Model-based Utility Optimization

A Universal Model of System Integration

ABSTRACT

The article presents a universal model for maximizing the system's utility. The operational utility of software depends on the performance, constrained by the performance envelope, which is defined by operational rules. Barriers to maximizing the operational utility include incidences of invading the performance envelope due to integration complexity. The integration complexity may be defined by that of the operational rules, in terms of scenarios. Reducing the integration complexity is critical to enabling carefree operation, and to preventing accidents. Root-cause analyses of reported incidences indicate that often the integration rules were not defined formally in the system design, resulting in uncoordinated activity. The model presented here is based on a formal definition of the operational conditions in terms of scenarios. Embedding model-based digital twins in the system integration may enable operational risk preview, early identification of incidences, and integrated troubleshooting. The universal model proposed here is based on formalizing the operational situations, activity, and behavior, and on a formal approach to exception management. The first version of this model was published in 2015, comprising various Generic Mini Models (GMM) used for formalizing the definition of system resilience. The first version was validated by a semi-quantitative analysis of 67 incidences. The GMMs in the new version are arranged in seven layers: structural, functional, performance, situational, activity, behavioral, and resilience. The article calls for developing and integrating tools for tracking, recording, and analysis of the integration in upcoming projects, which may enable assessment of the actual costs of incidences, as well as validation of the universal model proposed here.

CCS CONCEPTS

• Software design engineering • Interaction design • Architectures

KEYWORDS

Software engineering, Systems integration, Incidences, Error prevention, Safety, Formal methods, Scenario-based modeling

ACM Reference format:

1 Utility Optimization

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

1.1 Operational Performance

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

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

1.2 The Performance Envelop

Systems are designed to optimize performance when operating in nominal conditions. Hence, by definition, when diverting from the optimal conditions, the performance should reduce. Moreover, in extreme conditions, the operation might fail, a situation called 'incidence' (Hollnagel, 2006) [10]. The term 'performance envelope' refers to the limits of the performance, namely, the situations in which the operation results in incidences. An incidence is an event of invading the performance envelope.

The concept of performance envelope extends that of flight protection envelope, imposed to protect aircrafts in extreme conditions. It is an expression of the performance reduction due to constraining the operation to avoid incidences. For example, the speed of an airplane is limited by the stall threat and the Mach number, and the altitude is limited by the Coffin Corner (Swatton, 2011)[13].

1.3 Operational Utility

The operational utility is affected not only by the design features, but also by the constraints, defined by the performance envelope. The utility is an expression of the

actual performance, when avoiding the risks of invading the performance envelope, as illustrated in the following figure:

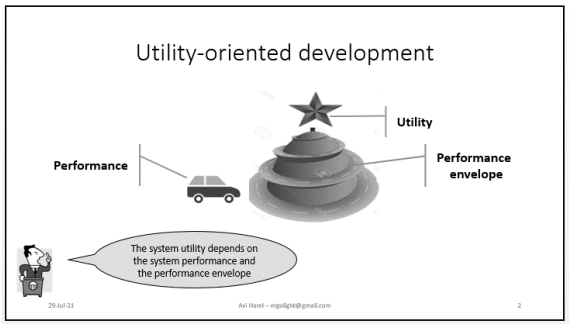

Figure. 1: Utility-oriented development

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

1.4 The Costs of Incidences

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

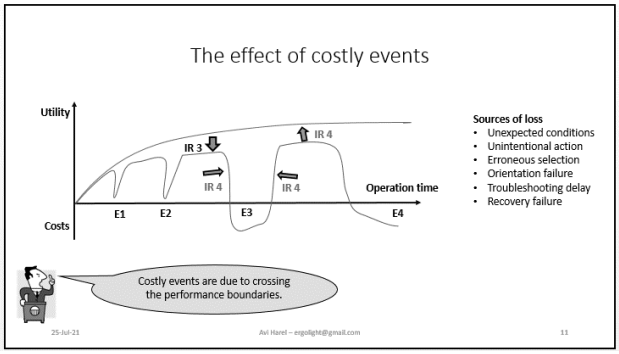

Figure 2: The effect of costly events

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

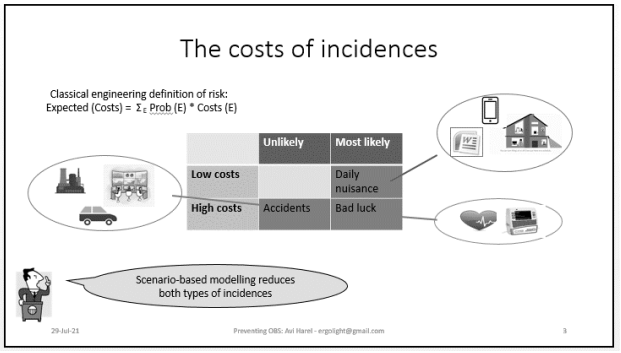

Figure 3: Risks of operational complexity

Taleb (2007)[14] argued that it is impossible to predict incidences due to Black Swans (rare events) because a-priori we do not have the data required for the prediction. Explicit incidences are those that in hindsight turn out to be costly.

1.5 Integration challenges

Accident investigations indicate that mishaps involve two key factors: triggers and failure enablers. Triggers are typically obvious, and there are methodologies, such as STAMP, for disabling risky events. Human-generated triggers are tackled by applying Human-Centered Design (HCD) practices. The new framework of Human-System Integration (HSI) extends both the traditional System Integration and the HCD perspectives of triggers. The topic of disabling risky behavior emerges from recent studies on system resilience.

It turns out that most failure enablers involve either inter-unit coordination problems, or operating in exceptional situations. It turns out that these failure modes may be eliminated by formalizing the concept of scenarios, and the ways the system may behave in various scenarios.

1.6 The scenario challenge

The dictionaries define “scenario” as “a sequence of events, especially when imagined”. In the context of system integration, the term refers to events that should be expected during the system operation. In HCD, scenarios define the context of operation, namely, the operational tasks and risks that should affect the operator’s behavior. In the framework of system integration, scenarios enable facilitating the definition of error-prone situations and activity.

Analysis of the reports of several accidents, used as case studies, indicate that the scenarios of the error conditions were not defined formally in the requirements documentation, and therefore, the system did not have the infrastructure required to account for their effect.

The scenario challenge is to formalize the scenarios, and to develop the infrastructure that may proper operation in the proper scenario. The design goal is to facilitate the inter-unit coordination, and to mitigate the risks of any activity in undesired situations.

1.7 The coordination challenge

Analysis of the reports of several accidents, used as case studies, indicate that they involve uncoordinated activities. The case studies examined include:

- Synchronization errors, such as in the Therac 25 accidents
- Several friendly fire accidents, due to wrong perception of the operational scenario.

In terms of scenarios, this means that various sub systems assumed operating in different scenarios.

1.8 The control challenge

Analysis of the reports of several accidents, used as case studies, indicate that they involve wrong assumption about the active scenario. The case studies examined include:

- Disabling safety-critical sub systems in functional operation
- Enabling risky features, designed to use in maintenance and setup, while in functional operation

2 Model-based System Integration (MBSI)

Barriers to seamless operation include instances of confusion and hesitation of the operators, due to anxiety about potential loss. Typically, the confusion is attributed to operator's errors. Root-cause analysis of operator's errors indicates that often they result from uncoordinated activity or from overriding implicit interaction rules. Model-bases systems integration is critical to enabling agile development, essential for reducing the development costs.

The methodology of model-based engineering is inspired by a similar methodology of rapid prototyping, developed in the framework of software engineering in the 70s (Grimm, 1998)[3].

MBSI is the modern version of software prototyping, a concept explored in the 80s Model-based system integration (MBSI) enables early integration by simulation, resulting in shortening the integration phase and reducing the development costs. (Luqi, 1989)[11]. MBSI may consist of project-specific, functional features, as well as universal features applicable to maintenance, resilience, training, etc. The universal features may apply to various domains and industries.

2.1 Modelling the System Integration

Hollnagel (2006)[10] proposed two ways for modeling the system integration. The proactive approach is about how to describe normal behavior, and the reactive approach is about how to describe extreme events. The need to present formal solutions was elaborated elsewhere. The means to avoid exceptional situations and to support exception management may be integrated into the model used for the interaction design. A method and procedure for modeling the system integration was described earlier. The procedure has seven steps, according to a sever-layer model of the integration. Integration modeling is a hybrid approach, in which we define normal behavior proactively, and we apply learning from failure. The reactive part is by serendipitous learning from incidences (e.g. Copeland, 2020)[2].

2.2 Scenario Models

Scenario-based design facilitates the coordination between the system elements and enables enforcing operation by the rules. The focus is on generic models intended to optimize the system value. It is essential for enabling seamless, carefree operation.

A key design goal is to enforce operating according to the rules. Scenario-based modelling (SBM) is a procedure of activity design, in which the system activity is expressed in terms of operational scenarios. The objective of SBM is to support the design of seamless, robust, coordinated interaction by the rules. Scenario-based design facilitates rule-based coordination between the system elements.

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

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

be broken down to sub scenarios such as: low risk, high risk, and emergency.

Often, the lower levels are mostly domain specific. For example, the functional scenario of a commercial airplane may own three primary sub scenarios: takeoff, navigation, and landing. Further down the tree, the navigation scenario may own two sub-scenarios: manual navigation and automatic navigation. The bottom level may comprise project specific scenarios. Component-level scenarios may be described by simple state trees, representing states about availability, reliability, activation, performance level, etc.

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

2.3 Normative Interaction Models

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

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

2.4 Model Realization

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

2.5 Situational Models

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

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

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

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

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

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

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

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

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

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

2.6 Situational Rules

Situational models enable structuring a framework of operational rules. According the principles of cybernetics, adopted for the STAMP paradigm, systems should operate according to rules. Many incidences may be attributed to

ambiguous, implicit operational rule. For example, the rules defining the properness of the operation of the elementary system are derived from the situational models of the Operative and Idle scenarios. If these rules are implicit, then the system might not be detect exceptional situations, such as when the switch is Off and the device is On.

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

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

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

Examples of generic rules:

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

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

2.7 Rule-based Exceptional Handling

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

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

2.8 Unexpected Situations

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

2.9 Activity Models

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

2.10 Activity Rules

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

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

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

2.11 Protective Rules

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

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

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

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

2.12 Activity Protocols

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

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

2.13 Transition Synchronization

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

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

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

2.14 Transition Models

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

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

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

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

2.15 Recovery Models

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

3 Engineering

Following Leveson's STAMP, the engineering of the system integration should be based on rules of normal behavior. Exceptions may be defined by exclusion from normal behavior. Engineering the scenario-based models may comprise the following topics:

3.1 State integration charts

The common practice of software design employs UML statecharts for graphical representation of state transitions. This kind of representation is not adequate for modeling the

integration of state machines. The problem is that events designed using UML statecharts are error-prone. The integration version of statecharts supports describing various attributes of mutual effects between state machines, as well as enforcing error-free transient inter-unit state transitions.

3.2 State Activity Diagrams (SAD)

A special variant of integration charts deals with inter-state transition. This term refers to the effect of changes in one state machine on others. The following diagram illustrates an activity model of an elementary system, consisting of a simple device and a power switch controlling the device activity:

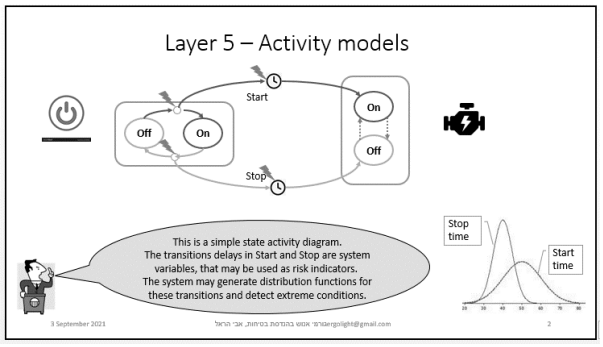

Figure 4: The State Activity Diagram of an elementary system

Exceptional values of the Start and the Stop may indicate exceptional conditions, such as due to component failure.

3.3 Error-proofing

Primary barriers to maximizing the utility are limitations of operating in exceptional situations, typically attributed to errors. These barriers result in hampering the system's usability. The root cause for many operator's errors, such as in using consumer products, is due to erroneous activation of feature that should be available in different scenarios. For example, a most prominent problem in operating home appliance is the unintentional activation of setting features. This failure mode is the source of several famous accidents, such as the B-17 accidents due to control substitution in WW II, and the Torrey Canyon supertanker crash in 1967. The scenario model may serve for designing the screens and panels, to prevent erroneous activation of features that do not comply with the active scenario.

3.4 Evaluation

For evaluating the model, we may employ the Layer Of Protection Analysis (LOPA) technique, commonly used in the process industry for assessing the protection needs. The evaluation is based on testing the effects of protection layers and calculating the potential risks (Baybutt, 2002)[1].

3.5 Infrastructure

Utility-critical systems should incorporate means, including sensors, trackers, recorders, and analyzers, for informing the operators and the developers about the time they could save. The infrastructure for model-based HSI may include special means intended to save the time wasted in handling exceptional situations. The means to avoid exceptional situations and to support exception management may be integrated into the model used to design the HSI. For example, they may include model interpreters that enable customizing the model transition to software units.

3.6 Data analytics

Tracking tools enable capturing and measuring the costs of daily, low-cost events. In prior studies it was found that data analytics may be applied in usability testing by problem indicators. Universal tracking is crucial also for enabling learning from rare events.

3.7 Digital twins

A digital twin is an executable virtual model of a physical thing or system (Wright & Davidson, 2020)[16]. The concept of digital twins is based on the concept of virtual prototyping, dated in the 80s, in which a model was used to replace system units by emulation. This feature enables early integration, by using virtual units instead of the real components that are not ready yet for the integration. This feature was recently adopted for systems engineering in the form of digital twins.

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

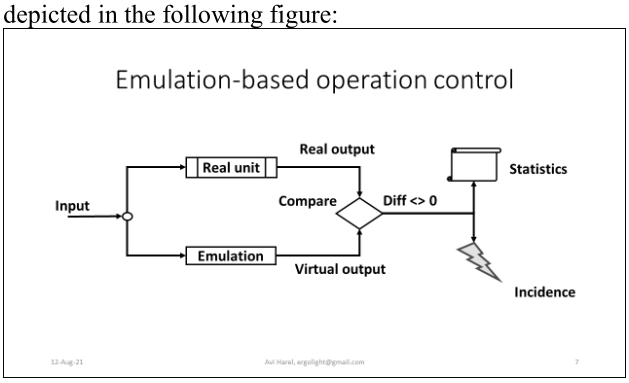

Figure 5: Digital twins used for incidence detection

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

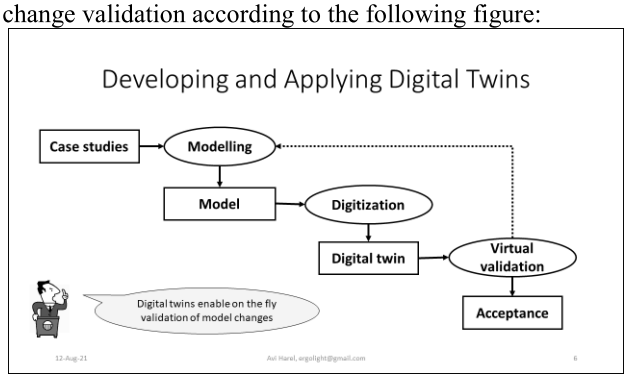

Figure 6: The role of digital twins in model based HSI

3.8 Customizing

The seven-layer model mentioned earlier is generic, applicable to various domains and industries. To adapt it to a particular project these models need customization. The customization process is according to the layers; each layer depends on the previous one.

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

3.9 Parameter development

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

3.10 Model development

Models enable saving development costs by enforcing seamless adaptation to design changes. The models should be defined iteratively; each cycle is followed by evaluation. Typically, the evaluation ends up with a list of requirements for design changes, intended to reduce the operational complexity. The development might end when it is obvious that all known significant risks are removed. Criteria form ending the development may be based on the Service Integrity Level (SIL) evaluation method commonly applied in the process industry (Redmill, 2000)[12]. Model-based integration is essential for seamless change validation based on digital twins.

3.11 Testability

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

3.12 Adjustability

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

3.13 Model Development

Earlier studies demonstrated that system integration may be based on models consisting of rules defining normal situations and activity, which may apply across various industries and domains. Typical activities in model development include:

- **Evaluation**. For evaluating the model, we may employ the Layer Of Protection Analysis (LOPA) technique, commonly used in the process industry for assessing the protection needs. The evaluation is based on testing the effects of protection layers and calculating the potential risks. Trackers should be developed and integrated in systems, to enable evaluating the effectiveness of this methodology.
- **Learning**. Earlier studies demonstrated the potential benefits of tools for tracking and analysis of the system activity. Utility-oriented engineering may incorporate means, including sensors, trackers, recorders, and analyzers, for informing the operators and the developers about the time wasted in incidences.
- **Agile development**. The models should be defined iteratively; each cycle is followed by evaluation. Typically, the evaluation ends up with a list of requirements for design changes, intended to reduce the operational complexity. The development might end when it is obvious that all known significant risks are removed. Criteria form ending the development may be based on the Service Integrity Level (SIL) evaluation method commonly applied in the process industry.

- **Testability-oriented design**. Testing rare events is challenging. To enable testing exceptions the system should incorporate a special tester unit that fakes various kinds of faults, in various conditions, that the testing team can customize. A special scenario should be defined, which is part of the operational conditions. Utility-critical systems should incorporate means, including sensors and data analytics, for informing the operators and the developers about the time they could save.

4 Conclusions

A challenge for the 4th industrial revolution is to develop a methodology for cross-industry model-based integration. This study demonstrates that we can define universal rules, suggesting that this goal may be reached: principles of integration may be phrased as scenario-based rules, which may be transformed to protocols for risk detection, recognition, and identification. The article explores various protection patterns, but certainly not for all possible design challenges. It may be interesting to explore operational rules for various tasks in various domains. Validation of the models proposed here may be conducted by analysis of the system performance, obtained by activity trackers, using statistical metrics, followed by traditional usability testing in the corresponding scenarios.

ACKNOWLEDGMENTS

REFERENCES

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

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

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

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

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

[12] Redmill, F 2000, Understanding the use, misuse and abuse of safety integrity levels, Proceedings of the Eighth Safety-critical Systems

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

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

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