helpful for the software programmers in the debugging, but is useless, and quite confusion, for the users. Interaction design using the framework of HMI enables solving this problem. In the 4th IR, new failure models are available to the system designers, enabling then to handle also the unexpected. Interaction designers employ interaction protocols to ensure that the operators can perceive and handle rare events, and recognize and understand unexpected situation.

Feature-oriented design and testing

Traditionally, interaction design is feature-oriented, which means that the by design the system offers as many features as the technology allows, regardless of whether the operators will need them, how they find them, or if they add noise to the operators' perception of the controls that they need to activate.

An example of the trouble caused by unnecessary features is nuisance of beeps that many home appliances generate, just for the fun of the designer. Another example is of the delay option in home appliances. A user of an air conditioner or a drier that has this feature might confuse the delay control with other time-related controls, and activate it unintentionally. Then, the appliance would not work, because it is in delay mode. A user who just tried the basic features might not understand the meaning of the notifications on the panel or the remote control, and might not understand the meaning of the warning sound. The user might call a technician to see the appliance does not work, but when the technician arrives, the delay time is already over, and the appliance works perfectly. Somebody has to pay for the visit of the technician, which was in vain.

Designer-centric interaction design

Software developers often optimize the user interface to suit their own needs, namely, to facilitate the software development. For example, during the software debugging, the developers need to repeat the same action over and over again, each time testing the effect and outcome of a code or parameter value change. The activation of those actions which are tested frequently is tedious, because the access to the actuator may be through a sequence menu selections and searching. To facilitate such procedures, the

software developers often define shortcut keys, such as those of Gmail, presented in the following figure:

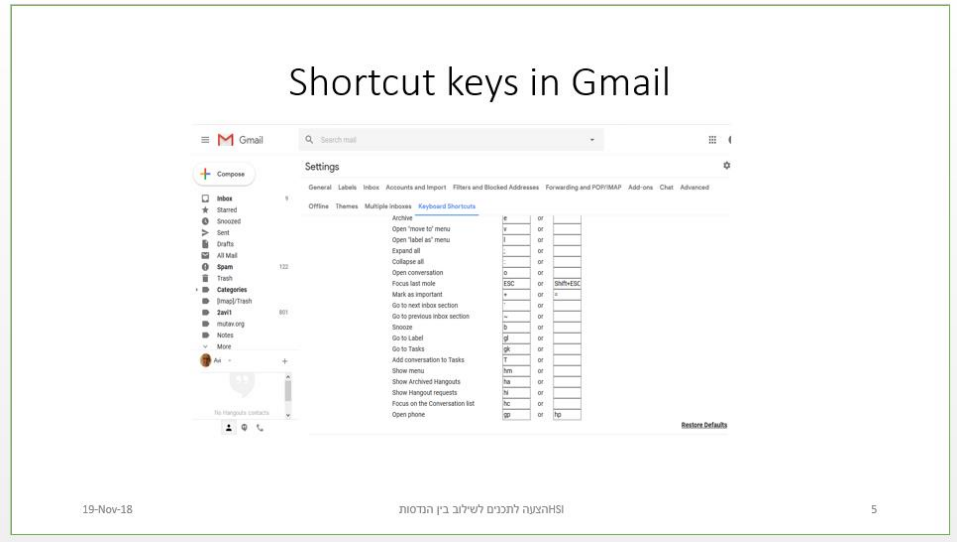

Figure 3 – Shortcut keys in Gmail

All the shortcuts in this figure were defined to facilitate the debugging of Gmail. Few, sophisticated users, may generate their own shortcuts. These shortcuts are sometimes problematic, when they are activated accidentally. Typically, the operator is not aware of the accidental activation of the shortcut key, and is confused by the unexpected change in the system situation.

The usability shift

In the beginning of the 3rd IR, system engineers were not aware of the role of human factors, and they did not consider it in the system design. The design was technology-oriented. People judged the value of systems in terms of functions. We did not care much about the operators. We thought about the ways the operators interact with the system, but not about the effect of the interaction on the other operator's tasks. For example, we assumed that professional practitioners might be willing to spend time in order to learn how to use our system.

By and by, following investigation of many accidents, systems engineers realized that they need to consider human factors in the design. By the end of the 3rd IR, more and more HF practitioners became part of the design team. Their charter is to look at the ways the operators use the system, and how human errors contribute to accidents and to performance reduction. Still, human factors are of lower priority. The functions were evaluated primarily in terms of performance, productivity and safety. Typically, the way to consider them is in two stages: first, the system is designed with the criteria of maximizing performance. Then, in a later stage, we review of the system design, looking for design flaws, hampering the system operation.

From the discussion above we can conclude that HMI based on traditional practices might not be sufficiently good for the 4th IR.

The single most important factor of productivity assurance is employing usability assurance methodologies. The need to incorporate human factors in system design became obvious early when people started to use computers for designing office applications (Landauer, 1996). According to a Gartner report (1995), system developers need to focus on business suitability and usability. Usability has a significant impact on the success of systems and products. It relates to the actual usage of a system, but also to its effective design and development. According to Landauer, failing to build usable system may degrade a project's ability to deliver in time, budget, functionality, and quality.

The science of usability engineering started to develop even earlier. Shneiderman (1980) suggested that software programmers could do better to ensure that the user of their programs find it friendly and easy to use. Norman and Draper (1986) introduced the principle of User-Centered Design (UCD) and Shneiderman (1987) proposed guidelines to implement these principles in the software design. Later, Card et al. (1983) explained and demonstrated how software developers can incorporate human factors to augment the productivity of software products. Today, it is a common practice to assign the task of user interface design (UID) to usability professionals, who know and understand the operational needs.

Human factors engineering

Any large organization whose mission is to design and develop systems for humans needs a well-developed integration and process plan to deal with the challenges that arise from managing multiple subsystems. Human capabilities, skills, and needs must be considered early in the design and development process, and must be continuously considered throughout the development lifecycle (Fitts et al., 1987). Human factors engineering (HFE) is a framework for describing how people may integrate with a human-made system.

People in prehistoric civilizations considered human factors whenever they had to design tools for their living. The need to assign an engineering term to natural behavior is due to accidents occurring during system operation (Meister, 1999). The *Encyclopedia Britannica* defines human-factors engineering, a "science dealing with the application of information on physical and psychological characteristics to the design of devices and systems for human use."

Traditionally, software engineers and systems engineers did not bother to consider the human factors. They preferred to focus on functionality and technology.

They used to delegate the responsibility for interaction design to human factors engineers. After getting some experience with operating interactive systems, it became clear that user interfaces designed by systems engineers are often difficult to use, and sometimes disastrous. Harel and Weiss (2011) proposed that the methods for mitigating human risks should be integrated in the system engineering practices. Eventually, they proposed to extend the scope of systems engineering, adding methods and guidelines to help protect from unexpected events. The idea presented to systems engineers was that they could benefit from considering human factors in the system design (Jackson and Harel, 2017).

For the purposes of systems engineering, it is helpful to consider two aspects of the HFE:

- The task view, in which we examine the ways people interact with the system
- The capability view, in which we examine physical and mental limitation of the human operators, hampering successful operation.

The challenges of the 4th IR about the task view are primarily in system thinking, about methodologies for system development, and about the opportunities to implement these methodologies using new technologies. The challenges about the capability view are about leveraging the physical and mental capacity of people through technology. This chapter focuses on the task view, namely, about methodologies for system development, enabling designers to improve the efficiency and safety in the human-machine interaction.

The** Human-System Integration **Engineering Framework

Unfortunately, systems engineers are not always aware of the benefits of considering human factors, and usability practitioners fail to explain their offer. There is a need to bridge this chasm from both sides. Systems engineers need to understand the benefits that they can get from incorporating human factors (Jackson and Harel, 2017), and usability practitioners need to demonstrate and explain to systems engineers how to integrate the theories of cognitive sciences in the system development. The way systems engineers implement their part is by "systems thinking." The way usability practitioners implement their part is by "cognitive engineering."

Recently, usability practitioners discuss challenges of incorporating human factors in system development (e.g. McDermott el al., 2017). Also, Sillitto et al. (2018) have distinguished interdisciplinary from transdisciplinary. Interdisciplinary has to do with multiple disciplines to accomplish a task. Transdisciplinary, on the other hand, has to do with using multiple disciplines together to accomplish the task. Transdisciplinary addresses the cooperation and collaboration between the disciplines. Hence, using engineering, psychology, and human factors together constitutes a transdisciplinary science.

Unfortunately, these works have not yet matured to an engineering discipline. The bridge that will enable crossing the chasm between SE and HF should be built using new methodologies about the way we define the interaction between the human operator and the machine. This bridge is the transdisciplinary framework of Human-System Integration (HSI), being developed recently. The following chart illustrates the location of the new discipline:

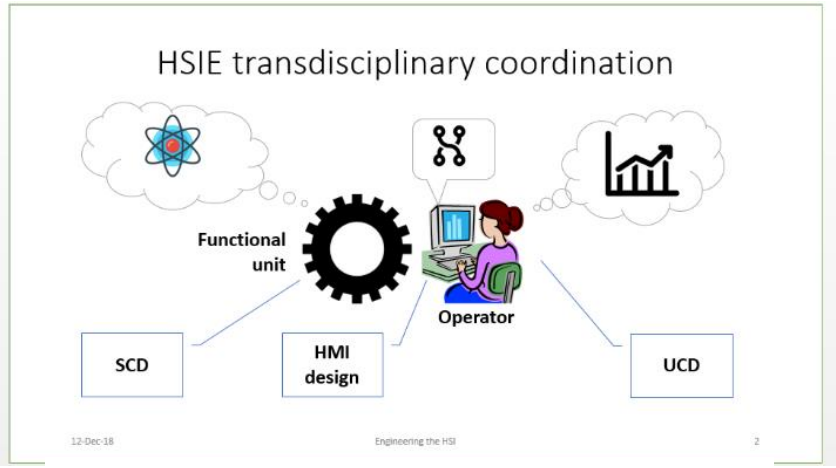

Figure 4 – HSI Engineering Discipline

HSI Engineering addresses the cooperation and collaboration between the disciplines. Hence, using engineering, psychology, and human factors together constitutes a transdisciplinary science. Unfortunately, these works did not mature yet to an engineering discipline. The bridge that will enable crossing the chasm should be built using new methodologies about the way we define the interaction between the human operator and the machine.

The transdisciplinary framework proposed here may enable to bridge the chasm between the SE and HF.

Human Machine Integration-RelatedTterminology

The concepts used in the Human Machine Integration (HMI) framework apply to various kinds of systems, defined in different industry domains. However, the use of HMI terms in the various domains is not the same. Also, terms used in a particular domain are not always adequate to the other domains. This HMI framework proposes a generic, common terminology, which may apply to all underlying domains. Table 1 compares terms of various concepts (column 1), in the context of consumer product (column 2) with those used in the context of safety-critical systems (column 3), and the generic term used in this chapter (column 4):

Table 1: Human Machine Integration (HMI) terms comparison

Concept	Consumer products	Safety-critical systems	Generic term chosen for this chapter
The human part of the system	User	Operator	Operator
The interface between the system and the human operator	User Interface (UI)	Control panel	Manual control
Designing for facilitating the human part of the interaction	UCD	Procedure design	HCID
Disruption from normal operation	Exception	Hazard	Diversion
Situation awareness	User orientation	Situation awareness	Situation awareness
Recovery from a disruption	Resumption	Resilience	Adjustment
Protection level	N.A. in 3rd IR	System Integrity Level (SIL)	Protection level
Protection analysis	N.A. in 3rd IR	Layer Of Protection Analysis (LOPA)	Protection analysis

The concepts of protection level and the protection analysis are specific to safety-critical system. The terms mentioned in this table apply specifically to the process industry. The generic terms used in this chapter are hereby explained:

Operators

The term User was adequate in the early years, when people used various electronic devices at home, and later, when they used computers for office application. Recently, the term Human replaced the original term, as it applies also to people integrated with the system, such as operators or passive people, who are not users.

It is helpful to use two distinct views of the operator: as a system controller and as a system unit. As a system controller, we are interested in functions: production, performance, effect, etc. As a system unit, we are interested in the operator's ability to make the system work, and about safety. For example, we want to detect a situation of a pilot passed out due to G-LOC (g-force induced loss of consciousness) and activate an Auto-GCAS (Ground Collision Avoidance System) to stabilize the airplane and the pilot (Dockrill, 2016).

As a system controller, the operator can have various roles: a user, motivated by functions and performance; a supervisor, motivated by the need to make sure that the system operates as intended; and a controller, who needs to manually make the system work. As a system unit, we are concerned about the operator's ability to function as a system controller, which is determined by qualification, motivation, vigilance, etc.

15.4 Human Factor Challenges typical of the 3rd Industrial Revolution

Understanding the Designer's Rsponsibility

A common design mistake, typical of the 3rd IR, is of assuming that the human operators can learn and keep operating according to all operational rules imposed on them. In reality, system operation often fails due to failure of the operators or users to follow the operational procedures, or to obey any other operational rules.

According to prior studies (such as by Zonnenshain and Harel, 2008) failure is mostly due to common flaws in the interaction design. Many operators have other tasks to do, and operating the system might hamper their primary tasks, which are not related to the system's operation.

The principle of HMI is that the system design should consider known limitations of the human operators, and design the interaction such that the system operation is seamless, and well protected from errors.

In the 4th IR, engineers are also concerned about the integration of the system operation with other operator's tasks, including operating complementary systems, as well as about the effect of the system on the environment.

Expanding the Scope of Quality Assurance

In the beginning of the 3rd IR systems, design focused on performance attributes, such as productivity and gain. Then, they realize that system failure is a key factor to performance assurance. They introduced the concept of system quality, which was about preventing system failure. The first stage in preventing system failure was to examine the failure of system components. At that stage, quality assurance was about the reliability of system components, namely, reliability assurance.

By and by, towards the beginning of the 4th IR system scientists gathered statistics about the sources of system failure, and realized that operator's errors contribute to failure even more than component failure. For example, it has been reported that 70-80% of the aviation accidents are due to human errors (Wiegmann and Shappell, 1997). Statistics about the productivity of text editing show that typing error result in extending the typing time by a factor of two, and that human errors are associated with most of the accidents. Consequently, in the 4th IR, the concept of quality should be extended, to preventing human errors, beside component reliability (Zonnenshain and Harel, 2015).

Similar to the traditional definition of quality, a way to evaluate the quality of a HMI is by measures of the rate of integration failure. We may regard the integration as being of high quality if the rate of failure is low. Accordingly, in order to evaluate the HMI, we need to assess the rate of various failure modes. A way to assess the failure modes was described by Zonnenshain and Harel (2015). According to their model, failure modes are associated with exceptional situations, namely, due to distraction from normal operation. Accordingly, the quality of a HMI may be assessed in terms of the rate of operating in exceptional situations.

Traditionally, according to scientific disciplines (Popper, 1968), quality assurance relies on verification testing. In the 4th IR, the most important concern about sustaining a required level of performance is the

operator's capability to respond gracefully to extreme conditions. The challenge of quality assurance in the 4th IR is about defining ways for assuring the system resilience to faults, both of critical components and of the human operators.

In the 4th IR the role of human factors (HF) is changing, and they lead the requirements. In the 4th IR, HF should be integrated in the design right from the beginning. HF should lead the requirements specifications, as well as critical design decisions, such as about the system architecture. The challenges of embedding HF in the 4th IR are to develop practices for:

- How to write usability-oriented requirements specifications
- How to translate the usability considerations to design features.

Expanding the Value of Human Machine I nteraction

Originally, the primary goal of UCD was to facilitate the operation of customer products (Norman, 1988). The vision was to design products that everybody can use. Therefore, the UCD methodology targeted the novices. Accordingly, the focus in the 3rd IR was on basic, normal operation, in normal situations. Later, gradually, the principles of UCD spread also to system design. Besides novices, the design principles also targeted experienced operators, which means that the design also targeted advanced stages of the operation. However, the focus still remained on normal operation, in normal conditions. Deviations from the normal were considered errors.

In the 4th IR, we expect that the design will support the whole life cycle. A primary challenge of HMI in the 4th IR is to develop practices for supporting operation in auxiliary activities, such as maintenance and training, and also unusual conditions, such as testing and troubleshooting.

The Focus of Human Machine Integration Design

In the 3rd IR (and also today), the primary concern of UCD is performance. New design practices are proposed for maximizing the performance in normal operational conditions, focusing on usability traits. For example, Boy (2016) suggested that tangible user interfaces may be easy and safe to use.

By and by, after gathering statistics about the sources of system failure, systems engineers realized that much of the operational time is spent operating in exceptional, non-productive situations. Typically, exception management is of lower priority in the system design. It is an informal task, not included in the requirements documents. It is a leftover, for the software engineers. Consequently, software development is often a bottleneck in the system development, typically behind schedule. The challenge of the 4th IR is to reduce the effect of exceptions. A challenge about HMI in the 4th IR is to develop practices for:

- Preventing deviation of the system situation from normal to exceptional
- Recovering from exceptional situations, and subsequently resuming normal operation.

Context-dependent Interaction Design

In the 3rd IR, the architecture of HMI was based on concepts borrowed from system design. A common practice of describing the interaction between two hardware units was through a thin layer, an interface: each of the hardware units is capable of receiving input from the other unit through the interface, process the received data locally, and send output to the other unit, through the interface.

In a typical architecture used for the HMI design, one of the two units was the system, and the other was the human operator. The interaction was through an interface, consisting of a control unit, enabling sending data and commands from the human operator to the system, and a display unit, enabling sending information about the system state from the system to the operator. A simple model commonly used in UCD proposes that the human activity is similar to that of the machine. However, in an HMI framework, the role of the two units is not symmetric. The human component is regarded as a black box. The interaction consists of combinations of the asymmetric architectures. The behavior of the human operator is subject to three mental activities, namely situation perception, decision making and command execution. Its behavior depends on information not available to the system designer, such as operational context, intentions, state of mind and personality. The designer cannot do anything to directly affect the operator's perception or decision making.

Using terms borrowed from hardware design, the human operator is a master and a client, and the system is a slave and a server. The master-slave model refers to the control aspect of the interaction, and the client-server model refers to the display aspect of the interaction. In the 3rd IR, it was the operators' sole duty to deal not only with the uncertainty, but also with unexpected behavior of the system part. This kind of architecture is described in the following chart:

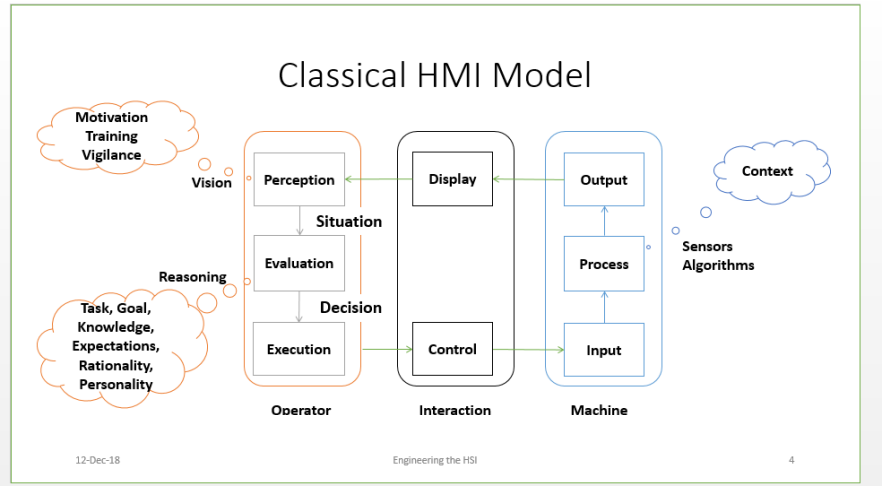

Figure 5 – Classical HMI Model

According to this model, the machine behavior is quite predictable, as long as all the components are reliable; besides the human operator, other external forces, defined as Context, also affect the machine behavior, through sensors, and algorithms. On the other hand, the behavior of the human operator is less predictable; the human perception is biased by improper vision, due to motivational, training and vigilance factors. The decision making, based on the situation evaluation, is biased by improper reasoning, due fuzzy task and goal setting, unsuited expectations, rationality and other personality traits of the human operator, and also by organizational pressures (Reason, 1990; Jackson and Harel, 2018). According to these models, the primary goals of UCD are: facilitating reliable perception, decision

support and protection from execution errors. The challenge of HMI in the 4th IR is to enable and guide system developers in designing systems that consider these factors. Limitations of this model are:

- It is process-oriented, therefore, it does not highlight the value of the system, in terms of performance
- Most significant factors affecting the system behavior are not in control. These factors are marked as clouds in the chart above.
- Poor validity, inasmuch as there is no practical way to measure these, or to estimate the direction or magnitude of their effect (Popper, 1968)

Human-machine Task Allocation

The dilemma of task allocation is about the integration of automated processes with those controlled by the human operator. Automation is required when the system needs to respond quickly and accurately, and when the operators are not capable of fulfilling this need. Examples are, when an airplane needs to respond quickly to a sudden wind blow, or when a pilot becomes unconscious due to extreme G force (Learmount, 2011).

Yet, automation might be a double-edged sword, when it disables the operator's control, as was the case the Air France 296 accident on June 26, 1988 (Casey, 1993). Automation is applicable only for problems in which the system response is well defined, namely, when the designer may assume that the data required for the response selection will be available when they are required, and that they are reliable. In the 3rd IR, this was not always the case. When the developers do not have all the information required to design resilient automation, they often let the operators take charge. The problem is that the operators often fail to recognize the situation (Norman, 1990), and often are not trained to handle it (Bainbridge, 1983).

In the 3rd IR, the task allocation was based on heuristics, namely, on the intuition and experience of the designers. A systematic approach to human-machine task allocation, commonly applied in the process industry, is by statistical process control (SPC). Kenett et al. (2009) suggested that we may apply this method also for usability control, by tracking usability changes. This method may be extended and applied for resilience control. The method is based on defining risk indicators. A risk indicator is a system parameter, such as operational temperature, with thresholds for warning, alarming, recovery time and emergency. The automation control is based on tracing the values of these parameters, calculating the trends, and responding when crossing the thresholds. The method applied to alarm design is illustrated in the following chart:

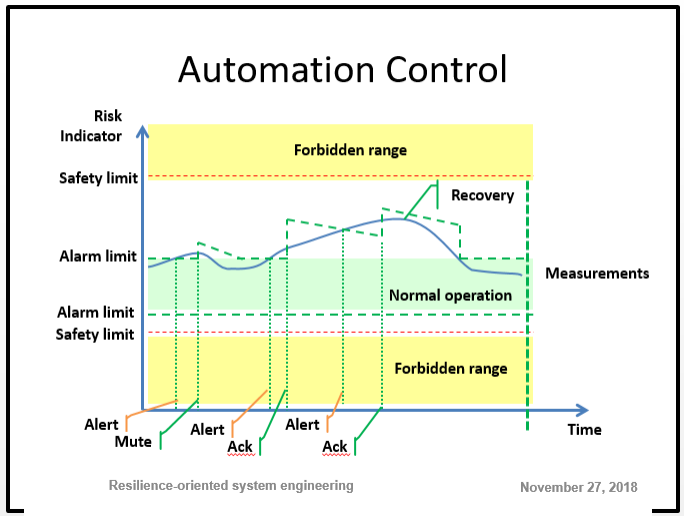

Figure 6 – Automation Control

In the 4th IR, we may develop practices for optimal task allocation. Such practices should apply to both normal and exceptional situations.

Managing the Human-machine Collaboration

Collaboration management is about adapting the operational procedures to scenario changes. The problem of collaboration management extends the problem of task allocation. The result of human-machine task allocation is a definition of the default behavior in normal and exceptional situations. Collaboration management is about situations when the default behavior is not appropriate. Specifically, when the operators need to take over a risky automated system behavior, as was the case of Air France 296 accident (Casey, 1993), or when the system needs to take over a risky manual control, as was the case of the Air France 447 accident (BEA, 2012).

In the 3rd IR, the collaboration management was based on heuristics, namely, on the intuition and experience of the designers. A key consideration was to assign tasks according to capability. Thus, routine tasks, requiring intensive, reliable data processing, which were easy to formulate, were assigned to the machine, while other tasks were assigned to the human operator. An example from transportation is the cooperation between the driver/pilot/captain and the machine. The machine needs to keep track of the route, in order to detect potential threats and to inform the operator about them. It is the operator's job to take over the machine and cope with the threat.

Accordingly, the collaboration may be by assigning the task of bookkeeping the history of system activity to the machine, while the operator's tasks are about deciding on critical changes in the operational procedures. For example, the machine can validate the operation according to the procedures, and notify the operators about distractions. The operator's tasks include responding to the notification, by changing the operational procedure, for example, to troubleshooting.

A challenge about collaboration management in the 4th IR is to develop practices, in form of protocols, for collaboration management. The protocols may define the conditions in which the operators may override automation, as well as those in which the operator may enforce automated behavior, overriding

the default manual control. The conditions may be stated in terms of priority, which may affect characteristics of the task termination, parameter preservation and reset, etc.

A new Human-machine Collaboration Model

Adversity in the system operation is manifested by operating in exceptional situations, and that mishaps are associated with difficulties in adapting to the exceptional situation. The traditional model of HMI is one-dimensional, based on the concept of user interface (UI), comprising a display unit and a control unit: the system displays it situation through the display unit, and the operators invoke their command using the control unit. This model does not represent well the need to deal with the exceptions. The following figure illustrates a new HMI model, emphasizing the role of human-machine collaboration, which may be implemented in the 4th IR.

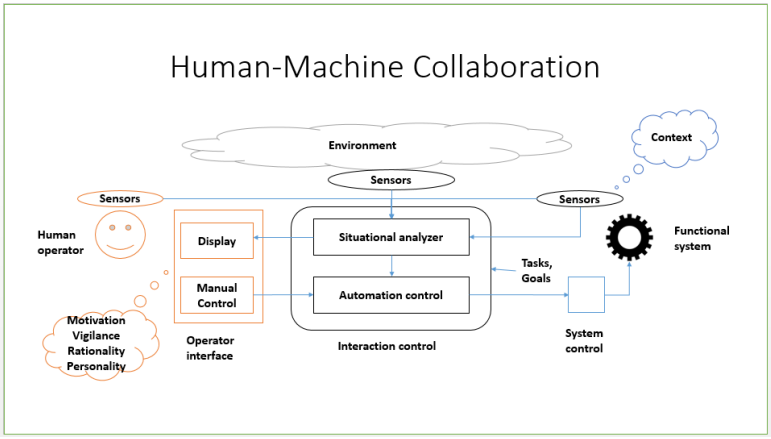

Figure 7 – Human-Machine Collaboration

The new model has several features, suited to comply with the opportunities of the 4th IR:

a. An operator interface replaces the traditional user interface. The significance of the terminology change is that the interface serves all kinds of operators, including those who are not users of the system. The display to the operators comprises information received from a situation analyzer.

b. A situation analyzer, tracing the situation changes, based on sensory data received from the system, the operator and the environment

c. An automation control unit, intended to control the automation according to the situation, enabling the operator to override the automated system control, and enabling the system to override improper operator's commands

The human machine collaboration (HMC) design model assumes that the quality of the operators' decision depends on the perception of the situation in general, and of exceptional situations in particular. Accordingly, the model focuses on avoiding, rebounding and recovery from exceptional situations. Critical factors, not controlled in implementations the classical model (above), are controlled in implementations based on the new model. The system operation is subject to rules, derived from the operational tasks and goals specified in the requirements documents. Operator's errors are controlled by automation.

Scenario-oriented Collaboration Design

Cooperation problems often emerge when the operators fail to comply with the operational procedure. This is often the result of a scenario change. Certain common barriers to successful integration may be attributed to the low level of interaction, and about insufficient support for changes in the operational scenario. In the 3rd IR, the interaction was defined in using an event-response model: the system responses to single actions, and the human response to information received from the system about its situation and activity. The specification of the high-level interaction was used as a framework for the interaction design. The design itself consisted of elaborating the specification in terms of the system response to events. What was missing is the operational context for the actions, namely, the system situation and the step in the operational procedure (the history of the action).

In the 4th IR, we realize that the model of atomic event-responses is not sufficient for defining adequate integration, because besides the event, the adequate response should also depend on the operational situation and the operational procedures. A challenging goal of scenario-based interaction design in the 4th IR is the development of means and practices for defining and storing task-level procedure protocols in a knowledge database, for tracing the interaction, and for responding to deviations from the required procedures. Such protocols may be used for controlling the operational procedures, to fit into the System-Theoretic Accident Model and Processes (STAMP) paradigm proposed by Leveson (2004).

Behavior Management

When a scenario changes, all the system units, as well as the operators, should synchronize with the change, to enable re-collaboration. Synchronization problems are due to wrong termination of the first scenario, or wrong resetting of the second scenario. Another problem is of rollback, namely, defining what happens when the second scenario ends, and how to resume the first scenario.

In the 3rd IR, the definition of the system behavior on scenario change was based on heuristics, namely, on the intuition and experience of the designers. The requirement documents did not specify the protocols of the scenario transition. A significant challenge of behavior management in the 4th IR is to develop practices, including priorities and protocols, for changing an operational scenario. For example, a standard protocol may describe the rules about scenario changes as follows:

- Save the system state, including procedure step
- Reset the system situation, to allow the next task
- Execute the interrupt
- Conditional (ask the operator) resume the interrupted procedure

Interaction styles

Interaction styles are attributes of the user interface, affecting the user experience. The attributes mostly discussed are visual styles, such as screen background and layout design, shape, color and density of screen objects, appearance attributes, such as animation, salience, location, etc. Styles apply also to

object selection, such as direct access vs. menu selection, and to audio attributes, such as melody, loudness, pitch, speed, etc.

Several guidelines about style definition were proposed during the 3rd IR. Shneiderman (1987) suggested to employ eight golden rules for user interface design. Nielsen (1993) proposed guidelines for designing web pages. Yet earlier, Shneiderman (1986) reported an experiment about interactive menu selection, in which interaction style optimized for novices was different from that of experienced user. Indeed, Zonnenshain and Harel (2015) have demonstrated that golden rules adequate for routine operation, with which the operators is very familiar, are not adequate for emergency operation, which the operators did not have any chance to experience beforehand.

In the 4th IR, the dilemmas of interaction styles should be examined and studied. Recently, Boy (2016) proposed the concept of tangible interaction as a means to improve the reliability of the human perception of the situation. However, different tangible interfaces may be adequate to different scenarios. A challenge about choosing the proper style is to define the boundaries of scenario validity of the various styles.

Beyond Root-cause Analysis

In the 3rd IR (and even today) design for failure prevention was based on cause-effect fault analysis. The focus of common practices was on failure of hardware components. Common practices for such analysis include Fault Tree Analysis (FTA), Event Tree Analysis (ETA), Hazard and Operability (HAZOP), Failure Mode and Effect Analysis (FMEA), etc. The focus was on the fault, triggering the sequence of events resulting in the accident. Other triggers, such as improper operator's action, or a software bug, are typically regarded as unfortunate or unexpected.

In the 4th IR, new failure models are available to the system designers, enabling them to handle also the unexpected. Scenario-oriented interaction design enables detecting situations of deviation from the operational procedures, even if the source for the deviation is unknown. Special procedures may enable the operators to respond gracefully to the unexpected situation, and to identify situations of operator's slip. Subsequently, system designers may explore the log of activity recorded during the operation, and decide on means to prevent recurring events.

A challenge about root cause analysis (RCA) in the 4th IR is to develop practices for coping with the unexpected, including:

- Develop database management systems (DBMS) enabling the system designers to define the operational procedures, store them, trace them at run-time, compare the current activity with the expected and notify about deviations
- Develop guidelines for responding to unexpected events gracefully
- Develop tools for analysis of recurring deviations from the expected, and for reporting on them.

Preventing Expected Diversions

Several methods for failure prevention were developed in the 3rd IR, based on cause-effect analysis. A well know example is the automatic shut-down of a system under high risk. Automatic emergency shut-down is part of collision avoidance systems (CAS), SCRAM at boiling water reactors, "reactor trip" at pressurized water reactors, etc. The common practice for deciding on activating the shut down procedure is based on absolute criteria.

Unexpected events are often due to operating in exceptional situations. Zonnenshain and Harel (2015) analyzed 67 incidents and developed a model of operational failure. A resilience model, based on the failure model, attributed the failure primarily to operating in exceptional situations. In the resilience model, the role of trigger is secondary to the primary source, which is operating in exceptional situations. A primary challenge in the 4th IR is to cope with exceptional situations, namely, to prevent them, and to facilitate the recovery, namely, the resumption of operating in normal situations.

Preventing unexpected diversions

Common system engineering methodologies and practices do not mitigate the risks of unexpected events, such as operator errors or mode errors, typically attributed to ‘force majeure’. Leveson (2004) proposed that the system should control its own behavior, by enforcing operation according to rules. Harel and Weiss (2011) proposed that we can mitigate such risks by design, by considering the human limitations in the interaction. Zonnenshain and Harel (2015) proposed that unexpected diversions are due to operating in exceptional situations. Therefore, to prevent unexpected diversions, the system needs to minimize the operation in exceptional situations. Also, they proposed that the system and the operators should collaborate in the troubleshooting and resuming a normal situation.

A challenge about preventing unexpected diversions in the 4th IR is to develop practices for balancing the need to enable operating in unexpected situations, we the need to protect from the unexpected events while in an unexpected situation.

Responding to Diversions

Protecting from failure is costly in terms of time and budget. Typically, system engineers inform the UI designer about certain failure modes known to bother the stakeholders, but not about those that did not materialize yet to real costs. To illustrate, consider an example of basic control of a simple machine, based on an On-Off switch. In a design typical of the 3rd IR, the switch enabled the basic functions of starting and stopping the machine. In a typical design scenario, a system engineer will analyze failure modes and require that the design will incorporate means to prevent these failures. However, it is rare that the systems engineer will require that the interaction design includes means to protect from operator’s errors, or that it includes procedures for troubleshooting. In the 4th IR, HMI practices may include guidelines for supporting features for error prevention and for designing troubleshooting procedures.

Protecting from the Unexpected

The effectiveness of this approach is limited, when a design needs to empower the human operator in order to cope with the unexpected. In the 3rd IR, unexpected events were nobody’s business: typically,

the design specifications did not mention them. Even today, it is the job of the software engineers to protect the system from unexpected events; typically, however, software engineers do not care much about how the operators may possibly become aware of such events, and how they can possibly recognize and understand the exceptional situation.

A theoretical limitation of these methods is that they deal with the expected, while many of the accidents are regarded as unexpected, even by hindsight. Harel and Weiss (2011) studied the nature of unexpected event, employing observations from Taleb (2007), and suggested the need to integrate in the system design special means for detecting and protecting from unexpected events.

New methodologies developed in the 4th IR enable systems engineers to prevent the unexpected. For example, the STAMP methodology enforces the system to behave according to rules, thus avoiding unexpected situations (Leveson, 2004). Robert et al. (1998) assumed that unexpected behavior is due to incomplete specifications, and proposed a methodology for assuring that the requirements about the acceptable situations are complete.

The interaction between the operators' tasks is a major contributor to operational errors. The need to putting people first implies that we need to consider ways to prevent user errors.

Expanding the Concept of Fault-tolerance

In the 3rd IR, the assurance of fault tolerance was based on heuristics, namely, on the intuition and experience of the designers. Typically, the design for fault tolerance was about specific critical component. A fault of the operator, such as due to illness or vigilance problems, was regarded as misfortune. For example, along the history of combat aircrafts, many of them crashed due to excessive G force, that the human body could not tolerate.

The new technologies developed in the 4th IR enable systems developers to shift much of the processing from the operators to the machine. More than ever before, automation enables overcoming embarrassing limitations of the human operators. For example, an autopilot can detect a situation of a pilot passed out due to G-LOC (g-force induced loss of consciousness), and alert colleagues, or activate safety measures such as Auto-GCAS (Ground Collision Avoidance System) to stabilize the airplane and the pilot (Dockrill, 2016). Referring to the examples above, in the 3rd IR, sensors for detecting sudden wind blow or for identifying the pilot mental state were not available, or were not reliable. In the 4th IR, such sensors may be available, enabling detection and proper responding also in extreme operational conditions.

Challenges to achieving fault tolerance in the 4th IR are to develop practices for assuring extended fault-tolerance, such as:

- Means to detect and protect from situations of disabled operators
- Notifying the operators about all deviations from the supported procedures
- Providing guidelines for designing resilient troubleshooting and recovery procedures

Exception Handling

Much of the productivity is wasted when the system reaches exceptional situations. The reason for this is because, the operators are not always trained well to cope with the exceptional situation, and the designers fail to facilitate the operator's mental activities in unfamiliar situations.

Software engineers often apply exception handlers to protect from expected exceptional situations. Even today, the unfortunate results of operating in exceptional situations is often regarded as an operator's error, and sometimes as unexpected.

A common limitation of traditional exception handlers is that they can capture only those exceptions which are expected, namely, which were identified in root-cause analysis. Another common limitation of traditional exception handlers is that they generate error message, of which the operators is not familiar.

In the 4th IR, exception handling may expand from normal to exceptional situations. A primary challenge about exception handling in the 4th IR is to develop practices, in form of protocols, used to define the interaction in exceptional situations. These protocols may include definition of protocols for capturing unexpected events, as well as of the interaction following the exception. The practices should include specification of the way the system may notify the operators about the exception, as well as practices for minimizing the nuisance due to the notifications.

Decision Support

A decision is often defined as a conclusion or resolution reached after consideration. Decision making is sometimes defined as deciding on an action to execute, such as selecting a choice from available options.

By design, we can control the likelihood of selecting a desired choice (Thaler and Sunstein, 2008). For example, Li et al. (2013) studied the effectiveness of alternative public policies targeted at increasing the rate of deceased donor organ donation. The experiment includes treatments across different default choices and organ allocation rules. The results indicate that the opt-out with priority rule system generates the largest increase in organ donation relative to an opt-in only program.

Obviously, if this is the case, then the design should prefer the automated selection over asking the operator to make a decision. Therefore, we should assume that at design time we cannot know what will be the best choice at run time. Rather, we need to rely on the reasoning of the decision maker.

Another method for affecting the option selection is by marking the preferred option as recommended. Both the default choice and the recommendation mark methods were employed extensively in the 3rd IR. However, the impact of employing these methods is much higher when people started to use the internet as a primary means for making business. An interaction designer needs to decide which of the two methods may be better for each of the possible decision situations. This is a challenge for the 4th IR.

Norman (1990) has demonstrated that often the problem of decision making is actually a problem of missing critical information. An operator needs information in order to make the proper decisions. The

challenge of decision support in the 4th IR is of providing the operators with the information they need. New guidelines should be developed, regarding dilemmas such as:

- How to make sure that the operators have all the information they need for the decision making
- How to make sure that the information required is available, and that the operators know how to get it
- How to make sure that the operators will notice the required information, and that they perceive it correctly
- How to avoid overwhelming the operator with distracting information.

Proactive HMI Terminology

In the 3rd IR systems engineers realized that they need to provide the operators with means to prevent failure, but they were not aware of the difficulties that the operators might experience in employing these means. If an operator failed to apply the means provided successfully, people attributed it to being irrational, or to making an error. The implication of this approach was that systems engineers did not feel responsible for assuring that the operators make the proper decisions, or for preventing errors. The approach to failure was reactive, namely, looking for someone to blame for the error.

The term "human error" often refers to an unintentional action that triggered a failure. Such definition is commonly used in studies of organizational behavior (Frese and Keith, 2014). The problem with this definition is that in many cases, the loss cannot be attributed to any unintentional action, or even to a judgment error. In these cases, this term should rather be attributed to the interactive complexity (Perrow, 1984), namely, to operating the system in exceptional situations (Hollnagel et al., 2006).

The new view of operators' errors is that the organization can and should prevent use errors. It does not make sense to demand that the operators avoid making errors, because they cannot. The operators follow the Human Factors version of Murphy's law: "If the system enables the operators to fail, eventually they will."

Operators' errors should be regarded as symptoms of the organizational deficiency that enables them, and not as the sources of the accident. The Human Factors Engineering approach to preventing user errors is by design, by considering the limitations of the users and the operators. This approach enables learning from incidents: instead of blaming the users, we focus on exploring why they failed, in order to understand how to prevent similar mishaps in the future. Recently, a new methodology for safety culture has been proposed, which defines the investigation of the stakeholders in the organization, such that safety considerations override personal interests (Reason, 1990).

The term *Use Error* is a new term which is recently replacing the term *User Error*. The need for changing the term was because of common mal-practice of the stakeholders (the responsible organizations, the authorities, journalists) in cases of accidents: instead of investing in fixing the error-prone design, the management attributed the error to the users (Dekker, 2007). User advocates (such as Reason, Hollnagel and Dekker) have noted that the user action is classified as an error only if the results are painful, implying that it is not the user who should be considered responsible for the errors (Hollnagel, 1983). The new term suggests that the incidence should be attributed to the way the system

is being used, rather than to the user. The term *'Use Error'* is used also in recent standards, such as IEC 62366: Application of Risk Management to Medical Devices. The standard defines *Use Error* as an:

> *"…act or omission of an act that results in a different medical device response than intended by the manufacturer or expected by the user"*

The term *Use Error* suggests that the error is the result of temporal conditions. However, from the record of accident investigations it is evident that use errors are enabled by poor design and other incessant operational conditions, imposed by the responsible organization (Reason, 1997). IEC 62366 includes an explanation (Annex A), emphasis ours:

> *"This International Standard uses the concept of **use error**. This term was chosen over the more commonly used term of "**human error**" because not all errors associated with the use of medical device are the result of oversight or carelessness of the part of the user of the medical device. Much more commonly, **use errors** are the direct result of "**poor user interface design.**"*

Failure to prevent a mishap may be regarded as a design or implementation mistake. In the 4th IR, the preferred approach to handle failure is proactively, namely, designing systems in which the operators cannot make errors. A problem in implementing the proactive approach is related to the terminology associated with failure. If the failure is associated with irrational behavior or with errors, then the system designer does not have sufficient incentive to prevent the failure. In case of failure, the investigation focuses on the operators' mistakes instead of finding ways to prevent similar failures in the future.

A challenge of human machine interaction (HMI) terminology in the 4th IR is to enforce the stakeholders to use proactive terms, especially in accident investigation. When the operators made a decision that seems improper by hindsight, the reference to this decision should be in terms of the circumstances of the decision, such as 'obscure information'.

Dynamic Adaptivity

In the 3rd IR, the HMI was examined in usability testing, prior to the system delivery and deployment. The problem is that after getting experience with the system operator, the operator's behavior changes, and the demands for effective interaction change accordingly. Designers believed that adaptivity helps the operators. Accordingly, Microsoft designed its Office applications with adaptive menus, eliminating, hiding or disabling menu items that were not used frequently.

The problem is that the operational demands change with experience. After the operators learned how to use the menu, they change, which means that the learning was useless. The operators need to learn the new setting, and to adapt the way the use the menus. This sequence repeats, as the system learns the new user's behavior, and responds by adapting the menu structure again and again.

Apparently, the straightforward adaptability described above is recursive, and to the user it appears as inconsistent. In the 4th IR, adaptivity should be by situation, not by the operator's experience. Items available to the operators are those relevant to the situation, and accordingly supported by the design.

The challenge of adaptivity in the 4th IR is to propose tools for assessing the benefits vs. the drawbacks of adaptivity, as well as guidelines for when to prefer it over consistency.

Interaction Adjustability

During the system development and after deployment, it is almost always the case that some of the requirements need to change. Adjustability is about the changes in the ways we adjust the system parameters, such as alarm thresholds, to the changes of the operational context.

A simple, straightforward technique for supporting agile development is by replacing constants by parameters, which the developers could change easily. This solution suited cases when the developer were not sure about the optimal value of a constant, for example, when the optimal temperature in a container in the process industry was not known before the integration testing. This solution suited also parameters used for the interaction design, such as risk indicators, used for alarming and for switching the operational mode.

In the 3rd IR, and even today, the search for the optimal parameter is often by trial and error. This method is often expensive in terms of time, and budget. Statistical methods, such as trend analyzers are often employed to extrapolate the effect of parameter changes. A challenge about interaction design in the 4th IR is about the practices for deciding about the need to change operational parameters, such as switching to safe-mode operation, or alarm thresholds. The tools required to handle the optimization process are handy, however, the efforts and investment required to incorporate them in a project are remarkable. We may expect that in the 4th IR new methods and practices will be developed, enabling fast and easy adjustment of system parameters in general, and specifically of parameters used in the interaction.

In the 3rd IR the thresholds defining the system behavior in various situations were often defined at design time. Often, the designers enabled customizing the behavior, by changing the thresholds. The typical way to decide on the need to change the settings was by experience. Typically, the way to decide on the rate of change was by trial. Typically, there were no guidelines for deciding on the optimal rate of signal to noise (S/N) in the variety of operational situations. Being unable to decide when and how to change the thresholds, system operators are reluctant to adjust the system behavior to the situation.

The adjustability dilemma, namely, how to adjust the thresholds to the situation, has not yet been resolved even theoretically. A primary challenge for the 4th IR is to define guidelines for defining the optimal rate of S/N in various operational situations. Zonnenshain and Harel (2015) proposed an architecture enabling capturing situational changes, as well as means for getting the S/N ratio.

A challenge about interaction adjustment in the 4th IR is about the practices for deciding about the need to change operational parameters, such as switching to safe-mode operation, or alarm thresholds. The 4th IR may present opportunities for applying data mining techniques in order to define guidelines for adjusting the S/N ratio. We may expect that in the 4th IR new methods and practices will be developed, enabling fast and easy adjustment of system parameters used for the interaction.

Situation Awareness

Situation awareness is about the human operators understanding of system status and the actual system state (Woods, 1988). Problems of situation awareness are key factors in accident development. Norman

(1990) attributed this kind of problems to design flaws, of not providing the operators with the information about the situation, which is critical for deciding on the appropriate behavior.

Often, in the 3rd IR, the problem if situation awareness is due to missing information about the risks. For example, if a sensor designed to notify about a risk does not function, as was the case of the Pressure Operated Release Valve (PORV) in the Three Miles Island (TMI) accident (Perrow, 1984). A primary barrier to solving this problem is primarily technological. The 4th IR offers a solution to this problem, as new sophisticated, reliable means and methods are being developed, which may fulfill future needs. However, there is a problem of managing the high volumes of data, and eliciting information relevant to the operator's perception of the situation.

Another problem is of nuisance due to excessive or non-specific notifications. For example, Harel (2006) studied the sources of the failure of the public to respond properly to alarms about missile attacks during the Israeli war with Hezbollah, and concluded that the reason was the it was due to such nuisance. Situation awareness should be evidence-based, namely. This means that the information used for the decision making should be elicited from a large body of data recorded during the operation. A challenge about ensuring situation awareness in the 4th IR is about the ways to manage the big data, and to mine it and how to present it to the operators. The tools required to handle the information elicitation for the big data are handy, however, the efforts and investment required to incorporate them in a project are remarkable. We may expect that in the 4th IR new methods and practices will be developed, enabling fast and easy information elicitation in general, and specifically of the information required to facilitate the operator's situation awareness.

Another problem to solving this problem is design neglect. In today's sophisticated, complex systems, it is quite probable that a designer might forget to include a sensor about a critical component in the design. Common design practices include ad hoc guidelines for testing critical components, but in a complex system even simple screws might become critical. Can we add a sensor for each screw in a complex system? This is a challenge for the 4th IR.

Information Overload

Let us assume that our system in the 4th IR has all sensors and algorithm to measure and report on all situational variables. The complementary problem is that the operators are overwhelmed with information, most of it irrelevant to their task. The problem is that they cannot find the needle in a haystack. The challenge in the 4th IR is to identify the information required for the decision that the operators need to make, and to present it to the operators, and nothing else. HMI in the 4th IR encourages interaction designers to present the data obtained from sensors in forms of information essential for proper decision-making, and to attract the operator's attention to indication about critical situations.

Rebounding from Operator Slips

A method for preventing errors due to operator slip, commonly employed in the 3rd IR, is by verifying that the system can handle the operator's input at the particular situation. Theoretically, system designers can specify the cases in which the system should reject the input in the requirements documents.

However, the amount of work involved in writing such requirements is huge, making it impractical. A practical method, is to specify the rules for proper operation, namely, the operational procedures, and to enforce the system to operate bound to these rules (Leveson, 2004).

A challenge about rebounding from operator slips in the 4th IR is to develop practices for specifying the operational rules, and for transferring them to a database that the system can use to verify that the operation complies with the rules.

Error Prevention

The best way to learn how to operate a system is by trial and error (Jones et al., 2010). In order to enable this kind of learning, the interaction should mitigate the risks of costly errors. In the 3rd IR, software providers did not invest in preventing costly errors. As a result, the operators hesitated before trying, and avoided trying features that they did not know. The term Error is used extensively in investigations, implying that the operator are accountable for the incidence, in order to justify distracting the discussion from costly investment in resilience assurance, to cheap and handy personnel changes (Harel, 2011).

To prevent errors, we need to understand them. Norman (1980) and Rasmussen (1982) proposed methods for classifying human errors. Also, Norman (1983) proposed various design rules based on analysis of human errors. Hollnagel et al. (2006) suggested that errors are tightly connected to deficiencies in the system resilience. It is a challenge of the 4th IR to develop and enforce guidelines for preventing costly errors, in order to facilitate learning by trial and error.

Proactive Investigation

Traditionally, safety engineering in the 3rd IR used to focus on the system side of the integration. Practices of safety assurance include assuring the reliability of critical components, robustness and redundancy. Typically, safety engineers did not deal with the human side.

Investigators of many celebrated accidents, such as in transportation and in the process industry, attributed the source of the accident to the operators' behavior, explaining that it did not match the system situation. Often, they attributed the behavior-situation mismatch to ambiguity of the HMI (e.g. Perrow, 1984). Traditionally, people expect the users to follow the operational instructions, and avoid making errors. In case of a use error, the user is accountable. For example, people expect that nurses respond promptly to all medical alarms, even though most of them are irrelevant. In case of an operational error, the operator is to blame. People expect that operators understand the safety implications of each option that they choose during the operation, in any future operational situation, based on unknown designers' reasoning.

In practice, users often fail to identify exceptional operational situations, to recall the operational instructions, and to predict the system behavior in these situations. Typically, in case of an accident, we accuse the user for negligence, and we accuse the operator for unreasonable operation. We consider the user errors as the source of the accident. In fact, most of the accidents are attributed to user errors.

Typical reaction to accidents in the 3rd IR were, and are still today, emotion-driven. Following an incident or an accident, the people involved typically focus on investigation issues rather than on improving the safety. In emotion-driven organizations, where the safety culture is biased, incident investigations often obey the "blame and punish" script (Zonnenshain and Harel, 2015). Emotion-driven response to incidents prohibits improving resilience, because the investigations do not focus on the design changes needed to improving the resilience. On the other hand, when the organization adopts safety culture, the investigations include recommendations for design changes, and the management promotes implementing these recommendations (Dekker, 2007). The guide proposes a procedure for continuous improvement of the system resilience by learning from mishaps, preventing this bias (Weiler and Harel, 2011). Following an incidence or an accident, the stakeholders' reaction is typically emotion driven. In emotion-driven organizations, where the safety culture is biased by investigation, incidence investigation often follows the "blame and punish" script, attributing the incidence to the operators.

Emotion-driven response to incidents hampers the efforts to learn from them, because the investigations focus on the stakeholders instead of on the design changes needed to improve the resilience. Typically, they focus on investigation issues (looking for "bad apples," Dekker, 2007) instead of on improving the safety. If a person blames the operators, then it may be the case that this person wants to distract the blame, or his/her impotence, to shift it to those who cannot protect themselves.

The interest of the stakeholders determines whether the decision is rational. In order to guarantee that learning from failure is effective, we need to avoid judging the decisions in terms of rationality. In case of a costly accident, the stakeholder often focus on looking for a person how may be nominated as responsible for the accident. This understandable behavior results in overlooking possible ways to prevent similar accidents by redesign.

Complying with the accountability bias is convenient for safety administrators, because if the operator is accountable for the accident, this implies that they are not. The accountability bias distracts the focus from the stakeholders, in charge of safety, to the operators, the victims of the design flaw (Jackson and Harel, 2018). The problem with this approach is that it inhibits processes of safety improvements. The users' typical response is to think more about their own risks, and less about the interests of the organization, or the public. The organization avoids acting to improve safety, because such actions are likely to manifest the accountability of the safety administrators (Dekker, 2006). For example, admitting the design mistake that cause the Airbus 320 accident in Mulhouse Habsheim in 1988 could have prevented the accident in Bangalore, India in 1990 (Casey, 1993). In this case, the safety administrators preferred to accuse the pilots instead of exploring the systemic circumstances. Also, accusing members of medical teams for accidents due to risky operational procedure is quite common.

The New View approach is often criticized for encouraging carelessness during the operation, which might result in accidents. Safety administrators often apply such reasoning to justify setting the system in ways that transfer their investigation to the users, which are risky to the public (Decker, 2007). For example, safety administrators are tempted to set alarm thresholds such that the users are overwhelmed with irrelevant alarms, in order to reduce the risks of missing alarms when needed.

In order to assure learning from incidences, we need to know the barriers to learning. The main obstacles to improving by learning from accidents are human biases, by the system stakeholders. A common practice of organization for coping with the investigation bias is by adopting a safety culture. This methodology encourages that investigations should include recommendations for design changes.

However, this methodology contradicts the human nature of blaming people. People praise the concept of safety culture as long as everything goes according to the plans. In case of mishaps, people typically change their strategy, and turn to their natural behavior, of blaming others.

The IEC 60601-1-8 standard (2006) triggers awareness of the risks involved in using medical alarms, by warnings about what might go wrong. However, it does not provide sufficient guidance for how to avoid these risks. Insufficient guidance about maintaining "safety culture" results in organizational settings that over-protect the authorities, leaving "holes" (in terms of the "Swiss Cheese" model by Reason, 1990) in the patient safety (Harel, 2011). Barriers to learning from incidences include:

- Reporting on incidences might suffer from the investigation bias
- Information extraction might suffer from lack of means for data aggregation
- Improving the operational procedures might be hampered by the responsible organization

To enable learning from incidences we need to provide data about the circumstances of the mishap. The challenge of ***proactive*** investigation in the 4th IR is to develop tools and practices to capture, analyze and report about incidences. These practices should consider the investigation bias, and provide defenses against it.

Resilience Development

In the 3rd IR people believed that fixing a design flaw should always improve the system safety. For example, Jackson (2009) discusses "a framework for implementation that both public and private organizations can use as a guide to establishing procedures for anticipating, surviving, and recovering from disruptions." In reality, this is not always the case. Indeed, Popper (1968) argues that "non-reproducible single occurrences are of no significance to science. Thus a few stray basic statements contradicting a theory will hardly induce us to reject it as falsified. We shall take it as falsified only if we discover a reproducible effect which refutes the theory".

The implication of this observation to engineering is that we can never be sure that adding a safety feature is always safe. Unfortunately, after fixing a safety problem, it may happen that the upgraded system suffers from a new, latent problem. A famous example is the reliability problem of the PORV indicator, which was fixed following a near miss in the Davis Hesse II nuclear power plant in 1978. The problem was fixed and the PORV indicator of the upgraded TMI nuclear power plant was more reliable. Because it was more reliable, the operators relied on it, even thought it did not function properly.

In the 4th IR, usability trackers will be embedded in the software delivered, enabling the system developers to learn the problems that the operators experience during the operation (Harel, 1999).

15.5 Summary

This chapter addresses possible changes in the 4th IR relating to the role of human factors in systems engineering. The chapter suggests that the 4th IR may involve various shifts towards HMI, associated with technology, methodology and HMI thinking and practices. These shift may affect the people productivity, quality of life and safety.

The main conclusion is that in the 4th IR, HMI needs to evolve into a scientific discipline, offering degree programs in universities that teach students the essentials of HMI engineering and interaction design.

References

Arnold, R.D. and Wade, J.P. (2015) A Definition of Systems Thinking: A Systems Approach. *Procedia Computer Science* 44, 669 – 678.

Bainbridge, L. (1983) Increasing levels of automation can increase, rather than decrease, the problems of supporting the human operator. *Automatica, 19, 775-779.* Reprinted in: (1987) Rasmussen, J., Duncan, K. and Leplat, J. (eds.) *New Technology and Human Error, Wiley, Chichester*, pp. 276-283.

Bartolomei, J.E., Hastings, D.E., de Neufville, R., and Rhodes, D.H. (2006) Screening for Real Options "In" an Engineering System: A Step Towards Flexible Weapon System Development. 4th Conference on Systems Engineering Research.v Los Angeles, CA, USA. April 2006.

Bertalanffy, L. von. (1968) *General System Theory: Foundations, Development, Applications*, rev. ed. New York: Braziller.

BEA (France) (July 2012), Final report On the accident on 1st June 2009 to the Airbus A330-203 registered F-GZCP operated by Air France flight AF 447 Rio de Janeiro – Paris

Boy, G. (2016) *Tangible Interactive Systems: Capturing the Real World with Computers*. New York: Springer. ISBN 978-1-4471-4338-3.

Boy G.A., Narkevicius J. (2013) Unifying human centered design and systems engineering for human systems integration. In Aiguier M, Boulanger F, Krob D, Marchal C (eds) *Complex systems design and management*. Springer, London, 2014. ISBN-13: 978-3-319 02811-8.

Card, S., Moran, T., and Newell, A. (1983) *The psychology of human- computer interaction.* Hillsdale, NJ: Lawrence Erlbaum Associates Inc.

Casey, S. (1993) A Leap of Faith"; in *S. Casey: Set Phasers on Stun, And Other True Tales of Design, Technology and Human Error*, Aegean Publishing.

Cooper, A. (1998) The Inmates Are Running the Asylum: Why High Tech Products Drive Us Crazy. Sams - Pearson Education.

Dekker, S. (2006) *The Field Guide to Understanding Human Error*, Ashgate Publishing.

Dekker, S. (2007) *Just Culture: Balancing Safety and Investigation*, Ashgate Publishing.

Dockrill, P. (2016) WATCH: F-16 Autopilot System Saves The Life of an Unconscious Fighter Pilot, *Science Alert*, 14 Sept. 2016.

Fitts, D.J, Sandor, A., Litaker, H.L. and Tillman, B. (1987) *Human Factors in Human Systems Integration*. NASA Space HFE Project.

Frese. M. and Keith. N. (2014) Action errors, error management, and learning in organizations, *Annual Review of Psychology* 66(1).

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

Harel, A. (2006) Alarm Reliability: What If an Alarm Goes Off and No One Hears It? *User Experience Magazine*: Volume 5, Issue 3.

Harel, A. (2010) Whose Error is This? Standards for Preventing Use Errors, *The 16th Conference of Industrial and Management Engineering*, Tel-Aviv, Israel.

Harel, A. (2011) - Comments on IEC 60601-1-8. Letter submitted to IEC/TC 62 working group.

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

Hollnagel, E., Woods, D. and Leveson, N. (2006) *Resilience Engineering: Concepts and Precepts*. Aldershot, UK: Ashgate Publishing Limited.

Hollnagel, E. (1983) Human error. Position Paper for *NATO Conference on Human Error*, August 1983, Bellagio, Italy.

Hollnagel, E. (2009) *The ETTO Principle: Efficiency-Thoroughness Trade-Off. Why Things That Go Right Sometimes Go Wrong*. Ashgate.

IEC 60601-1-8 (2006) Medical electrical equipment -- Part 1-8: General requirements for basic safety and essential performance -- Collateral standard: General requirements, tests and guidance for alarm systems in medical electrical equipment and medical electrical systems. International Electrotechnical Commission *(20) (PDF) Medical Audible Alarms and IEC 60601-1-8*.

Jackson, S. (2009) Architecting Resilient Systems: Accident Avoidance and Survival and Recovery from Disruptions. In: *Andrew P. Sage, (ed.) Wiley Series in Systems Engineering and Management*. Hoboken, NJ, USA: John Wiley and Sons.

Jackson, S. and Harel, A. (2017) Systems Engineering Decisions Analysis can benefit from the Added Consideration of Cognitive Sciences. *Systems engineering magazine SyEN 55*.

Jackson, S. and Harel, A. (2018) Improving Decisions to Mitigate the Risks of Organizational Accidents. *Preprints*, 2018030042 (doi: 10.20944/preprints201803.0042.v1).

Jones, R.S.P. and Clare, L. and MacPartlin, C. and Murphy, O. (2010) *The Effectiveness of Trial-and-Error and Errorless Learning in Promoting the Transfer of Training.* European Journal of Behaviour Analysis, 11 ((1)). pp. 29-36.

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

Kenett, R.S., Zonnenshain, A. and Swarz, R.S. (2018) Systems Engineering, Data Analytics, and Systems Thinking: Moving Ahead to New and More Complex Challenges, *Proceedings of the 28th annual international symposium of INCOSE*, Washington, DC.

Kopainsky, B., Alessi, S.M., and Davidsen, P.I. (2011) Measuring Knowledge Acquisition in Dynamic Decision Making Tasks. In The 29th International Conference of the System Dynamics Society (pp. 1 – 31). Washington, DC.

Landauer, T.K., 1996. *The Trouble with Computers: Usefulness, Usability, and Productivity*. A Bradford Book.

Learmount, D. (2011) *Global airline accident and safety review for 2010*. FlightGlobal Aviation Connected.

Leveson, N.G. (2004) A New Accident Model for Engineering Safer Systems, *Safety Science*, Vol. 42, No. 4, pp. 237-270.

Li, D., Hawley, Z. and Schnier, K. (2013) Increasing organ donation via changes in the default choice or allocation rule. *J Health Econ*. Dec;32(6):1117-29. doi: 10.1016/j.jhealeco.2013.09.007. Epub 2013 Sep 20.

McDermott, P.L., Ryan, M., Bonaceto, C., Potter, S., Savage-Knepshield, P., Kosnik, W., Dominguez, C. (2017) Improving How We Weave Cognitive Engineering into Military Acquisition: Understand, Design, Validate. *Proceedings of the Human Factors and Ergonomics Society* Annual Meeting.

Meister, D. (1999) *The History of Human Factors and Ergonomics*, CRC Press.

Nielsen, J. (1993) *Usability Engineering*. Academic Press, Boston.

Norman, D.A. (1980) Errors in human performance. *National Technical Information Service, US Department of Commerce*. Springfield, VA.

Norman, D.A. (1983) Design Rules Based on Analyses of Human Error. *Communication of the ACM*. 26(4):254–258.

Norman, D.A. (1988) *The Design of Everyday Things*. New York: Basic Books. ISBN 978-0-465-06710-7.

Norman, D.A. (1990) The "problem" of automation: Inappropriate feedback and interaction, not "over-automation". In D. E. Broadbent, A. Baddeley and J. T. Reason (Eds.), *Human factors in hazardous situations* (pp. 585-593). Oxford: Oxford University Press.

Norman, D. A. and Draper, S. W. (Editors) (1986) *User-Centered System Design: New Perspectives on Human-Computer Interaction*. Lawrence Earlbaum Associates, Hillsdale, NJ.

Perrow, C. (1984) *Normal Accidents: Living with High Risk Technologies*. NY: Basic Books..

Popper, K. (1968) *The Logic of Scientific Discovery*, New York: Harper and Row.

Rasmussen, J. (1982) Human Errors. A Taxonomy for Describing Human Malfunction. in *Industrial Installations, Journal of Occupational Accidents*, 4, Elsevier Scientific Publishers, pp. 311-335.

Reason, J.T. (1990) *Human Error*. Cambridge, UK: Cambridge University Press.

Reason, J. T. (1997) *Managing the risks of organizational accidents*. Aldershot, Hants, England: Ashgate.

Richmond, B. (1994) Systems Dynamics/Systems Thinking: Let's Just Get On With It. In *International Systems Dynamics Conference*. Sterling, Scotland.

Robert, D., Berry, D., Isensee, S. and Mullaly, J. (1998) *Designing for the User with OVID: Bridging User Interface Design and Software Engineering*, Software Engineering Series, Macmillan Technical Publication.

Schwab, K. (2016) *The Fourth Industrial Revolution*. World Economic Forum. ISBN 1944835008.

Senge, P. (1990) *The Fifth Discipline, the Art and Practice of the Learning Organization*. New York, NY: Doubleday/Currency.

Shneiderman, B. (1980) Software Psychology: Human factors in computer and information systems. Winthrop computer systems series.

Shneiderman, B. (1986) Designing menu selection systems. *Journal of the American Society for Information Science*, 37 (2) 57-70.

Shneiderman, B. (1987) *Designing the user interface: strategies for effective human-computer interaction*, Reading, MA: Addison-Wesley.

Sillitto, H.G., Martin J., Griego R., McKinney, D., Arnold, E., Godfrey, P., Dori, D., Krob, D. and Jackson, S. (2018) Envisioning Systems Engineering as a Transdisciplinary Venture. *IS 2018*, Washington, DC.

Standish Group (1995) *The COMPASS report*, Forbes.

Stave, K.A., and Hopper, M. (2007) What Constitutes Systems Thinking? A Proposed Taxonomy. In *25th International Conference of the System Dynamics Society*. Boston, MA.

Sweeney, L.B., and Sterman, J.D. (2000) Bathtub dynamics: initial results of a systems thinking inventory. System Dynamics Review, 16(4), 249–286. doi:10.1002/sdr.198.

Taleb, N., 2007. *The Black Swan: The Impact of the Highly Improbable*, Random House Trade Paperbacks.

Thaler, R.H. and Sunstein, C.R. (2008) *Nudge: Improving Decisions about Health, Wealth, and Happiness*, Yale University Press, New Haven, CT.

Wiegmann, D. and Shappell, S. (1997) Human factors analysis of post-accident data: Applying theoretical taxonomies of human error. *The International Journal of Aviation Psychology*, 7, pp. 67-81.

Weiler, M. and Harel. A. (2011) Managing the Risks of Use Errors: The ITS Warning Systems Case Study. *The 6th Conference of INCOSE-IL*, Herzelia, Israel.

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

Woods, D.D. (1988) *Coping with complexity*: The psychology of human behavior in complex systems. In L.P. Goodstein, H.B. Andersen, and S.E. Olsen, editors.

Zonnenshain, A. and Harel, A. (2008) Extended System Engineering – ESE: Integrating Usability Engineering in System Engineering. *INCOSE International Symposium*, 18(1), Utrecht, the Netherlands, June 15–19, pp. 1542-1556.

Zonnenshain, A. and Harel, A. (2015) A practical guide to assuring the system resilience to operational errors. *INCOSE Annual International Symposium*, Seattle.