Risk Assessment to Support Liability Allocation Performed by the System GUI Analysis

R. Cassino, A. Vozella, G. Gigante and D. Pascarella

Abstract Main productive, administrative and social organizations represent interconnected socio-technical systems, namely, complex systems constituted by technical artifacts, social artifacts and humans. The Graphical User Interface (GUI) is a system interaction approach which allows different actors (human, software, …) of involved organizations to interact with each other by manipulating graphical objects. Often, these complex systems require sophisticated interfaces characterized by dynamic components which can provide information about system behavior. In this paper we present a research in progress, related to the integration of a paradigm borrowed by risk theory, within a tool for the evaluation of software systems through the analysis of visual components of its interface. The idea is to support organizations to define and properly allocate liability among system actors in order to identify recurring errors to possibly evaluate a potential reengineering of the system.

Keywords Systems analysis · Risk assessment · Failure mode effect analysis (FMEA) · Errors liability · Graphical user interface (GUI) evaluation

R. Cassino
Università degli Studi della Basilicata, Potenza, Italy
e-mail: rocassino@gmail.com

A. Vozella (✉) · G. Gigante · D. Pascarella
Italian Aerospace Research Centre, Capua, CE, Italy
e-mail: a.vozella@cira.it

G. Gigante
e-mail: g.gigante@cira.it

D. Pascarella
e-mail: d.pascarella@cira.it

T. Torre et al. (eds.), *Empowering Organizations*, Lecture Notes in Information Systems and Organisation 11, DOI 10.1007/978-3-319-23784-8_18

1 Introduction

Information and Communication Technology (ICT) widely spreads over productive, administrative and social organizations. They represent interconnected socio-technical systems where interactions are possible among different domains. Risks of such ICT pervasiveness lie in vulnerability to hazards for systems and citizens; in fact, malicious attacks and system failures may result is catastrophic effects. Most of the Critical Infrastructures (Internet, Banks, plants, airports…) are mainly software related and they are more and more interdependent so a disruption could lead to a domino effect. Let's think about Supervisory Control and Data Acquisition (SCADA) applications that get data about a system in order to control that system. They are used to automate complex industrial processes where basing only on human control is impractical. A wide variety of applications exist: Electric power generation, transmission and distribution is one example. Electric utilities use SCADA systems to detect current flow and line voltage, to monitor the operation of circuit breakers, and to take sections of the power grid online or offline. Another example is the water and sewage industry. State and municipal water utilities use SCADA to monitor and regulate water flow, reservoir levels, pipe pressure and other factors.

Air quality control in specific buildings and facilities is another application.

The manufacturing industry uses SCADA systems to manage parts inventories for just-in-time manufacturing, regulate industrial automation and robots, and monitor process and quality control.

Mass Transit authorities use SCADA to: provide and control electricity to subways, trams and buses; automate traffic signals for rail systems; track and locate transportation; control railroad crossing gates.

In industry, managers need to control multiple factors and the interactions among them (Management Information Systems—MIS). SCADA systems provide the sensing capabilities and the computational power to track everything that's relevant to operations. Most of these applications require sophisticated interfaces characterized by dynamic components by which the actors interact with the system.). In particular, suitable GUIs are implemented to let end-users of different systems interacting with each other.

In case of an accident, understanding the cause and the responsibility at the whole infrastructure level, requires huge efforts, sometimes thwarted by various attempts of manual search.

This situation has lead society to be cautious in the adoption of new technology for GUI, especially targeted to safety critical applications [1].

In this perspective, it would be desirable to have an automatic system to support organizations to define and properly allocate liability among actors, identifying recurring errors in order to evaluate a potential reengineering of the system and/or the use of different solutions.

Usability theory techniques are normally used to drive requirements of GUI for SCADA systems [2]. Safety assessment approaches, derived by risk theory, have rarely been applied to the design of Human Machine Interfaces (HMI) of Nuclear Power Plant [3].

On the other hand, several automatic approaches, based on the analysis of the system through its interface, exist in literature. Information on occurred errors on GUI usage can be found in log files which need to be post processed it in order to properly identify the wrong behaviour of the actors and to allocate its liability. An approach that combines different techniques—including formal analysis of models, simulation and analysis of log data—in a model-based environment is described in [4]. The log data at model level can be used not only to identify usability problems but also to identify where to operate changes to these models in order to fix usability problems [5]. On the other hand, a reverse engineering approach is used to automatically recover models of the GUI by dynamically "traversing" all its windows and extracting all the widgets, properties and values [6].

The proposed research integrates two approaches to identify liability in case of system failure:

1. the analysis of the software through the interactions of its graphical user interface;
2. a technique by risk theory based on the analysis of the failure modes (failure mode effect analysis, FMEA).

In particular, through automatic detection of the dynamic components of the interface the corresponding wrong tasks are identified and characterized in details, also in terms of liability allocation.

In this perspective, the innovation of the presented research is twofold:

1. To evaluate the correctness of the interaction mechanisms of a software system by the FMEA, as described in Sect. 2.
2. To characterize and record system malfunctions by the analysis of the GUI screen shots and the interaction with the system actors.

The paper is organized as follows. Section 2 presents an introduction to the risk assessment technique (FMEA); Sect. 3 describes the target application domain of the SCADA systems which represent safety critical applications for the GUI. Section 4 presents the integration of the FMEA module, within a tool for the evaluation of software through the analysis of visual components of the interface. Section 5 shows some conclusions and further research.

2 FMEA Technique by Risk Theory to Support Liability Allocation

The FMEA technique, is generally used to drive software requirements at design stage or to test the system [7]. In scenarios where several operators/controller and several automated support systems interact together for the fulfilment of a task, humans and technology represent a joint (cognitive) system. The term "joint cognitive system" means that control is accomplished by an ensemble of cognitive systems and (physical and social) artefacts which cooperate towards a common goal. Under this perspective, a still open question is that of how to deal with cases in which (as in some recent aviation accidents) conflicting information is provided to operators by humans (controllers) and automated systems, and more generally what kind of priorities should be given to different signals, and when humans may override automatic devices.

The question concerns how to properly analyse and manage the shift of liability due to automation, in order to achieve an optimal allocation of burdens. This will imply reconsidering the role of liability, not only as a tool to redistribute risks and allocate sanctions for errors and accidents, but above all as a means to prevent those accidents and to increase levels of safety and performance fostering the development of a safety culture within organizations. Thus, it will be essential:

1. to identify error tasks and roles of operators and automated tools by the interaction mechanisms of the system interface;
2. to identify violations of the expected level of performance for each task;
3. to consider different kinds of errors (unintentional rule violations, reckless behaviours, intentional violations);
4. to allow automatic recording with all the necessary details of different errors, risks and accidents.

Safety–critical industries and organizations have built paradigms and guidance on a so-called "just culture" to learn from failure with appropriate accountability.

The aim consists in no longer seeing such accidents as meaningless, uncontrollable events, but rather as failures of risk management, and behind these failures are many different actors.

A just culture, then, asks for the sustainability of learning from failure through the reporting of errors, adverse events, incidents [8].

Among the different analysis techniques available in the Risk assessment theory, the one which can support in characterizing the error is FMEA.

Considering that some forms of accountability, and accountability relationships between stakeholders, can be more constructive for safety than others we propose to integrate a module to trace failures in terms of: failure mode description, cause, actor, effects (local and global), detection mode (parameters, time constraints…), associated mitigation actions.

We propose to adopt FMEA techniques to implement a module which can support the identification of run time errors and their complete characterization improving a tool for software system evaluation through the analysis of its graphical user interface.

FMEA is mostly used to drive system component reliability requirement or to assess the system reliability of integrated products. It goes through the following steps:

- Identification of each component and related function/functions;
- Identification of the system boundary;

- Identification of the potential failures and the probability of that failure (occurrence) for each component;
- Definition of the system impact of each failure (severity).

We extend the previous scheme by adding the following features:

- Failure cause, actor, effects (local and global);
- Detection mode (parameters, time constraints…);
- Associated mitigation actions.

A GUI component of a software system allows operators to perform some tasks in a defined process.

A "dynamic" graphical widget is the element of the interface with which it is possible to interact triggering the associated function. Each interaction corresponds to a specific workflow aimed at implementing steps in a process. This workflow may have some constraints in terms of steps to be followed, time to complete the interaction, input to be fed to the system etc. Our approach consists in defining, for each workflow and for each interaction on the GUI, the following information:

- Component Id
- Component Name
- Function
- Potential Failure modes
- Effect of Failure
- Probability of Failure (Occurrence)
- Severity
- Failure cause
- Actor
- Detection mode (parameters, time constraints…)
- Associated mitigation actions

A function is a specific process, action or task that the system is able to perform.

Failure is the loss of an intended function of a device under stated conditions. Severity is the consequences of a failure mode. It considers the worst potential consequence of a failure determined by the degree of injury, property damage or system damage which could ultimately occur.

Occurrence is defined as how often or how likely a failure mode will occur.

Cause is a circumstance that may result in a failure think of the cause as why it would go wrong.

Failure effect is the immediate consequence of a failure on operation, function or functionality, including impact on a customer, both internal and external, if a failure mode is not prevented or corrected.

Then, it is necessary to define the impact that each failure would cause (severity) to the system. In this sense, the next step is to determine the severity of the failures identified in the previous step. Severity is determined in the same manner as occurrence:

- No effect
- Very minor (only noticed by discriminating customers)
- Minor (very little impact on the system, but noticed by average customer)
- Moderate (most customers are annoyed)
- High (causes a loss of primary function; customers are dissatisfied)
- Very high and hazardous (the failure may result in unsafe operation and possible injury/death).

An example will be provided in next section.

3 SCADA and GUI

This paragraph introduces the main parts of a SCADA system and its functionalities, to characterize it as a complex system in terms of heterogeneity of its components and interactions of different actors. A SCADA system performs four main functions: data acquisition, networked data communication, data presentation and control. These functions are performed by four kinds of system components to operate in a cohesive fashion.

The first are sensors (either digital or analog) and control relays which directly interface with the managed system. These report to Remote Telemetry Units (RTUs). These are small computerized units deployed in the field at specific sites and locations. RTUs serve as local collection points for gathering reports from sensors and delivering commands to control relays. These RTU's report to SCADA master units. These are larger computer consoles that serve as the central processor for the SCADA system. Master units provide a human interface to the system and automatically regulate the managed system in response to sensor inputs. These master units use the communications network to control the RTUs in the field.

A real-life SCADA system needs to monitor hundreds or thousands of sensors. Some sensors measure inputs into the system (for example, water flowing into a reservoir), and some sensors measure outputs (like valve pressure as water is released from the reservoir). A real SCADA system reports to human operators over a specialized computer that is variously called a master station, an HMI (Human-Machine Interface) or an HCI (Human-Computer Interface).

The SCADA master station has several different functions. The master continuously monitors all sensors and alerts the operator when there is an "alarm"—that is, when a control factor is operating outside what is defined as its normal operation. The master presents a comprehensive view of the entire managed system, and presents more detail in response to user requests. The master also performs data processing on information gathered from sensors—it maintains report logs and summarizes historical trends. In advanced SCADA systems master can add a great deal of intelligence and automation to system management, making operator job much easier.

The HMI Toolkit is the collection of all of the design elements for the displays (all of the static and dynamic objects) and the related operating console. The toolkit is a separate element, since one set will exists for each control or SCADA system in use.

All aspects of the HMI is intended for a specific set of purposes (primary and secondary) and a set of users (again primary and secondary). The actual tasks to be performed by the users are identified and put on a sequence diagram.

The HMI is conceptually designed to contain the following concepts.

1. Equipment Model -> Models site/equipment and linkage of SCADA values
2. Events -> Conditional expressions or time based events that trigger workflows
3. Workflow Templates -> Configured Templates that contain automated process and manual steps
4. Schedules -> Combines events with workflow templates to initiate workflows

The model of site and equipment stores real time SCADA values for evaluation.

The Conditional Expressions use values stored in Equipment to evaluate expressions.

Events can be triggered based on multiple criteria e.g. Time Based Expression when it uses date/time expression to determine when event should be triggered. Events can be set so one or all criteria must be met. Expressions can also evaluate criteria stored in the workflow process such as a process already running.

The workflow templates contain configured steps and processes which are executed automatically and/or with user interaction. Those steps can be modified by workflow authors and services can be added by administrator. The Workflow Schedule combines workflow template with events, either conditional or time based to determine if a workflow should be generated. When a workflow is triggered, either manually or automatically as part of a conditional or time based event, a series of processes occur an email is triggered, notifying appropriate personnel of the workflow and supplying a work order number if generated Information from SCADA is displayed in the workflow, along with specific information for troubleshooting. Steps guide users through resolution of the issue. They have expiration timers, if steps are not processed in time escalation processes occur (notifications). Besides the logged and recorded information we propose to extend the recorded information by the interactions and the changes on the system interface in terms of screen images for identified failure modes. Table 1 shows an example of FMEA performed for a component on a GUI of a SCADA system.

4 The Proposed Research

In this paper we propose a methodology for assessing the responsibilities of system errors by analyzing the software through the mechanisms of interaction of its graphical user interface.

Cassino and Tucci [9] presents a tool that implements an automatic evaluation system to measure some of the canonical usability parameters of the graphical interface of an application. As described in the cited paper, after the components of

Table 1 Example of FMEA workflow

Component name	Push button
Function	To activate Display RTU# numeric data with facility internal temperature
Failure mode	No activating Display RTU# numeric data with facility internal temperature
	Wrong Activating Display RTU# numeric data with wrong facility internal temperature
Effect of failure (local and global)	Wrong understanding of the state of some temperature sensor
	Worst case effect: wrong decision to go on powering the system with fire which breaks out
Probability of failure	Depending on the reliability of the system
Severity	Can be catastrophic
Failure cause	Error due to: GUI; Communication link; RTU; Sensor
Actor	Humans, automatic system…
Detection mode (parameters, time constraints…)	Inconsistent parameters in the shown string (no values, out of range values, less values then expected…) Visual feedback not matching the expectations…
Associated mitigation actions	Upload pictures for clarification and fast recognition
	To activate deeper control on the specific sensor, RTU, communication link

the GUI are identified, the system determines the type of each element (label, button, link, text-area, menus, etc.) by means of the changes to the interface itself which are activated by interacting with it. An automatic interaction process is implemented. As a matter of fact, the task is carried out by two modules: an automatic interaction module that simulates the input events generation, and a decision algorithm that chooses the interface element to interact with (and what type of input is to generate) and deduces, according to the observed changes in the graphics output, the class of object. Depending on the type of input and on the detected visual feedback, the following typologies of elements are determined:

- Input/output: dynamic element (it is possible interact with it);
- Pause: the system provides a description of the item;
- Press and release of left button of the mouse: the element is a button or a menu item;
- Double click: if the visual feedback is isolated to a neighbourhood of the point with which you interacted, the element is a character and all pixels affected by the change form a group (string);
- Click and double click: the item is a link;
- Insertion of a character: the element is a text field or a text-area.

Now, we extend the interaction analysis performed on the dynamic components of the GUI by the FMEA method to identify and allocate malfunction liability among system actors.

Fig. 1 System model

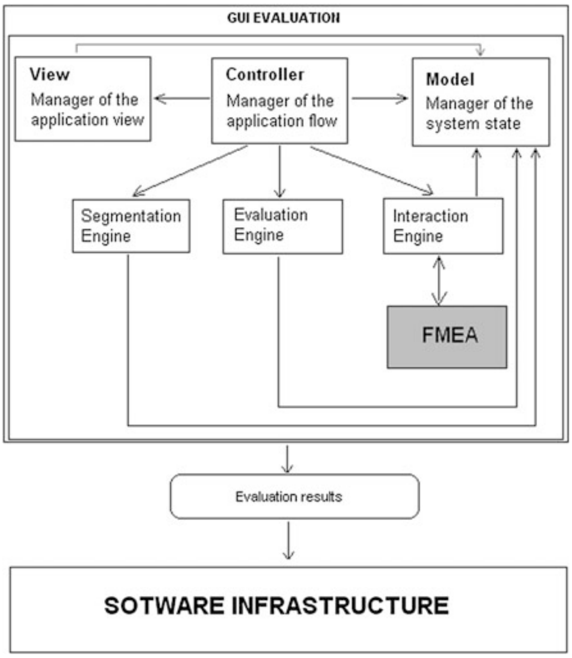

Figure 1 shows the system model designed. The view module includes the functionalities related to the rendering of the system. It sends the controller each user requests and allows the controller to select a particular view.

The controller module defines the flow of execution. Its classes map user requests into actions performed on the model and select a view related to user requests. For each input event, a specific class describes the action to be performed on the model and selects the next view.

The model module encapsulates the application state.

Below a formal description of the FMEA module is provided.

Given a set of workflows on the HMI, we build a set of corresponding nominal (due to correct interactions) visual feedbacks by running the system for GUI evaluation.

For each WFi let G(WFi)j be the corresponding visual feedback j.

Then we generate the corresponding off nominal visual feedbacks by associating wrong visual feedbacks to the specific workflow.

Specifically, for each defined workflow, a list of associated failure modes are identified with all the necessary attributes in terms of actors, causes, effects, parameters, associated visual feedbacks- and recorded in a database. Then:

$$(\mathrm{WFi}) \rightarrow [\mathrm{Fk}(\mathrm{G}(\mathrm{WFi})\mathrm{j}), \ldots \mathrm{Fn}(\mathrm{G}(\mathrm{WFi})\mathrm{j})]$$

With Fk(G(WFi)j) we mean a string made by the description of the Failure Mode k for Visual feedback j of Workflow WFi, the cause of Failure Mode k for Visual feedback j of Workflow WFi, the actor of the Failure Mode k for Visual feedback j of Workflow WFi, the effect of the Failure Mode k for Visual feedback j of Workflow WFi.

We use the tool with the FMEA module to generate for a specific workflow all the set of possible nominal and off nominal visual feedbacks for it, completed with the corresponding information. Each workflow with its related information will be stored in a database.

When an interaction is performed within a specific workflow, the decision algorithm will identify and deduce, according to the observed changes in the graphics output, the class of object according to the observed changes in the graphics output. If this object corresponds to an off nominal visual feedback for that workflow, within the database all the related information about liability and other attributes is already available for further use.

5 An Example

Let us start the example by considering the first following case: a complex interface adopted in a mission control centre to command and control the on board processing of an unmanned vehicle. The interface allows the operator to implement different mission procedures (Fig. 2).

During the execution of a mission a failure occurs. Only the GUI execution trace and the GUI FMEA are available to the operator for processing. The generation of FMEA is completely automatic and independent on the application usage of the GUI.

In a preliminary off-line phase the software tool is executed by giving the GUI under study as input to the View module. The GUI is processed by the segmentation engine and the evaluation engine [9] and the relative hierarchical tree is built (Fig. 3).

The interaction module feeds the tree to the FMEA module. The module identifies:

- For each node the direct child that is the expected next widget to be stimulated in the nominal sequence;
- Any off nominal workflow by considering any combination of widgets different from those represented by the tree.

The identified pair (*parent, child*) represents a record in the FMEA table for which the failure modes are all the combination of *parent* with all other widgets different from *child*.

Let:

- $W_i = \{w_{i1}, .., w_{ik}\}$ a nominal sequence of widgets on the GUI.
- (w_{im}, w_{in}) the current pair parent–child
- $\Delta = \{(w_{im}, w_{ij}) | j \neq n\}$ the set of off-nominal pairs

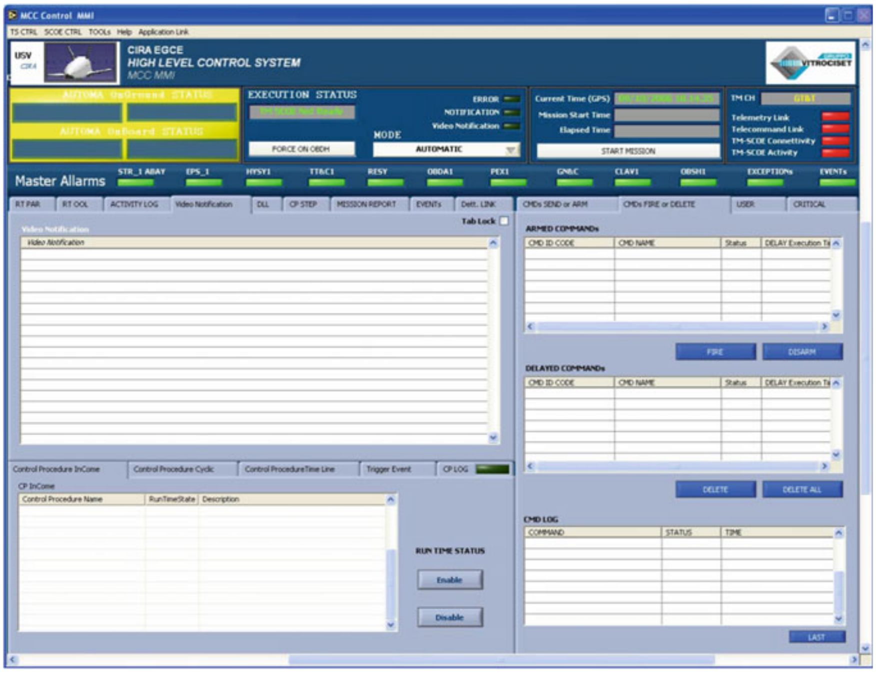

Fig. 2 Mission control center main GUI

The module automatically generates the FMEA table storing it into the database.

In post-processing phase a software program can process the trace log and checks for each pair of stimulated widgets if they are in the failure mode column. In the positive case a possible deviation point from the nominal workflow has been detected identifying also the correct sequence that should have been executed.

Let us follow on the example by considering a second case: The generation of FMEA is partially automatic and dependent on the application usage of the GUI. This means that the FMEA module allows an operator to associate each path in the tree with a specific procedure to be executed by the operators during mission.

Let $\{P_1, \ldots, P_n\}$ the set of procedures to be executed. Each procedure can be described in terms of a sequence of widgets to be stimulated addressing its nominal execution. For each procedure information as actors and the consequence on the system of the failure of each step can be described.

In this case in post processing the software analyser could intercept the deviation point and also other important information as the mission procedure failed, the relative step and the actors involved.

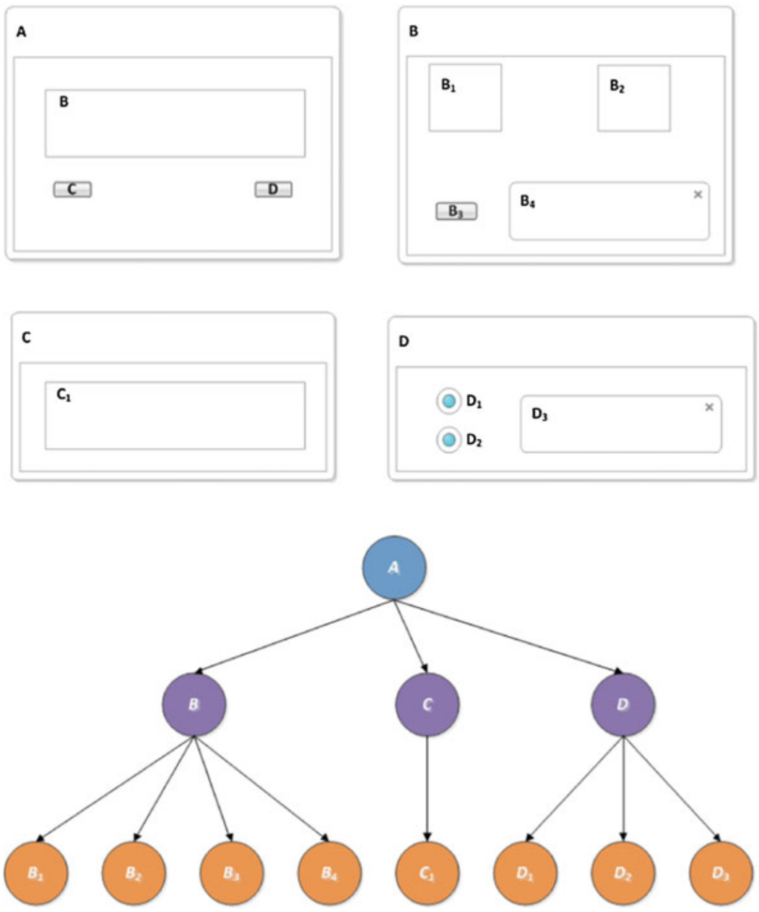

Fig. 3 Hierarchical tree built up from the abstract view of the GUI

6 Conclusions

In the domain of productive, administrative and social organizations, the interconnection and the interactions among different actors can expose the system to serious risks with critical consequences. Supervisory Control and Data Acquisition (SCADA) applications get data about a system in order to control those critical system allowing different actors to operate on its Human Machine Interface (HMI). In case of accident it is not so easy to properly identify liability and failure causes.

In this paper we propose to integrate a technique by risk theory, FMEA (Failure Mode Effect Analysis), to support the identification and allocation of liability in case of critical system accidents within an existing automatic tool for software system evaluation through the analysis of interaction performed by its graphical user interface. The visual feedback of the tasks allows identification and characterization of possible failure modes of the system through “this visual approach”, so to drive system requirement definition.

Medium term research work will involve the deepening of the integration features between the two control systems, the implementation of the related environment and the following use of this mechanism to analyse errors in system interactions with the GUI at run time, to build statistical data and derive whole system improvement issues.

References

1. Vozella, A.: Promoting trust protecting citizens’ rights in digital citizenship. In: AICA 2013 proceedings
2. Vozella, A. et al.: Usability issues for an aerospace digital library. In: AVI 2012 proceedings
3. Orekhova, V.K.: Safety case-oriented assessment of human-machine interface for NPP I&C systems. Integrated HMI safety assessment methodology for safety-critical I&C sytems, RT&A # 03 (26) (vol. 7) (2012)
4. Palanque, P., Barboni, E., Martinie, C., Navarre, D., Winckler, M.: A model-based approach for supporting engineering usability evaluation of interaction techniques. In: Proceedings of the 3rd ACM SIGCHI symposium on engineering interactive computing systems, pp. 21–30. ACM (2011)
5. Melody, Y., Ivory, Marti, A.H.: The state of the art in automating usability evaluation of user interfaces. J. ACM Comput. Surv (CSUR) Surveys Homepage Arch. **33**(4) (2001)
6. Atif M.M.: Using reverse engineering for automated usability evaluation of GUI-based applications. Human-Centered Software Engineering: Human-Computer Interaction Series, pp 335–355 (2009)
7. Kraig, S.: Using FMEA to improve software reliability. In: PNSQC 2013 proceedings
8. Sidney, W., Dekker, A.: Just culture: who gets to draw the line? Springer, London (2008)
9. Cassino, R., Tucci, M.: Automatic usability evaluation of GUI: a front-side approach using no source code information. Lecture Notes in Information Systems and Organization, vol. 2, pp. 439–447A (2013)