Engineering the HSI

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

Avigdor Zonnenshain
Gordon Center for Systems Engineering
Technion
Haifa 32000, Israel
+972 52 289 1773
avigdorz100@gmail.com

ABSTRACT

This article is about ways people may be integrated in systems and about engineering activities enabling successful integration. The article proposes a two-layer model of humans in system integration (HSI) thinking. The outer layer is about improving the societal impact of the system, productivity, quality of life and safety. The inner layer is a framework of HSI engineering, integrating methodologies and tools used in user centered design (UCD) with special methodologies and tools proposed for human machine interaction (HMI) design (HMID). The goals of HSI engineering may be defined in terms of performance and reliability, based on a model of the situation and their transition in system operation. HMID may focus on preventing, detecting and handling exceptional situations. The article presents an architecture applicable to the 4th industrial revolution, for detecting both predictable and unexpected diversions from routine operation, as well as for comprehensive, adaptable exception handling.

Keywords
HSI, HMI, UCD, models, engineering, interaction, design, exception, architecture.

INTRODUCTION

Case Study

Few years ago we bought a new drier for our home. It had only few options, and it looked very easy to use. It had few controls, with nice, clear icons marking the meaning of these controls. Two of the controls were for time setting: one for the drying time, the other for delay. We used the drier occasionally. Then, one day it did not start. We were about to call a technician to fix the machine. Fortunately, we noticed just in time that we had set the delay control instead of the timer. Luckily, we saved the embarrassment and the costs of unjustified visit of the technician. By the time the technician would arrive, the delay would have terminated, and the drier would work perfectly, as designed.

HSI2019, September 11-13, 2019, Biarritz, France.

System Integration (SI)

System Integration (SI) is a methodology of Systems Engineering (SE) for creating a functional system by integration of non-functional or partly functional system modules. The focus of SI is on the validation of the interaction between the modules. The Systems Engineering Body of Knowledge (SEBoK) classifies this discipline in the category of specialty engineering, where 'specialty engineering' is "the collection of those narrow disciplines that are needed to engineer a complete system" ([1]).

Humans in System Integration (HSI)

The concept of HSI emerged with the understanding that an operational procedure is more than the sum of event-responses on the way to accomplishing a task.

Traditionally, the human operators were not treated in the system analysis, specification, design and testing, as parts of the system. Instead, the relationships between the operators and the system were according to the client-server models, where the operators are the clients and the system is a server. The resulting systems suffered from problems during the system operation, characterized by mismatch between the operator's intention and the system behavior. HSI is an extension of SI, in which part of the modules are human operators.

The need for a discipline

Traditionally, systems engineers and software engineers disregard the ways people might operate the system. We can learn about the need for a discipline from an early report about the need for applying engineering disciplines in software development. The Standish Group 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 examination of the way engineers manage projects of building bridges. 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" ([15]). This article proposes a definition of a new discipline, HSI engineering, intended to avoid repeating interaction failure.

The lacuna

SEBoK includes a list of topics that should be addressed by an engineering discipline, and a description of each of the topics as addressed by HSI. Many of the topics of this list, including discipline management, discipline relationships with interactions and relationships, discipline standards, metrics, models, tools, pitfalls, proven practices and other practical considerations, are pending. Currently, these topics still are tagged as "Information to be supplied at a later date" ([1]). The article addresses the lacuna of these topics, focusing on the interaction between the operators and the machine.

The scope of HSI design

SEBoK adopts the definition of HSI by ISO/IEC/IEEE 2011, as "an interdisciplinary technical and management process for integrating human considerations with and across all system elements, an essential enabler to systems engineering practice." The domain considerations include: "manpower, personnel, training, human factors engineering, occupational health, environment, safety, habitability, and human survivability" ([1]). The 4th industrial revolution is about a shift in our view of the effect of technology on our experience of using systems. The potential impact applies to various kinds of resilience-critical systems:

- Safety critical systems - where the impact could be injury, environmental impact etc,
- Performance critical systems where it might impact on profit, efficiency
- Consumer and entertainment products - frustration, loss of sales, negative brand image etc.

System Thinking

It is commonly agreed by system scientists that the system development ought to begin with 'system thinking'. According to SEBoK, system thinking is the application of system sciences to assist in solving real world problems.

Richmond ([12]), the originator of the term, defines systems thinking as "the art and science of making reliable inferences about behavior by developing an increasingly deep understanding of the underlying structure" ([1]).

Human Machine Interaction (HMI)

System operation involves interaction between the human operators and a machine. In traditional systems engineering (SE) the human operators are users of the machine. The users are not part of the system design, and they interact with the machine via User Interfaces, which also define the boundaries between the machine and its users.

In the system design we need to consider the capabilities and limitations of the user, but we need to admit that we have little effect on their behavior. Therefore, we need to focus on adapting the machine to the human operators; accordingly, the subject of the system design is the machine, not the human operators. This premise is the motivation for the UCD paradigm ([11]), which subsequently developed to cognitive engineering and to the user experience (UX) approach to modern systems. Traditionally, the HMI is defined in the system specification. At design time the interaction is expressed in terms of task-oriented operational procedures. The operational procedures are defined as sequences of short event-response sessions, typically independent of each other, avoiding dealing with the complexity of situation-dependent activity.

Modeling the HSI

To evaluate the role HSI we need to agree on what makes a system successful. We may examine case studies of both good and poor systems from various domains, and we may apply statistics. Boy suggested that HSI development may be modeled similarly to the way an orchestra performs music [2]. The orchestra model compares development activities with those of the people employed in music production, such as the composer, the instruments, the players, the conductor, the audience, etc. This model emphasizes the channels used for information transfer between team members, highlighting the similarity between the function of music notes and that of requirements specifications.

Modeling the HMI

Traditionally, the common architecture of HMI was based on concepts borrowed from system design. 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. 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. This kind of architecture is described in the following chart:

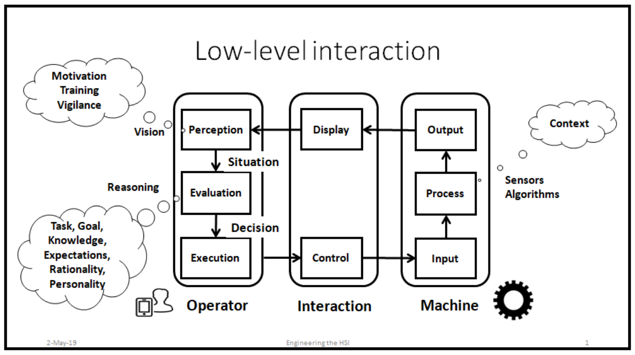


Figure 1 – low-level interaction

The traditional integration model is one-dimensional, based on the concept of user interface (UI). 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. The interaction between the machine and the operators is defined in terms of short sessions of event-response, by a display unit and a control unit:

- The machine gathers data from sensors about the environment and the machine itself, processes the data to update and evaluation the system situation, to obtain measures of performance and risks, and displays situational information regarding the system state and performance, environmental conditions, etc. through the display unit
- The operators enter data and operational parameters, and invoke their commands using the control unit.

Besides the human operator, other external forces also affect the machine behavior, through sensors and algorithms.

The machine behavior is quite predictable, as long as all the components are reliable; On the other hand, the behavior of the human operator is less predictable, especially when applying new artificial intelligence and machine learning technology. An explanation to this hypothesis is that the human perception is biased by improper vision, typically attributed to motivational, training and vigilance factors.

Designing for Rare Situations

Traditionally, it was the operators' sole duty to deal not only with any unexpected behavior of the system part. This did not work: when dealing with rare events, the situation is typically different from that expected at design time. The problem with rare situations is that there are so many of them. Consequently, the designer cannot do anything to directly affect the operator's perception of unexpected situations, or to assist in the decision making. A method for controlling rare situation, proposed by Zonnenshain and Harel ([17]) assumes that mishaps are often associated with difficulties in adapting to the exceptional situation. This method is based on the System Theoretic Accident Model and Process (STAMP) paradigm by Leveson, proposing that the system should control its own behavior [8].

Rule-based Operation

The concept of rule-based operation is based on the idea of rule-base programming. The idea was introduced by Colmerauer and Roussel ([4]) and implemented in Prolog, a declarative programming language, based on first-order logic. The program logic is expressed in terms of relations, represented as facts and rules. This concept was adapted later and applied in Ergolight tools for usability testing. The adapted concept was of rules for identifying exceptional operation. These rules enabled detecting user errors by checking compliance of the operation with special constraints called Usability Problem Indicators (UPI) ([6, 7]).

It should be noted that while rule-based programming and testing was proposed for any software design and testing, the original STAMP paradigm was proposed for accident prevention ([8]). Yet, the STAMP approach is appropriate for any resilience-critical system.

Transdisciplinary Collaboration

Sillitto et al. ([14]) have distinguished interdisciplinary from transdisciplinary. Interdisciplinary has to do with applying multiple disciplines in parallel and independently, to accomplish a task. This method is appropriate when the design considerations in each of the discipline are defined based on the requirement documents, and they do not depend on the work being done in the other disciplines. When the design considerations of any of the disciplines depends on another discipline, the better way is the transdisciplinary approach, which "enables inputs and participation across technical and nontechnical stakeholder communities and facilitates a systemic way of addressing a challenge".

The Risks of Minor Flaws

If we look closely at the interaction of any system, in any domain, we may realize that they involve many problems, which we commonly disregard because the effect of each of them is not significant. We think about these problems only occasionally, when the costs are significantly high. Most of the time we accept the flaws because they are part of a deal: take it or leave it. So, we need to compromise: we take it, together with the interaction flaws.

The line between success and failure is sometimes very thin. The same problem of the Pressure Release Operated Valve (PORV) detected in an incidence of the Davis Basses nuclear power plant, when repeated in the TMI incidence, resulted in recession of the whole industry of nuclear power plants. Also, a mistake of selecting the wrong control, similar to that of the timer of a home appliance, was a source for the death of many pilots of the B-17 in WWII. It should be a design goal to make this line thicker. This goal may be reached by engineering. Engineering enables setting design rules and guidelines, with large margins for failure-free operation.

The Human Side of the Interaction

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

- The task view, in which we examine the ways people interact with the system
- The capability view, in which we examine physical and mental capabilities, motivation and limitation of the human operators, and their effect on performance and successful operation of the system.

Accordingly, 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 [5].

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.

According to the low-level interaction model, the behavior of the human operator is subject to three mental activities, namely situation perception, decision making and command execution, as described in the low-level HMI model above. The UCD toolbox offers several models and methodologies enabling to predict the behavior of operators in various operational conditions. The design based on these tools is about ideal behavior, in predicted situations. For example, the design may assume that the operators have access to the documents describing troubleshooting procedures. In practice, however, the operational condition, operational context, intentions, state of mind and personality might be different from the expected, and the operator's behavior might divert from the prediction. The operator's behavior depends on the information they have, but sometimes critical information is missing or misleading. Decision making based on the situation evaluation is biased by improper reasoning, due to vague, ambiguous task and goal setting, unsuited expectations, rationality and other personality traits of the human operator, and also by organizational pressures).

HSI ENGINEERING

The Engineering Chasm

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. A primary challenge of system design in the 4th IR is about the people experience in going through this change. Recently, usability practitioners discuss challenges of incorporating human factors in system development. 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 and usability practitioners need to demonstrate and explain to systems engineers how to integrate the theories of cognitive sciences in the system development.

Pitfalls of Traditional HMI Design

In a common 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.

The Engineering View of Errors

The term error is commonly used to divert the responsibility for the failure of complex systems from the stakeholders to the people who happened to operate the system ([4]). .

The engineering view, on the other hand, is very pragmatic. The idea is that engineers should prevent errors by design, for mistakes do not happen in a vacuum. Cars and aircrafts should be equipped with sensors and algorithms enabling detecting and evaluating the risks of collision, and a means for automated response, such as by activating an Automatic Breaking System (ABS). The source for the risky situation, whether it is the driving, or the driving condition, or any external hazard, is too complicated to handle, and is irrelevant to the solution. Therefore, analysis of the human factors involved in the situation is redundant.

The engineering view is probably the more natural and pragmatic approach to errors. When your house collapses, you do not call it an error. You just fix the design problem. In contrast with the cognitive and the community view, which attribute the errors to the operators, the engineering view attributes human errors to the design, which enables making the error.

When taking the engineering view, we may characterize system-level attributes of errors, such as information gaps, scenario ambiguity, mode errors, error-prone controls or automation biases. It is the design challenge, discussed in this book, to avoid the failure modes associated with these attributes.

HSI Thinking

The HSI approach to solving real world problems complies with the system approach defined by SEBoK as "a set of principles for applying systems thinking to engineered system contexts" ([1]). Following Richmond, with system thinking a system engineer "can see both the forest and the trees; one eye on each". Accordingly, we may consider two aspects of HSI thinking:

- The 'trees view' is the internal aspect, about the functional units integrated with the operators, collaboration between components of the engineered system, and
- The 'forest view' is the contextual aspect, about the interaction of the engineered system with the real world, namely, the customers and stakeholders, as well as the operational constraints.

Boy ([2]) suggested that system design should be from purpose to means, from outside-in. 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. The following chart presents a two layers model of HSI thinking:

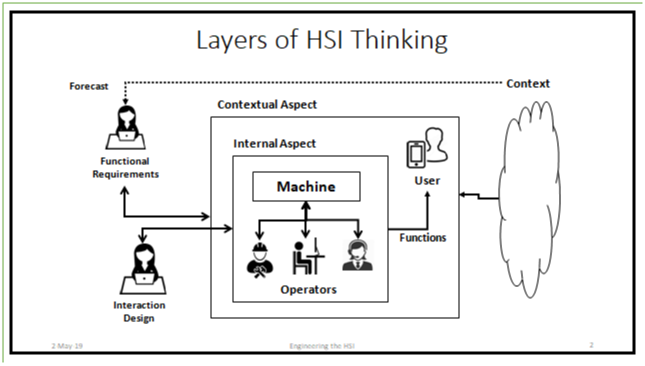


Figure 2 – layers of HSI thinking

Agile HSI Thinking

In the early days of systems engineering, 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. HSI thinking is a continuous process, integrated with agile development. The contextual aspect includes sensing the need to change the requirements and triggering the change. The internal aspect is the traditional response to changes, typical of agile development.

FROM THEORY TO PRACTICE

A High-level Model of HMI

As discussed above, the system performance is affected by attributes of success and failure. Although the rate of failure is typically low, the effect of failure may be high due to the high costs. Therefore, a useful model of HMI should include activities of routine operation and of operating in exceptional situations. The following chart describes a model of the transitions between the operational scenarios:

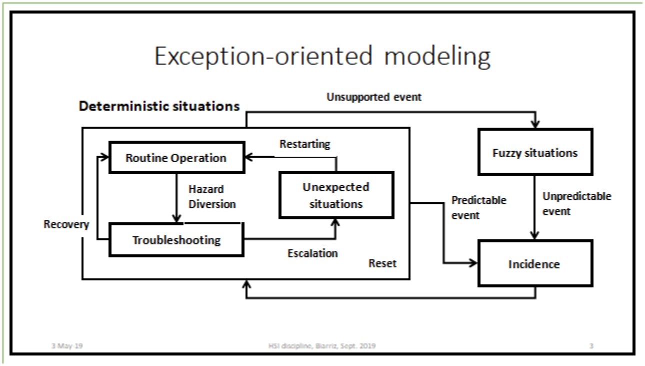


Figure 3 – exception-oriented modeling

The model shows that the interaction may be in normal most of the time, when the situation is deterministic. When the situation is fuzzy, the interaction is liable to end up in an incidence. When a situation is non-routine, and the operators do not know about it, the risks are especially high as explained below:

Interaction Chaos

In chapter 5 “To Err is Human” of his book “The Design of Everyday Things” (1988) Donald Norman, warned about the risks of chaos due to user mistakes ([9]). Chaos is a term we often use to express our perception and feeling about uncontrolled complexity. Interaction chaos has several basic forms:

- Situational confusion; fuzzy situations, due to under-specification of situation dependency, namely, missing conditions, such as about activity availability depending on the active scenario
- Situational errors, due to overloading several actions on the same control
- Over-automation, when the machine might prohibit necessary operator’s intervention
- Under-automation, when the operators might fail to manage a exceptional situation
- Missing information, when the operators do not have the information required to become aware of an exception, or for deciding how to act, for example, in response to component failure
- Information ambiguity, when the information presented to the operators is unclear or confusing, for example, when the operators do not know how to respond to an alarm.

In addition, an interaction chaos may involve a combination of the basic forms.

Exceptions

A machine situation not included in the scope of routine operation may be regarded as exceptional, deserving special means to prevent, detect and recover from. Sources of exceptions that should be considered in the interaction design include:

- Sync delay (shortcuts, smart selection, popups,)
- Information ambiguity (such as in control selection)
- Chaos (such as of scenario confusion)
- Diversion (such as component failure)
- Mode confusion (assuming the wrong mode)
- Coordination failure (such as due to unit reset)
- Over-automation (such as setting default values)
- Under-automation (such as synchronization delay)
- Specification, design and implementation mistakes (bugs)

Latent Exceptions

A latent exception is an exceptional situation of which the operators are not aware. A pseudo routine situation is a latent exception. Also, if the machine provides an indication about the exception, but the operators does notice it, the exception may be regarded as latent. The risks of latent exceptions are that the operators are not aware of them.

The machine can handle automatically certain exceptions, such as operator slips. However, the machine cannot handle many other exceptions, such as a component failure or inconsistent system state.

By default, following a diversion from routine operation, the exceptional situation is latent, which means that by default, the operators are not aware of the exception. These might remain latent if the design does not provide a proper indication about it. The operators may become aware of the exception only if the machine advertises it. Only then the operators can be aware of it.

Latent Operational Modes

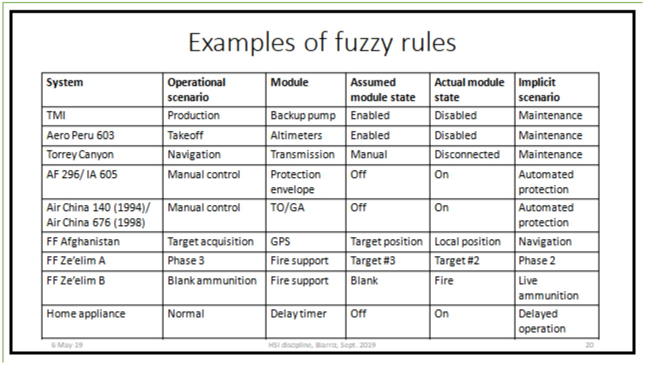
This source of mode errors is encountered when the system does not display the operational mode, or when the operators do not notice it, because it is not visible, or audible, or because the operators are in stress, etc. An example is of mode setting by default. If a mode is set by default, and the system does notify the operators about the change, they might actuate the wrong function

The Costs of Late Detection

To enable the operators to resolve a problematic situation, the exception should be detected as early as it is generated. Unfortunately, when the hazard detection is employing probe-based, the time elapsed from generation to detection of the exceptional situation might not leave sufficient time for the operators to handle the situation. An example is of disabled backup utility. The exceptional situation of the backup pump in TMI remained latent until the pump was called to push coolant from the backup container. This was too late to enable the immediate cooling required.

Fuzzy Situations

Ideally, all the system requirements are specified in the requirement document. Practically, much of the requirements are implicit. People agree on the requirements, but do not bother to write down all the details, because they are tedious, and they are obvious. The problem with the implicit requirements is that at run time, the operators might miss critical details, such as events of scenario change. The following table presents a list of celebrated accidents due to fuzzy situations:



Examples of fuzzy rules

System	Operational scenario	Module	Assumed module state	Actual module state	Implicit scenario
TMI	Production	Backup pump	Enabled	Disabled	Maintenance
Aero Peru 603	Takeoff	Altimeters	Enabled	Disabled	Maintenance
Torrey Canyon	Navigation	Transmission	Manual	Disconnected	Maintenance
AF 296/ IA 605	Manual control	Protection envelope	Off	On	Automated protection
Air China 140 (1994)/ Air China 676 (1998)	Manual control	TO/GA	Off	On	Automated protection
FF Afghanistan	Target acquisition	GPS	Target position	Local position	Navigation
FF Ze'elim A	Phase 3	Fire support	Target #3	Target #2	Phase 2
FF Ze'elim B	Blank ammunition	Fire support	Blank	Fire	Live ammunition
Home appliance	Normal	Delay timer	Off	On	Delayed operation

6 May 19 HSI discipline, Biarriz, Sept. 2019 20

Table 1 – examples of fuzzy rules

In the example of unintentional delay setting above, the design may avoid the risks of user errors if different operation procedures were defined for two scenarios: normal operation and delayed operation. The design could not avoid the risks, because the two scenarios were not reflected in the system design. They were defined implicitly, enabling the user to make the mistake. The design could prevent the wrong control selection if the scenario was defined explicitly, and delay control was disabled in normal use of the appliance. The implication of this observation to system design is that operational scenarios should be defined explicitly, and that all the system units should share the same scenario

Goal Setting

Traditionally, the primary goals of UCD are to leverage productivity, performance, safety, user satisfaction, enjoyment, etc. When transforming these properties to the mental activities described by the low-level interaction model, the design goals may be stated as: facilitating reliable perception, decision support and protection from execution errors. The challenge of HSI engineering 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 the system goals
- Key factors affecting the system behavior cannot be controlled automatically, by design. These factors are marked as clouds in the chart above.
- Poor validity, as there is no practical way to measure these, or to estimate the direction or magnitude of their effect.

The implication of this discussion is that the low-level HMI model may be upgraded, to highlight the role of goals and performance, and to suggest on ways to protect from failure by automation.

Design Highlights

The discipline of HSI engineering may involve changes in the following aspects of the HMI design.

Timing: traditionally, human factors are added ad-hoc to the system design. This is too late. It is the responsibility of systems engineers to integrate human factors in the stage or system analysis and requirements specification.

Time span: traditionally, usability considerations focus on the stages of marketing and initial operation. It is essential to extend the scope of usability assurance to the whole life cycle.

Automation control: a main consideration in HMI design is the balance between automation and human control. The new discipline will propose guidelines for collaboration design, optimizes for maximal performance and minimal risks.

Failure analysis: Traditionally, failure prevention is based on root-cause analysis. Such analysis does not support coping with the unexpected and proposes developing rule-based protection. Rebounding from operator's slip should be integrated in the system design. The new discipline proposes that applying new methodologies for structured rebounding.

Error tolerance: a common practice in system design is to apply means for fault tolerance. Traditionally, systems engineers do not apply such means for protection from operator's errors. The new discipline proposes applying a model and means for preventing operator's errors.

Extended exception handling: traditionally, interaction design focuses on procedures of normal operation. However, system failure involves difficulties in operating in exceptional situations. Applying golden rules applicable to normal operation, as proposed by Shneiderman ([13]), might hamper the interaction in exceptional situations. It is about time to expand the HMI design practices, such as of deciding on interaction styles, to also support exception detection, troubleshooting, recovery and emergency operation.

Modeling the HMI: a common practice for UI design is in terms of event-response. The new discipline considers typical sources of unexpected diversion, advocates scenario-based interaction styles and applies rule-based procedure-oriented definition.

Human-machine collaboration: the new discipline proposes a new model of human-machine collaboration, enabling to cope with the exceptions. Also, it proposes that the implementation should be based on protocols describing proper interaction, which will enable diversion detection. Special safe-mode operational procedures are essential to deal with the unexpected in emergency.

Situation awareness: Norman noted that a primary source for system failure is the lack of information required for situation awareness ([10]). A key related problem is of attention distraction due information overload, and the role of nuisance alarms. The new discipline proposes to develop a means for assessing the effect of various S/N ratio of notifications and alerts.

Incidence investigation: traditionally, system failure is attributed to the operator. The new discipline encourages radical changes in system thinking, to mitigate the risk of common biases in interaction design and to enable learning from mishaps. These changes should be accompanied by technological advances, including activity trackers and analyzers, based on data mining technology.

Glossary: various industry domains use specific terms for common attributes of HMI. The new discipline propose a glossary that may enable engineers of the different domains speak using the same language.

Formalizing the Machine Behavior

To implement the concept of self control, we need to define the data that the situation analyzer will use to decide whether the situation is routine or exceptional. The data used by the situation analyzer is a representation of the machine behavior, in terms of situation changes. Expected behavior may be defined by protocols of operational procedures, and of inter-unit compliance. For example, a protocol for operating a delay feature of a home appliance may include a condition that this procedure is available only when the machine assumes operating in a normal scenario. Also, a protocol for compliance between a safety-critical unit and a safety feature, such as a backup unit, may be that both units share the same scenario.

The machine situation may be formalized as a collection of the active states of state machines, each of which represents a protocol. Critical components may be associated with primary state machines, representing their availability, On/Off condition, Enabled/Disabled condition, and functional state.

Design Practices

Practices for exception prevention

Practices for exception prevention include:

- User centered design (UCD)
- Applying fuzzy logic for slip and delay detection
- Applying STAMP principle of self control
- Standard protocols obtained from standard operational rules
- Formalizing the system activity in terms of state machines
- Ongoing risk information, obtained by risk indicators
- Rules for balancing automation with manual control

Practices for exception detection

Risk Indicators: To detect risky situations, the machine may be equipped with sensors. For example, each component may be equipped with a sensor indicating its primary states.

Indirect risk indicators: Additional risk indicators may be applied to warn about hazards and threat not detected by direct sensors of component primary states.

Escalation indicators: special risk indicators, based on auxiliary sensors. For example, a thermometer may indicate hazards of uncontrolled overheating, and threats of extreme temperatures.

UPIs: special risk indicator used to detect situations of the operator's confusion ([6]). Statistical UPIs enable detection or usability flaws in website design ([7]). Such indicators were demonstrated in Ergolight toolkit for usability testing.

Practices of exception handling

Traditional exception handlers used in software design are of limited effect in HMI design. The essentials of HMI-oriented exception handling include:

Exception detection

- Expected exceptions, by root-cause analysis
- Unexpected exceptions, by searching the design scope
- Risk indicators
- Detectability, by implementing the scenarios and states

Alarm control

- By statistics of alarm S/N

Troubleshooting

- By dynamic root-cause analysis

Integration

The classical HMI model presented above describes short event-response sessions. It is low level. It does not show how the machine and the operators are engaged in solving high-level issues, such as human-machine control allocation, scenario sharing, detecting deviations from routine operation, troubleshooting, detecting unexpected situations etc. The following figure illustrates a way to implement the STAMP paradigm.

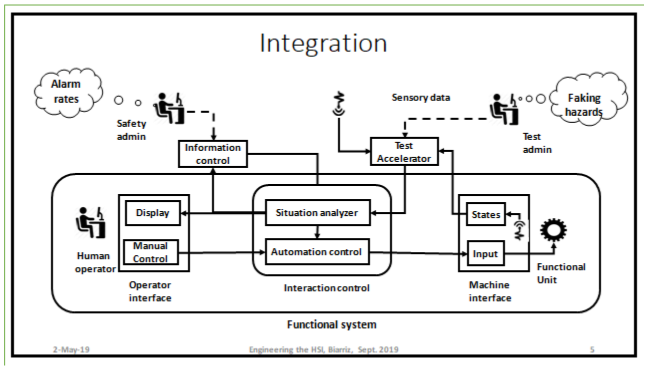


Figure 4 – integration

Human-Machine Collaboration (HMC)

The HMC 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 based on the classical model (above), are controlled in implementations based on the new model.

The Situation Analyzer

The situation analyzer traces the situation changes, based on sensory data received from the system, the operator and the environment. The new situation is tested with respect to a model of normal machine behavior, and classified as routine, exceptional or unexpected. The analyzer informs the operators about the non-routine activity, and sets the HMI style according to the category of the situation.

The situation analyzer has two features:

- It checks the compliance of the machine situation changes, as represented in the shadow, and provides warning in case of protocol violation.
- It checks the compliance of the operator's action with the shadow, according to the protocols, provides preview information to the operators about the potential effect of their action, and provides notifications and warnings in case of protocol violation.

For example, in the case study of the unintentional delay of a home appliance, the system may operate in two scenarios: normal operation and delayed operation. The delay timer may be available in the delayed operation, but not in the normal operation. The situation analyzer has this rule stored in the knowledge base. Accordingly, the analyzer may disable the delay control when operating in normal operation, or at least warn the users when they try the delay control in normal operation.

The Automation Controller

An automation controller unit monitors 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. It receives information from the situation analyzer about the next activity, and transforms the information to actionable information for the operators and the machine. When the situation received from the analyzer is of a routine scenario, the automation controller allocates the control to human operator and to the machine based on predefined rules, enabling the human operators to supervise the execution by the machine. When the situation is exceptional, the machine may have a more active role, providing warnings and indications of the system situation. In emergency, the machine may have an even more active role, initiating emergency evacuation procedures, and disabling risky operator's activity, by rules of safe-mode operation.

The Operator Interface (OI)

The OI may consist of dedicated control stations, used for distinct operator's roles and situations. In a recommended architecture, different stations to are assigned to distinct operational tasks:

- Primary station - for managing the interaction in routine operational conditions
- Recovery station - for managing the interaction in exceptional operational conditions. The operator's primary task is troubleshooting
- Rescue station - for managing the interaction in emergency, due to unexpected operational conditions
- Supervision station - for managing the interaction when performing the top-level tasks, and for managing the interaction during hazard detection

Because the mental mode is not the same for the different roles, the interaction style should also not be the same. For example, in the design of the primary and the supervision station we may assume that the operators may be trained to learn the access to the various features, and to respond properly to events. Such assumption is not valid for recovery and rescue stations, in which we should assume that the operators did not see the interfaces beforehand, and they need the machine support for learning how to respond and behave. Another example is the special attention to the possibility of tunnel vision in the design of the rescue station, because people are more prone to this effect in emergency.

Testability of exceptional situations

It is unlikely that in testing normal operation we may encounter all critical exceptional situations, unless we fake these situations.

The validation procedures applied in traditional SI sessions do not target the interaction between the users and the system. Subsequent to the SI, the validation of the interaction between the users and the machine are conducted in special sessions of task-oriented usability testing, practiced by usability professionals, applying practices which are traditionally out of the scope of SE. These practices are most effective in validating routine operational procedures, but they are short of detecting rare events, and of identifying unexpected situations. In the framework of HSI, the validation of the interaction between the human operators and the machine modules (HMI) is embedded in the SI procedures, enabling validating also rare events and unexpected situations.

The architecture used for the HMID should include means to fake and control exceptions efficiently, including a dedicated test station and means to bypass the actual machine situation. As an integral part of the project, a special interaction control unit should be developed, to enable simulation of rare event, such as hardware fault, for the validation testing.

Alarm Control

A primary source of operational errors is due to improper adjusting the thresholds that determine the transition to notification and to alarming. Special statistics may be employed to evaluate the rate of nuisance, and to compare the risks of nuisance with those of missed alarms. Special means, including an alarm control, may enable evaluating the risks and adjusting these thresholds.

The New Discipline

The discipline of HSI 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 way systems engineers implement their part in the collaboration is by ‘system thinking’. The way usability practitioners implement their part is by methodologies of UCD. The following chart illustrates the location of the new discipline of interaction design as a mediator between SE and UCD:

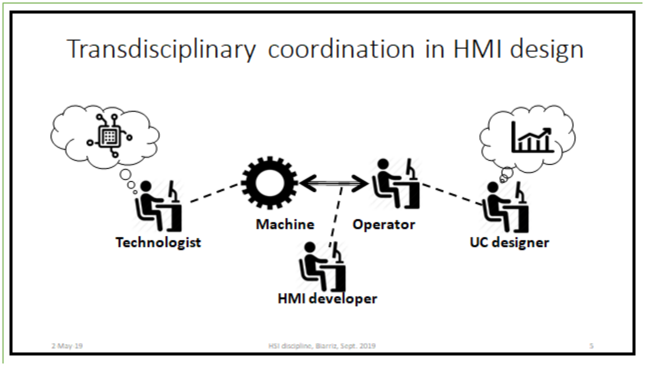

Transdisciplinary coordination in HMI design
Technologist
Machine
Operator
UC designer
HMI developer
2 May 19
HSI discipline, Biarritz, Sept. 2019
5

Figure 5 – transdisciplinary coordination in HMI design

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.

HSI development

HSI development includes special activities added to traditional system development. A waterfall model describing HSI development is depicted in the following chart:

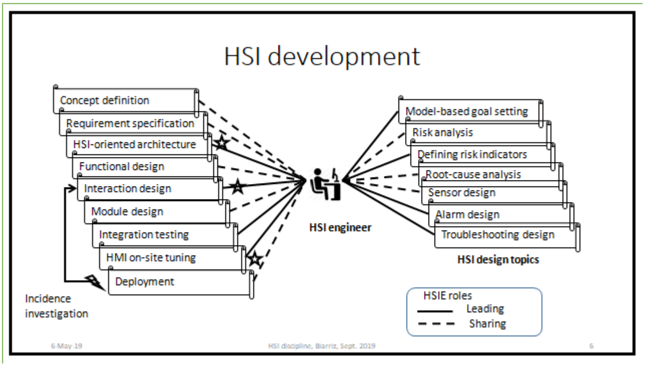


Figure 6 – HSI development

The left side of the chart is an enhanced version of the traditional waterfall model. The major enhancements are about the system architecture, the interaction design and the integration testing. The right side of the chart is a waterfall of design topics in HSI, across the various development stages. Most notable are the model-based goal setting, alarm design and troubleshooting design.

The Roles of HSI Engineers

The job of the HSI engineer is to manage the design and testing of the operational procedures used to handle all possible situations and events. The roles of HSI engineers are depicted in chart above. The solid lines refer to major responsibilities of the HSI engineer.

The HSI engineer is a coordinator between the technologists, the usability practitioners and the software engineers who implement the programs. Specifically, the HSI engineers are in charge of deciding what sensors should be installed to inform the system about its state. Also, the HSI engineers need to supply the usability practitioners with the information they need to have in order to be aware of the situation.

In order to enable the design and testing of the HSI in a project the HSI engineer may need to do the following:

- At the stage of defining the design concept, set the design goals, including performance and reliability attributes, based on the HMI model.
- At the stage of requirements specification,
    - Define the operational scenarios and the machine states in the system documents, and represent them in the project knowledge base, used for the situation analysis
    - Specify the requirements for integrating the knowledge base in order to implement the situation analyzer
- At the top-level design of the system normal behavior, define the actions applicable to the different system states, and represent them in the project database

- At the stage of failure analysis, specify the various expected deviations from normal operational conditions in the system documents, including the maximal response time that will be used as thresholds for safe recovery, and represent them in the project database
- At the stage of interaction design, specify:
    - The human-machine task allocation in the various scenarios
    - The sensors available in the project, what they measure, and their association with the various expected deviations, including virtual sensors, namely, computed measures used for indirect deviation detection
- At the stage of exception definition, specify the alert and notification thresholds, and the safety thresholds for the sensory data, including:
    - Attributes, such as modality, intensity, desired effect for the various addressees
    - Preferred/default operators' response for each of the alerts/ notifications
    - Define procedures for optimizing the thresholds used in the various risk indicators
    - Secondary risks of failure of the alarm system
    - Procedures for identifying the secondary diversions
- At the stage of test design, specify:
    - The requirements from a test controller, a special unit used to simulate rare events, such as component faults, for the testing
    - The method for optimizing the thresholds used in risk indicators.

CONCLUSIONS

HSI engineering is primarily about HMI design and testing. In order to ensure long-term high performance, the design should focus on enforcing operational reliability. A main conclusion from the complexity and variety of related considerations and methods is that in the 4th industrial revolution the theory of HSI engineering should evolve to a sub discipline of systems engineering. The 4th industrial revolution may involve various shifts towards HMI, associated with technology, methodology and HMI thinking and practices. These shifts may affect the people productivity, quality of life and safety. The new discipline may be based on scientific foundations, which may require, high degree in universities. The discussion above suggests that the focus of these studies will be on HMI design, testing and optimization.

REFERENCES

1. BKCASE Editorial Board. 2017. *The Guide to the Systems Engineering Body of Knowledge (SEBoK)*, v. 1.9.1 R.J. Cloutier (Editor in Chief). Hoboken, NJ: The Trustees of the Stevens Institute of Technology. Accessed DATE. www.sebokwiki.org. BKCASE is managed and maintained by the Stevens Institute of Technology Systems Engineering Research Center, the International Council on Systems Engineering, and the Institute of Electrical and Electronics Engineers Computer Society.
2. Boy, G.A. *Orchestrating Human-Centered Design*. New York: Springer, 2013. ISBN 978-1-4471-4338-3
3. Colmerauer, A. & Roussel, P., The birth of Prolog (PDF). *ACM SIGPLAN Notices. 28* (3): 37. 1993, doi:10.1145/155360.155362
4. Dekker, S. *Just Culture: Balancing Safety and Accountability*. CRC Press, ISBN 9781409440604, 2007,
5. Dockrill, P. WATCH: F-16 Autopilot System Saves The Life of an Unconscious Fighter Pilot, *Science Alert, 14* Sept. 2016.
6. Harel, A., 1999. Automatic Operation Logging and Usability Validation, *Proceedings of HCI International '99*, Munich, Germany, Vol. 1, pp. 1128-1133
7. Harel, A., Kenett, R.S. and Ruggeri, F., Modeling Web Usability Diagnostics on the basis of Usage Statistics, in *Statistical Methods in eCommerce Research*, ed. W. Jank and G. Shmueli, Wiley, 2008, pp. 131—172.
8. Leveson, N.G. A New Accident Model for Engineering Safer Systems, *Safety Science, 2004, Vol. 42*, No. 4, pp. 237-270.
9. Norman, D.A., *The Design of Everyday Things, 1988*, ISBN 978-0-465-06710-7
10. Norman, D.A. The "problem" of automation: Inappropriate feedback and interaction, not "over-automation". In D. E. Broadbent, A. Baddeley & J. T. Reason (Eds.), *Human factors in hazardous situations* (pp. 585-593). Oxford: Oxford University Press, 1990
11. Norman, D.A. & Draper, S.W. *User Centered System Design; New Perspectives on Human-Computer Interaction*, L. Erlbaum Associates Inc. Hillsdale, NJ, USA, 1986 ISBN:0898597811
12. Richmond, B. Systems Dynamics/Systems Thinking: Let's Just Get On With It. In, *International Systems Dynamics Conference*. Sterling, Scotland, 1994.
13. Shneiderman, B. *Designing the user interface: strategies for effective human-computer interaction*, Reading, MA: Addison-Wesley, 1987.
14. Sillitto, H.G., Martin J., Griego R., McKinney, D., Arnold, E., Godfrey, P., Dori, D., Krob, D. and Jackson, S. Envisioning Systems Engineering as a Transdisciplinary Venture. *IS 2018*, Washington, DC.
15. Standish Group. *The COMPASS report*, Forbes, 1995.
16. Weinberg, G.M., 1971. The *Psychology* of *Computer Programming*. New York, Dorset House Publishing.
17. Zonnenshain, A. & Harel, A. *INCOSE Annual International Symposium*, Seattle, 2015