Safety in Interactive Hybrid Online Labs

http://dx.doi.org/10.3991/ijoe.v11i3.4557

Karsten Henke, Tobias Vietzke, Heinz-Dietrich Wuttke, Steffen Ostendorff
Technische Universität Ilmenau, Ilmenau, Germany

Abstract— **Based on the complex grid infrastructure of the Ilmenau Interactive Hybrid Online Lab various possibilities of occurrence of malfunctions as well as mechanisms to avoid them will be discussed. This will be demonstrated by different examples. One implementation challenge is to protect the physical systems in the lab against wrong control algorithms or malicious trying to sabotage or to destroy the system - without defining too many design constraints. Therefore, a high level of safety must be guaranteed for such an autonomous working system. Therefore the first step is the verification of sensor and actuator constellations to guarantee safety in online labs. Based upon this an efficient possibility for a validation of the student's design will be shown as subsequent step.**

Keywords*— *control engineering education; laboratories; Web-based education; virtual and remote labs; Web-based design tools; distance learning.

I. INTRODUCTION

The Integrated Communication Systems Group at the Ilmenau University of Technology is an expert in the field of Internet-supported teaching of digital system design and is well experienced in the area of integrated hard- and software systems for over 10 years. The students have to pass hands-on examinations in a lab to complete the learning outcomes by own experiences. For all students, hands-on experiences are important to deepen their knowledge about topics they learned during lectures. With our interactive hybrid online lab, called *GOLDi* (Grid of Online Lab Devices Ilmenau), we want to offer the students a working environment that is as close as possible to a real world laboratory. Under real laboratory conditions disturbances can appear and lead to failures of the control algorithm that cannot be detected under virtual lab conditions. Facilities of hybrid online labs provide permanent online access for students and supervisors, and give possibilities to check different parts of the designs most easily. Besides the advantages for students, this also reduces the costs for academic teaching and improves the overall quality by offering more practical training options.

This gives students the chance to realize correct designs, to organize self-study process of the student more efficiently, to control student's work and to broaden the ways of communication in research work with companies as well. This solution is intended

- for the use in teaching materials dealing with the design of digital control systems and embedded systems – from the basics up to complex design tasks as well as

within the newly established Tempus projects "ICo-op – Industrial Cooperation and Creative Engineering Education based on Remote Engineering and Virtual Instrumentation" [1] and "DesIRE – Development of Embedded System courses with implementation of Innovative virtual approaches for integration of Education, Research and Production in UA, GE, AM" [2].

The detailed concept, including the possibilities and limitations of such an approach, as well as the infrastructure and many different fields of applications of the Ilmenau Interactive Hybrid Online Lab *GOLDi* (see Fig. 1) were presented in previous papers [3, 4, 5, 6].

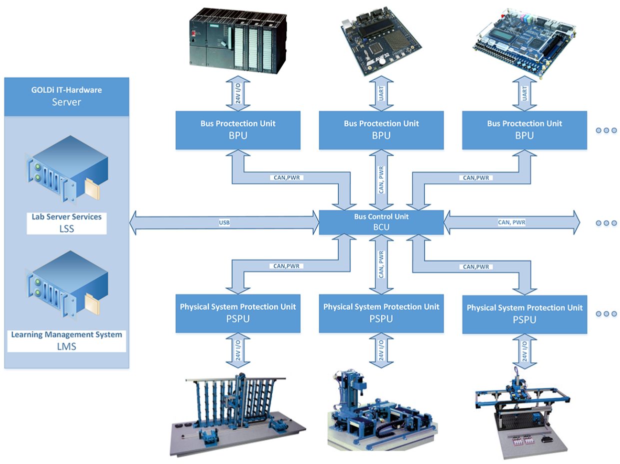

Figure 1. Overview of the Ilmenau Interactive Hybrid Online Lab *GOLDi*

II. SECURITY AND SAFETY IN ONLINE LABS

One implementation challenge is to protect the physical systems (the electro-mechanical hardware model, e.g. elevator, water-level control, high-storage warehouse, 3-axis portal) in the lab against

- wrong control algorithms of unskilled students or
- malicious trying to sabotage or to destroy the system

without defining too many design constraints. Therefore, a high level of safety must be guaranteed for such an autonomous working system.

Generally two types of "security/safety" must be considered for systems with access to the Internet – like our online labs:

(1) Information Security (security)

defines procedures and requirements concerning the protection of information processing as well as some policies to avoid unauthorized data manipulations [7] – see Fig. 2.

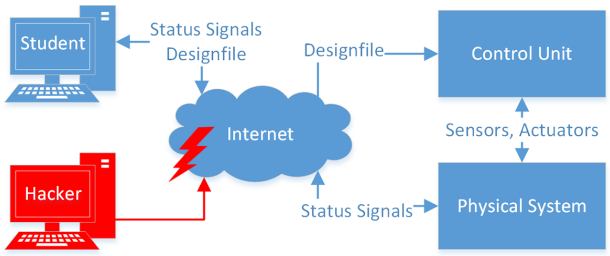

Figure 2. Information Security in Online Labs

This topic concerns first and foremost the remote lab server as well as the experiment booking system but should not be the focus of this article.

(2) Operational safety (safety)

defines functional safety of electrical, electronic, programmable electronic safety-related systems. It defines several safety integrity level (SIL) requirements for any

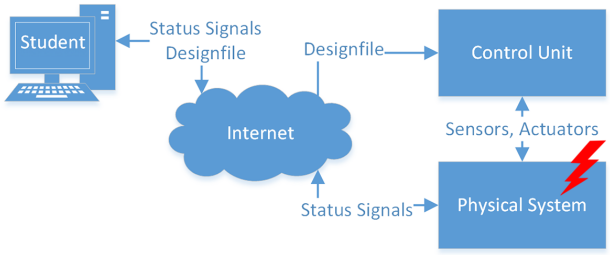

Figure 3. Operational Safety in Online Labs

safety function and provides a risk-based conceptual framework [8] – see Fig. 3.

There are different protection strategies already existing in the industry; however, they are designed for a specific application and require special software or they impose design constraints. More flexibility is required for online labs with constantly changing tasks, especially when students are free in their design decisions and should be encouraged to develop their own creative solutions for a given task / problem.

To guarantee the safety within the *GOLDi* infrastructure two main components are necessary (see Fig. 1)

the *bus protection unit* (BPU) to interface various control units where the students design task is running on (e.g. FPGA, Microcontroller, PLC, …) to the internal remote lab bus and to protect it from misuse and damage as well as

the *physical system protection unit* (PSPU), which protects the physical systems against deliberate damage or accidentally wrong control commands and which offers different access and control mechanisms.

The *bus protection unit* receives commands from a control unit and simply checks them for bus validity. This is done by using the specific *GOLDi* transmission protocol [9]. The content of the transmitted data and addresses are not checked, because this depends on the selected physical system and is therefore done by the specific protection units of each physical system. The function of the bus protection unit is to prevent a control unit from blocking the bus and causing others to be affected. The bus protection unit is based on the same hardware as the protection units for the physical hardware models in the remote lab but uses different add-on boards and a different firmware. This simplifies production and maintenance.

The *physical system protection unit* is necessary when students execute their algorithms on the selected control unit and want to be completely free in their choice of design tools. All protection mechanisms discussed within this paper are executed inside an FPGA on this PSPU.

Compared to the approach used so far, there was no possibility to check, if the executed commands are safe for the physical system. This means, damage could be caused by invalid commands. The first task of the protection unit (the *verification* process) is to check for command safety by filtering all commands. Only commands that will not cause any malfunction will be transferred to the physical system. All others are discarded. A *validation* as second step can be used as indicator for the quality of the students design. The feedbacks from the verification and validation module will be reported to a learning management system (LMS) to give the student immediately information about the occurrence of a fault and inform him about the quality of the realized design task – see Fig. 4.

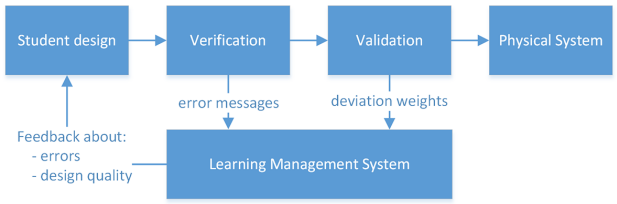

Figure 4. Observation of the student's design – optionally under LMS control

Using such universal protection units gives the students the largest degree of freedom for their design, because no precautions have to be taken into account. Therefore, no additional security framework (workbench) within the software and hardware control design is required to prevent malfunctions of the physical system. The complete design flow is carried out at the students' side, giving them a more authentic look at a real world project design flow.

A normal faultless communication during a running experiment is shown in Figure 5. . The physical system (e.g. an elevator) generates different sensor signals (x_{sensor}). Based on these sensor input signals the control unit generates the corresponding actuator output signals ($y_{actuator}$) according the implemented student's control algorithm [10].

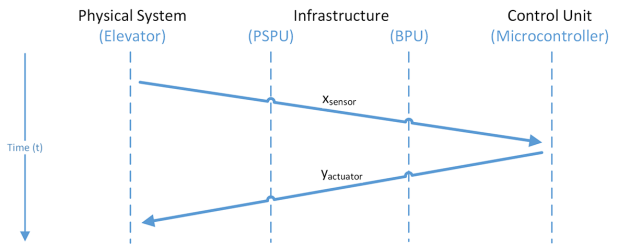

Figure 5. Faultless communication during a running experiment

Figure 6. shows an example for a sequence diagram for a 3-floor elevator which drives upwards from the 2nd floor with destination floor no. 3. The sensor signal x_1 *[cabin position 2nd floor]* signalizes that the cabin is still in floor no. 2. This signal will be transmitted via the PSPU and the

BPU to the connected control unit. Thereupon the control unit will activate the actuator signal y_0 *[drive upwards]* which will be transmitted via the BPU and the PSPU back to the physical system to finally move the cabin upwards.

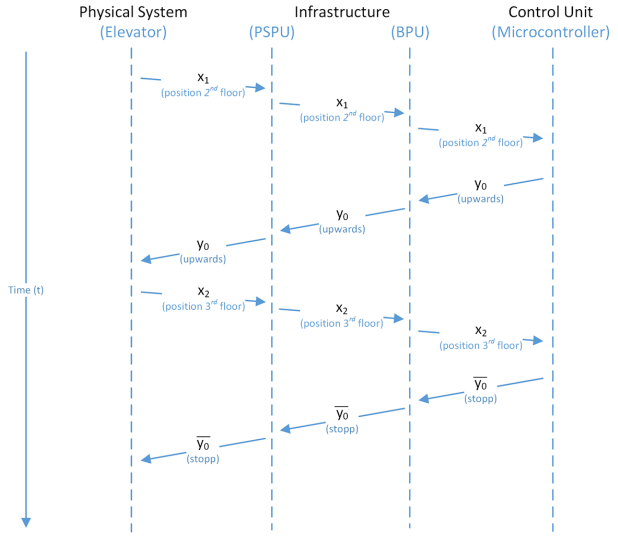

Figure 6. Sequence Diagram example: *"move upwards and stop at 3rd floor"*

If the cabin reaches the 3rd floor sensor signal x_2 *[cabin position 3th floor]* will be activated and the whole communication cycle starts again – the control unit will now deactivate the actuator signal y_0 to stop the cabin motion.

In both directions the PSPU checks the signals on validity – only correct actuator signals will be transferred to the physical system. The detailed verification and validation concept will be described in the next sections.

III. OPERATIONAL SAFETY THROUGH VERIFICATION

A verification of the control signals is used to guarantee the operational safety of the remote lab infrastructure. It means the electromechanical hardware models (physical system) must be protected against destruction. Malfunctions in the control can have multiple causes. They are defined as control signals (actuator signals), which cause a damaging or destruction of the physical system according the actual status of the input signals (sensor signals).

For online labs we have to distinguish between two causes of faults: on the one hand faults caused by the user *("user-based faults"*) as well as faults due to the remote lab infrastructure and the communication within this infrastructure on the other hand (*"infrastructure-based faults"*). The whole fault diagnosis and handling will be managed through the PSPU by an FPGA. In any case the user must receive information on the occurrence of the fault. Additionally these faults will be logged to give feedback to a connected LMS or to the responsible tutor.

These two types of faults as well as the error detection mechanisms will be described in the following.

A. User-based Faults

These faults are caused by the users e.g. by wrong control algorithms or malicious trying to sabotage or to destroy the system. Figure 7. shows an example for a faulty elevator control algorithm. The cabin is already located on the top (3rd) floor. After receiving the corresponding sensor signal (x_2 *[cabin position 3rd floor]*) the student's control algorithm still generates the wrong actuator signal (y_0 *[move cabin upwards]*) to drive upwards. In this case the PSPU must

reject this actuator signal,

stop the cabin,

give the student a feedback about his mistake and

optionally inform the connected LMS about the situation.

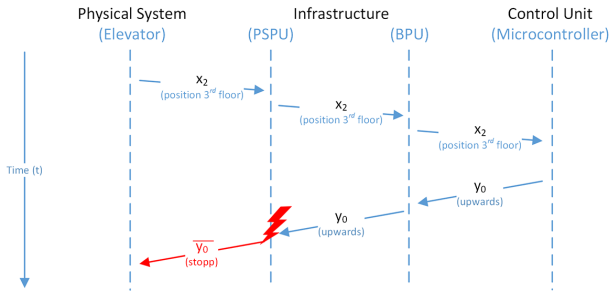

Figure 7. Example for an user-related fault

B. Infrastructure-based Faults

The *GOLDi* infrastructure itself and the communication between all remote lab components can also cause fault situations.

1) Invalid sensor value constellation

The simplest case is a faulty electrical connection between a sensor and the PSPU or a defective sensor. This leads to invalid sensor values. Figure 8. gives an example for two simultaneously active sensor signals in the 2nd floor (x_1 *[cabin position 2nd floor]*) as well as in the 3rd floor (x_{02} *[cabin position 3rd floor]*). The PSPU detects this situation and stops all actuator signals which will bring the physical system in a fail-save state. The user will be informed about the situation by an error message and a continuation of the experiment is only possible after solving the problem.

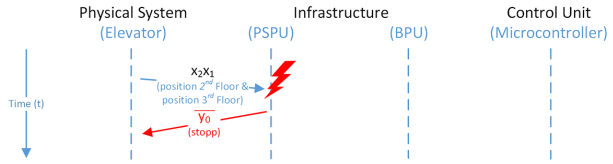

Figure 8. Example for invalide sensor constellation detection

Furthermore it is not so easy to detect a defective sensor which generates no sensor signal. Only such floor sensors can be detected if there are additional sensors above and below. Defective floor sensors in the lowest and highest floor cannot be detected.

2) Infrastructure-based delay (without timeout)

The quality of the Internet connection speed leads to a more or less large time delay between the activated sensor signal (e.g. x_2 in Figure 9. Figure 4. Figure 3.) and the calculated corresponding actuator signal (e.g. y_0 in Figure 9. Figure 3.). Especially if the selected control unit for this experiment is not located in the remote lab, but outside on the student's client PC or a tablet.

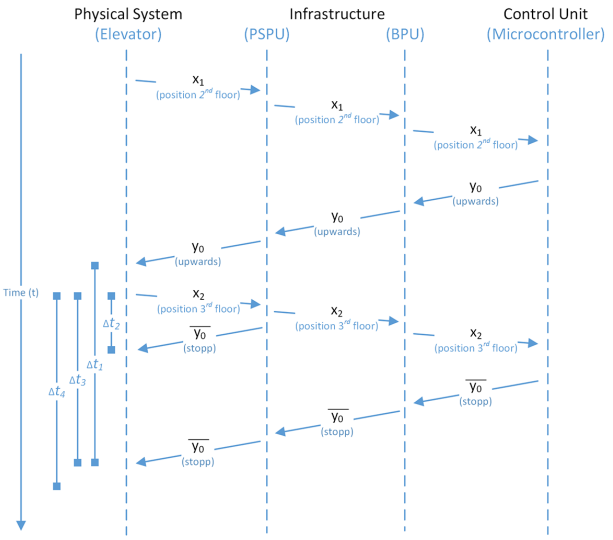

Figure 9. Infrastructure-based delay (without timeout)

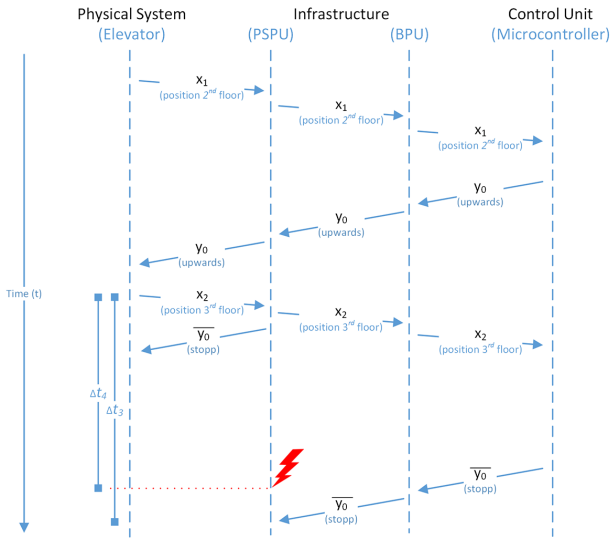

Figure 10. Infrastructure-based error (timeout)

This means that the cabin moves continuously upward for a certain time Δt_1(see Figure 9.) and could damage the physical system. That's why the PSPU must stop the cabin motion immediately after Δt_2. According to the internal PSPU architecture this takes only a few nanoseconds. This means the physical system is in a fail-safe state temporarily. This is the same effect as the above described *invalid sensor value constellation* detection. But in this case the PSPU will wait for a response of the control unit for a certain time Δt_4. Normally the control unit detects the situation (activated sensor x_{02}) and reacts within Δt_3 with the corresponding actuator signal (deactivated y_0) to stop the motion. If this response time Δt_3 is within the range of the given time slot Δt_4, the PSPU will not generate any error messages. Figure 9. demonstrates this effect. This time slot Δt_4 is adjustable – for the *GOLDi* infrastructure it is defined with 2 seconds.

3) *Infrastructure-based error (timeout)*

The just described situation assumes that the control unit will generate an actuator signal (as the result of any sensor changing) within a defined time slot Δt_4.

If the reaction time Δt_3 is greater than the specified time slot Δt_4; the PSPU must generate a timeout and cancel the experiment to bring the physical system in a stable state (see Figure 10.). The user will be informed about this. In general it makes no sense to continue the experiment if the Internet speed is too slow.

A timeout will also be generated if something goes wrong within the control unit e.g. the control unit doesn't send actuator signals anymore.

C. *Error Detection Mechanisms*

After the description of possible faulty situations during a running experiment (user-based, infrastructure-based) the mechanism for the detection of these errors will be described in the following.

As mentioned before the whole fault diagnosis and handling will be managed through the PSPU by an FPGA. Thereby three modules within the PSPU have to analyze

(1) sensor input signal (x_{sensor}) constellations

This module monitors the sensor signals of the physical system and detects forbidden constellations; for instance simultaneous activation of sensor x_1 *[cabin position 2nd floor]* and sensor x_2 *[cabin position 3rd floor]*.

(2) actuator output signals ($y_{actuator}$) constellations

This module monitors the actuator signals of the physical system and detects forbidden constellations; for instance it is not allowed to open the cabin doors (activated y_7 *[open cabin door at 3rd floor]* while driving upwards (activated y_0 *[move upwards]*).

(3) constellations of sensor and actuator signals

This module monitors the sensor and actuator signals of the physical system and detects forbidden constellations; for instance it is not allowed to move the cabin upwards (activated y_0 *[move upwards]*) as long as the door in the 2nd floor is open (activated x_{11} *[door 2nd floor is open]*).

Because the error handling will be done by an FPGA, all three modules can be executed in parallel. If one of the modules detects an error, a corresponding error code will be generated and transferred to the main *experiment flow control* module (see Figure 11.). This flow control module finally decides whether an intervention in the ongoing experiment is necessary.

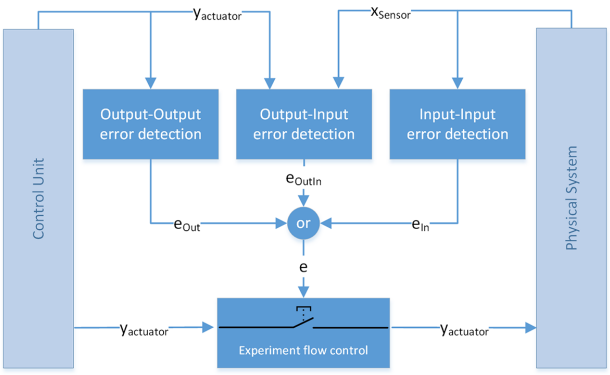

Figure 11. Error Detection Mechanism

Heart of each of these three diagnosis modules is a simple matrix structure to register all possible forbidden constellations. Figure 12. gives an impression for such a matrix for the third module (sensor/actuator constellations).

x_{25}	x_{24}	x_{12}	x_{11}	x_{10}	x_9	x_8	x_7	x_5	x_1	x_0	y_{23}	y_{15}	y_{14}	y_{13}	y_8	y_7	y_6	y_5	y_4	y_3	y_1	y_0
0	0	1	-	1	-	1	-	0	-	-	-	-	-	-	-	-	-	-	-	-	-	1
0	0	1	-	1	-	1	-	-	-	0	-	-	-	-	-	-	-	-	-	-	1	-
-	-	-	-	-	-	-	0	-	-	1	-	-	-	-	-	-	-	-	-	1	-	-
-	-	-	-	-	-	0	-	-	-	1	-	-	-	-	-	-	-	-	1	-	-	-
-	-	-	-	-	0	-	-	-	1	-	-	-	-	-	-	-	-	1	-	-	-	-
-	-	-	-	0	-	-	-	-	1	-	-	-	-	-	-	-	1	-	-	-	-	-
-	-	-	0	-	-	-	-	1	-	-	-	-	-	-	-	1	-	-	-	-	-	-
-	-	0	-	-	-	-	-	1	-	-	-	-	-	-	1	-	-	-	-	-	-	-
-	-	-	-	-	-	-	-	-	-	1	-	-	-	-	-	-	-	-	-	-	-	-
-	-	-	-	-	-	-	-	-	1	-	-	-	1	-	-	-	-	-	-	-	-	-
-	-	-	-	-	-	-	-	1	-	-	-	1	-	-	-	-	-	-	-	-	-	-
1	-	-	-	-	-	-	-	-	-	-	1	-	-	-	-	-	-	-	-	-	-	-

Figure 12. Matrix for the Sensor/Actuator Diagnosis module

Let's consider the following constellation exemplarily. The actuator signal $y_0 = 1$ (highlighted line in Figure 12.) of the physical system could be activated by the main *experiment flow control* module **only if**:

the actuator signal y_0 generated by the control unit is activated and

the cabin is not already in the 3rd floor ($x_2 = 0$) and

all doors in the three floors must be closed (x_8=1, x_{10}=1, x_{12}=1) and

the overload and the emergency stop indicators must be deactivated (x_{24}=0, x_{25}=0).

This means that all functionality of the error detection can be reduced to combinational equations implemented in VHDL on the FPGA of the PSPU, e.g. for the described situation:

$$y_0 = y_0(control\ unit) \wedge \overline{x_2} \wedge x_8 \wedge x_{10} \wedge x_{12} \wedge \overline{x_{24}} \wedge \overline{x_{25}}$$

IV. EVALUATION THROUGH VALIDATION

In addition to the *verification* of the students design necessary to guarantee the safety inside the online lab (described in the previous section), users want to have a fast feedback to their realized control task. Therefore, possibilities for an automatic *validation* of the user's solutions within the *GOLDi* infrastructure were implemented as well (see Figure 4.). Thus, an assessment of the quality of the developed solution can be given immediately. Therefore a reference design and a method to check the students' design against this reference are needed. The reference design should be independent of the used control unit and the development tools and only specifies certain objectives of the given task.

It is the main advantage of this verification that only valid sensor/actuator constellations (passed through the verification module) will serve as inputs for the reference automaton. Eventually these are not efficient according the given task; but they will not cause any faulty problems on the physical system. Both changes in sensor signals as well as changes in actuator signals will act as "events" to trigger the reference automaton (one event = one automaton execution step).

To evaluate the students design, the given task will be divided into several *subtasks*. The most efficient path through the automaton graph (best quality of the realized design) is a vertical processing from state to state. If there are any deviations from the ideal design, each subtask contains several "co-states" where the tutor can define a "weight" for this deviation. Strong deviations from the ideal design mean large weights; little deviations mean small weights. For the overall quality of the students design it finally means: the lower the earned deviation weight points the better the design! The design of the reference automaton with the possibility to allow several deviations of the optimal design in each subtask and the specification of the weights for a certain deviation is a very ambitious task for the tutor and requires a lot of experience. That's why only an extract of a very simple design task will be used to illustrate the validation mechanism.

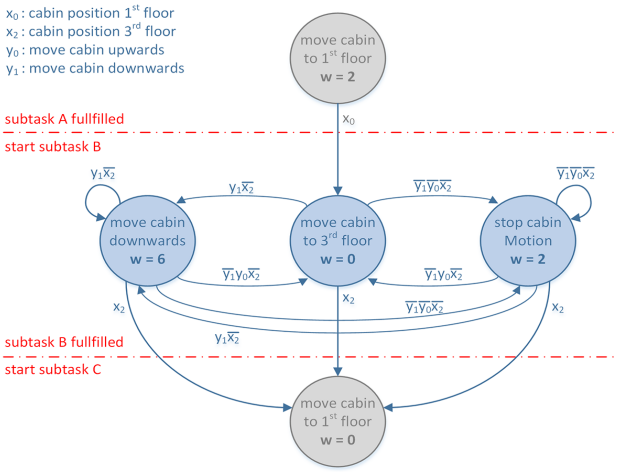

Figure 13. Example for a reference design automaton graph (excerpt)

Figure 13. shows an extract of an automaton graph for a reference design of the following task: *"Move the elevator as quick as possible from the 1st floor to the 3rd floor – without any stop or opening of the doors. Stop at the 3rd floor and then open the doors"*.

The reference design for this simple task can be divided into several subtasks. The first subtask *"move cabin to 1st floor"* is fulfilled, when sensor x_0 … *cabin position 1st floor* is activated. For the next subtask (main state) *"move cabin to the 3rd floor"* the following two deviations (co-states) are allowable: "stop cabin motion" (with a weighting of 2) and *"move cabin downward"* (with a weighting of 6 – because this means a stronger deviation from the given task). This subtask can be fulfilled from any of these three states (main state or the two co-states) by activating the sensor signal x_2 … *cabin position 3rd floor*.

	Actual state	Event	Next state	weight
(1)	Move to 3rd floor	$\overline{y_1}\ \overline{y_0}\ \overline{x_2}$	Stop motion	2
	Stop motion	$\overline{y_1}\ y_0\overline{x_2}$	Move to 3rd floor	0
			Earned weights:	***2***
(2)	Move to 3rd floor	$y_1\overline{x_2}$	Move downward	6
	Move downward	$\overline{y_1}\ y_0\overline{x_2}$	Move to 3rd floor	0
			Earned weights:	***6***
(3)	Move to 3rd floor	$\overline{y_1}\ \overline{y_0}\ \overline{x_2}$	Stop motion	2
	Stop motion	$y_1\overline{x_2}$	Move downward	6
	Move downward	$\overline{y_1}\ \overline{y_0}\ \overline{x_2}$	Stop motion	2
	Stop motion	$\overline{y_1}\ y_0\overline{x_2}$	Move to 3rd floor	0
			Earned weights:	***10***

Figure 14. Matrix for the Sensor/Actuator Diagnosis module

Starting from the main state (see Figure 13.), the following three sequences will give an impression how to calculate the overall weighting for the actually performed design task – with derivations from the ideal design:

1. The motion starts upwards according the given task. But now the motion will be stopped. After continuing motion upwards the task will be fulfilled by reaching the 3rd floor. The student will earn in sum 2 derivation points.
2. The motion starts upwards according the given task. But now the motion will change to downward. A strong derivation! After continuing motion upwards the task will be fulfilled by reaching the 3rd floor. The student will earn in sum 6 derivation points.
3. The motion starts upwards according the given task. Now the motion will be stopped. After that the motion will change to downward and stops again. Finally after continuing motion upwards the task will be fulfilled by reaching the 3rd floor. The student will earn in sum 10 derivation points.

V. CONCLUSION

Main focus of this paper is a complex *verification system* to protect the electromechanical hardware models (physical systems) in a remote lab against wrong control algorithms of (unskilled) students or malicious trying to sabotage or to destroy the system – without defining too many design constraints. For online labs we have to distinguish between two causes of faults: faults caused by the user *("user-based faults")* as well as faults due to the remote lab infrastructure and the communication within this infrastructure on the other hand (*"infrastructure-based faults"*). This task will be executed within the physical system protection unit (PSPU) by filtering all commands. Only commands that will not cause any malfunction will be transferred to the physical system.

In addition, users want to have a fast feedback to their realized solutions. Therefore, possibilities for an automatic *validation* of the user's design within the *GOLDi* infrastructure were implemented as well. Thus, an assessment of the quality of the developed solution can be given immediately. For an efficient usage of the validation system, the reference design and a method to check the student's design against this reference design step by step will be traced by a learning management system (LMS) like "moodle" to analyze any experimental results of the user and generates feedback to improve the given design.

ACKNOWLEDGMENT

Main ideas of this contribution based on the M.Sc. Thesis of Tobias Vietzke [10].

The authors would like to acknowledge the work of Marcus Hamann, Alexander Härtel, René Hutschenreuter and Tobias Fäth for their work within the *GOLDi* framework.

REFERENCES

[1] ICo-op project Website: http://www.ICo-op.eu

[2] DesIRE project Website: http://tempus-desire.thomasmore.be

[3] K. Henke, St. Ostendorff, H.-D. Wuttke, "A Flexible and Scalable Infrastructure for Remote Laboratories - Robustness in Remote Engineering Laboratories", The Impact of Virtual, Remote and Real Logistics Labs - ImViReLL2012 in: CCIS 282 pp. 13–24, Springer Verlag, DOI: 10.1007/978-3-642-28816-6_2, Bremen, Berlin, Heidelberg, February 2012. http://dx.doi.org/10.1007/978-3-642-28816-6_2

[4] K. Henke, St. Ostendorff, St. Vogel, H.-D. Wuttke, "A Grid Concept for Reliable, Flexible and Robust Remote Engineering Laboratories", International Journal of Online Engineering (iJOE), Vol 8 (2012), pp. 42–49, Vienna, December 2012. http://dx.doi.org/10.1109/rev.2012.6293110

[5] K. Henke, St. Ostendorff, H.-D. Wuttke, T. Vietzke, Ch. Lutze, "Fields of Applications for Hybrid Online Labs", International Journal of Online Engineering (iJOE), Vol 9 (2013) - Special Issue REV2013; pp. 20–30, Vienna, May 2013. http://dx.doi.org/10.1109/rev.2013.6502899

[6] K. Henke, H.-D. Wuttke, T. Vietzke, St. Ostendorff, "Using Interactive Hybrid Online Labs for Rapid Prototyping of Digital Systems", International Journal of Online Engineering (iJOE), Vol 6 (2014), pp. 57-62, Vienna, October 2014. http://dx.doi.org/10.1109/rev.2014.6784222

[7] ISO/IEC27001:2013. The internationally acclaimed standard for information security management, http://www.bsigroup.com/en-GB/iso-27001-information-security/ISOIEC-27001-Revision

[8] IEC61508-1. Functional safety of electrical/electronic/programmable electronic safety-related, http://www.iec-normen.de/dokumente/preview-pdf/info_iec61508-1%7Bed2.0%7Db.pdf.

[9] *GOLDi* transmission protocol specification, http://www.tu-ilmenau.de/GOLDi/documentation, TU Ilmenau, 2015.

[10] T. Vietzke, "Sicherheitskonzepte in Remote Labs" (in German), M.Sc. Thesis, TU Ilmenau, 2014.

AUTHORS

Karsten Henke, Tobias Vietzke, Heinz-Dietrich Wuttke, and **Steffen Ostendorff** are with Technische Universität Ilmenau, Ilmenau, Germany. E-mail: [karsten.henke|tobias.vietzke|dieter.wuttke|steffen.ostendorff]@tu-ilmenau.de

This work was supported in part by the European Commission within the program "Tempus", "ICo-op – Industrial Cooperation and Creative Engineering Education based on Remote Engineering and Virtual Instrumentation", Grant No 530278-TEMPUS-1-2012-1-DE-TEMPUS-JPHES. This article is an extended and modified version of a paper presented at the International Conference on Remote Engineering & Virtual Instrumentation (REV2015), held in Bangkok, Thailand, 25 - 28 February 2015. Submitted 23 March 2015. Published as resubmitted by the authors 04 May 2015