# Automatic event handling during robot assisted eye surgery

Miklos Kozlovszky[1,] Roland Dóczi[1], Eszter Jósvai[2],Gernot Kronreif[2]
[1]Biotech Research Center, EKIK, Obuda University, Kiscelli str. 78-80, H-1032, Budapest, Hungary
[2]Austrian Center for Medical Innovation and Technology,Wiener Neustadt,Austria

E-mail: kozlovszky.miklos@nik.uni-obuda.hu

***Abstract*— Medical eye surgery robots are supporting the surgery process in many ways, they primarily can help the surgeon during surgery procedure, they can provide valuable status information about the environment, about the patient and about the forthcoming steps of the surgical process, and by definition they need to protect the patient too. During eye surgery we can differentiate phases of the process and we can identify generic workflows and states for each type of treatments. Transitions between the states or between the workflow steps are triggered by events. Such events can be initiated by any of the surgery the medical expert, by the environment or patient status parameters, or by any components of the system. We have developed an Event handling framework for robot surgery devices. We have evaluated our solution using two different eye surgery robot device. In this paper we are providing information about the framework, its implementations, and detail some evaluation results.**

***Keywords*— *eye surgery robot, event handling, safety framework***

## I. INTRODUCTION

Medical robotics is an enabling technology and recently causing a paradigm shift in medical therapy. Surgical robot assistance can improve surgical outcomes, can decrease patient trauma, and hospital stays, brand new surgical procedures can became feasible as hardly accessible points of the body surrounded by delicate tissue became accessible.

Safety is a critical property of robot systems and especially the surgical robots can be defined as safety-critical systems. Their malfunction, or any errors in their system code can lead easily to severe patient injury or the loss of human life. Systematic methods to specify safety requirements, implement safety features, or verify and validate safety cases are the only ways to realize safe, reliable and robust surgical robot systems [1, 2]. In this paper we present a run-time software safety environment integrated into a modular medical robot software system. Due to the high complexity the majority of recent surgical robots are build up as component-based software systems and the traditional safety analysis techniques can be combined with such component models. Kaiser et al. [3] converts traditional fault tree analysis into components, this method allows partitioning fault tree into multiple components. Jung et al's [4] Safety Framework (SF) provides runtime software safety platform, which can be used for component-based surgical robots. Their main idea is to decompose safety features into reusable safety definitions and mechanisms. In Section I. we give literature overview about existing safety frameworks in medical robotics domain, in Section II. we propose a safety software framework architecture for medical eye surgery robots, in Section III. we introduce its error end event tree, and identify event handling scheme, its event numbering scheme, and the identified error tree. In Section IV we detail the event handling rules and how these rules can be automatically evaluated.

## II. SURGERY ROBOT SAFETY FRAMEWORK

Eye surgery robots are very special, complex and precise devices. Eye surgery procedure involves minimum a surgeon, a nurse, the patient and the eye surgery robot equipped with smart instruments. Our work targets to build up a generic safety framework with well-defined mechanisms for component-based medical eye surgery robots, furthermore to make the concept reusable for other medical surgery robot domains too.

So far our framework has been used for the robots developed within the EurEyeCase project [5]. The EurEyeCase project is focusing on intra−ocular interventions. Such interventions are requiring exceptional skills from the surgeons as the targeted tissues are very delicate and they need very high manipulation and positioning accuracy. The project utilise co-manipulation and tele-operation robotic technologies to enable high precision vitreoretinal eye surgery (such as treatment of retinal vein/artery occlusion through cannulation and epiretinal membrane treatment).

Our developed safety framework includes solutions to monitor and identify errors (and events). Furthermore it contains solutions to automatically react (rules and action definitions) and report according to the situation.

### *A. Error detection*

We have categorized the errors in the system by the following error categories:

- Detectable errors
    - Wrongly identified errors
    - Correctly identified errors

Research funded by The EU Framework Programme for Research and Innovation - Horizon 2020 - Grant Agreement No 645331. 978-1-5386-2064-9/17/$31.00 ©2017 IEEE

- Self-identified errors
- Externally identified errors (externally triggered)
    - Threshold related errors
    - Bad response errors

- Undetectable errors (which are transparent from system view)

### B. *Error handling*

Inside the set of detected errors we can identify a smaller error set which contains recoverable errors, and within this set there are the errors, which are the automatically recoverable errors. Medical surgery robot systems are keen on controlled stop and manual (mechanically assisted) tool retraction can be a feasible solution in most of the cases (shown in Fig.1.).

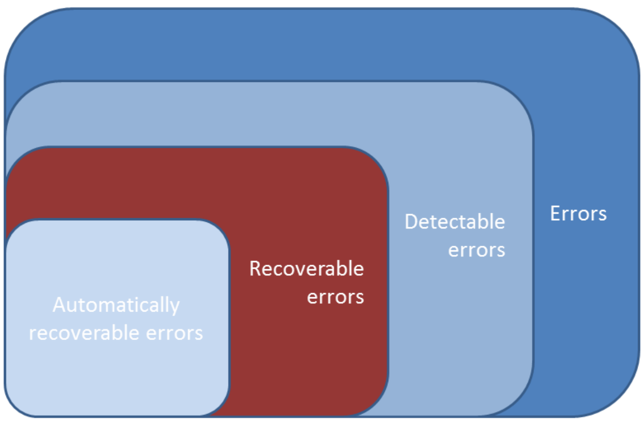


Fig.1.:Generic Error sets

As such systems are providing "just" the so called robot assisted surgery, human supervision is always available. The system is able to detect all the errors, but it needs to show only very limited automatic recovery capabilities. Basically it should inform immediately the medical experts near by the system and should merely suggest alternative options to recover from the error state. Error handling in the safety framework was defined with bottom up and top-down approaches in parallel.

### C. *Workflow of the safety framework development*

System requirement inputs have been distilled from three major sources: from medical experts (surgeons and nurses, from the eye surgery robots, and from standards (e.g.: IEC 62304). Our safety framework is easily adaptable and extendable and provides reusable safety mechanisms to be easily instantiated, adapted to custom needs and it facilitates the testing and certification of the surgery robot systems with tele-manipulation and co-manipulation. The system itself has a modular architecture and follows Component Fault Tree Analysis to classify potential errors and estimate severity of the errors. We have identified state diagrams for two eye surgery procedures namely for Retinal Vein Cannulation and for epiretinal membrane peeling and we have combined these diagrams with the support/maintenance/testing/alarm related states. The defined numbering scheme was the basic foundation to identify and name all the system/software errors and events, and also it enabled us to straightforward merge together the different hardware specific errors and events.

### D. *Capabilities and features*

The safety framework is able to integrate various hardware infrastructure (surgery robot devices with co- and tele-manipulation, instruments) and software modules/components into a common system. Main features of the safety framework are:

- Modular/Component based safety framework
- Extendable with new events, errors
- Extendable with new software modules and components
- Continuously component availability checking
- Distributed error detection (at component level and at system level)
- it reacts to errors in a pre-configured way (E-rules)
- Generic and component specific event/error groups
- Each error has a severity level (calculated according to error type, component source, system state)
- Automatic and manual error resolution (Human factor support)
- Fast and reliable error identification and management
- Extensive logging of error/event messages
- Early identification of possible failures, quality assurance
- Robustness
- It supports error identification, error handling and safe working environment of all the software and hardware components.
- Event/Error patterns and action/reaction patterns

Since robot-assisted eye surgery robot software system is dealing with sensitive information and patient safety, human-in-the-loop technique is involved in order to comply with the safe working environment. Surgeon should is informed during the decision making process of the safety framework. System allows full participation of the surgeon in order to handle critical condition of the system.

### E. *State machine of the full system*

The system has pre-surgery, surgery and post-surgery states. Additionally from all the states the system can go to error state. Furthermore a demo state has been defined, which is a frequently used state for debugging, simulation and testing (shown in Fig.2.).

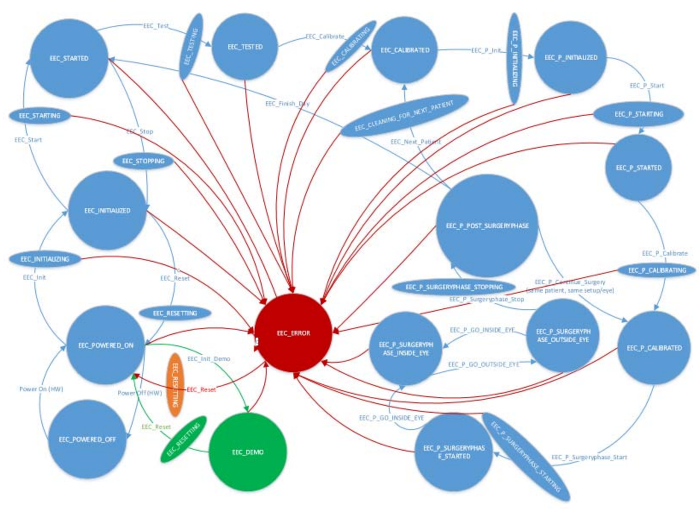

Fig.2.:State machine of the whole system

## III. System Errors and events

### A. *Error and event groups (E-groups)*

We have identified in the system the following E-groups:

- Common E-groups (error and event groups)
  - Errors
    - Communication errors
    - State transition errors
    - Software errors
    - Setup/configuration errors
    - Performance / speed errors
    - Timeout related errors
    - Hardware errors
    - Processing/Functionality errors
  - Events
    - State transition events
    - Communication events
    - Performance / speed errors
    - Hardware events
    - Processing/Functionality events
- Component specific E-groups (error and event groups)

Events and errors are formed in a similar tree-like hierarchical structure:

Group — sub group — E- leaf

Each E-leaf has its own unique identification number and categorized always within an E-group.

### B. *E-numbering scheme structure*

- Error and event numbering scheme requirements
  - Easy to change
  - Easy to append
  - Structured
  - Hierarchical
  - Group-able
  - Usable
  - Numbered
  - Reusable for all the components/modules
- Realized numbering scheme
  - The full E-ID is 8 digits long (4x2 digit).
  - The first two digits is the component ID
  - The second and third two digits are coding the group and sub group IDs
  - The last two digits is the E-leaf number.

With this numbering scheme the source of the E-ID can be tracked, and the E-handling can be hierarchically structured as well.

### C. *Severity levels and their assignment*

Severity levels in the system

- 1 - low
- 2 - moderate
- 3 - high
- 4 - severe
- 5 - critical

Error and event handled similar way in the framework. Severity levels are calculated by the E-Handler service.
Events have the lowest severity level, namely level 1.
Error has higher severity level, namely levels 2-5 (increasing numbers means increasing severity level). Error severity level calculation based on: Error type, source component type and system state. E-ID and its severity level handled independently.

### D. *Basic structure of the safety framework*

- The E-handling of the safety framework

The safety framework has multiple error and event handling (E-handling) levels. At the lowest level each component has its own generic E-handling mechanism defined. Software components have their own try-catch elements, and software exception handling is automatically redirected to an inbuilt component function, which tries to handle the error locally and reports it to the higher level. If the error cannot be resolved locally it is forwarded to the higher level automatically. The system level E-handler is a common centralized component in the framework, which receives all the not resolved errors and events and also all the reports. The E-handler is working together with the Supervisor. The E-handler assigns severity level to each received errors according to the actual system state and system level E-rules. Then it forwards it to the Supervisor, which search the appropriate System level action rule for the error/event report. According to the matched System level action rule, the

Supervisor can instruct the component or the medical expert via the user interface how to act in the situation.

- Component availability monitoring of the safety framework

The component availability monitoring is realized by the Health Monitor service which runs parallel with the E-handler component.
The Health Monitor receives heartbeat messages from each active component. If the heartbeat not arrived before a pre-defined timeout, the Health Monitor initiates a status query of the component. If the component is not answering within the pre-defined status query timeout duration, the Health Monitor sends a component not alive message to the E-handler.

- Feedbacks of the safety framework

Various feedbacks have been identified in the framework. An event or an error is able to use a single or a combination of feedbacks, according to the predefined e-Handling procedures.
Available feedbacks:

  - Audio feedback
    - With pre-defined audio tone/pitch/delay signal patterns.
    - Text-to-speech audio feedback.
  - Visual feedback on the GUI
    - The GUI has a warning mechanism to inform users about system problems (see in Fig.3.)


Fig.3.: Visual feedback example of the GUI (EurEyeCase Assistant)

  - Haptic feedback and control changes in the robot movement
  - Writing inside long term logs

- Logging of the safety framework

Logging is realized at component level, means each component logs its work with timestamp on its own, in a separated logging area.
The basic structure of the safety framework is shown in Fig.4.

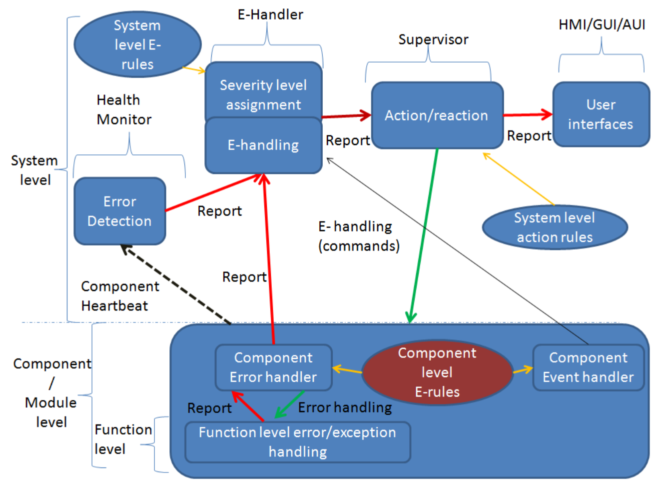


Fig.4.: Safety framework of the EurEyeCase system.

### *E. Sytem level action rules*

Action rules are separated from the source code. Action rules are stored in a form of text based, labeled, hierarchical action description with arbitrary number of parameters. Action rules can be combined together easily, and they can be used as modular templates or building blocks. Its JSON based structure enables workflow visualization and automatic validation. Each action rule has minimum the following parameters:

- E-ID
- Sender component name
- Attributes receiver component name
- Action name
- Action event description
- Action event time
- State change information

Example JSON code of an event action rule:

```
{
 "events": {
  "05520004": {
   "senderModul" : "GUI",
   "attributes":[],
   "receivers":
   [
   {
    "replyEvent": "",
    "targetModule": "Supervisor",
    "timer": ""
   }
   ],
   "nextState": ""
  },
```

## IV. Conclusions

Recent medical robot systems are designed as safety critical systems and they need to fulfill all the environment, functionality and regulation requirements. In this paper we have detailed a software safety framework, which has been used successfully with various eye surgery robots. The evaluation of the system was done successfully with experimental campaigns both with tele-operation and co-manipulation eye surgery robots targeting vein cannulation and membrane peeling. The system is planned to be used in human trials at Q3 and Q4 of 2017. The safety framework includes solutions to monitor, detect errors (and events) and according to the situation it contains solutions to automatically react (rules and action definitions) and report. The framework is capable to do multi-level (component and system level) error-handling to realize a scalable safety system. Its error and event categories build up in a tree like structure. It is using multi-level severity parameters. It is using centralized error severity level calculation (by E-Handler), which is based on: error type, source component type, and system state/status. The Health Monitor component is responsible to component availability monitoring and fault detection. E-handler and Supervisor components are responsible to fault removal and provide ability to sustain safe running environment of the entire software system. According to the request received from the surgeons, we have started to supress the majority of the events and only errors with highest severity levels have been reported. The realized safety framework also applicable to operate outside the robot assisted eye surgery domain, with medical robot systems, that are built using component-based architecture.

## V. FUTURE WORK

In our solution the action rules are separated, and they are not part of the safety framework source code. This enables us to validate and visualize the safety procedures with offline tools. We are planning to realize an automatic safety procedure validation tool set during our development.

## REFERENCES


[1] A. Kapoor, A. Deguet, and Peter Kazanzides, Software components and frameworks for medical robot control, in IEEE Intl. Conf. on Robotics and Automation (ICRA), May 2006, pp. 3813–3818.

[2] Min Yang JUNG, Marcin BALICKI, Anton DEGUET, Russell H. TAYLOR, Peter KAZANZIDES; Lessons learned from the development of component-based medical robot systems; Journal of Software Engineering for Robotics 5(2), September 2014, 25-41 ISSN: 2035-3928.

[3] B. Kaiser, P. Liggesmeyer, and O. Maeckel. A new component concept for fault trees. In Proc. of the 8th Australian workshop on Safety critical systems and software, volume 33 of SCS ’03, pages 37–46. Australian Computer Society, Inc., 2003.

[4] Min Yang Jung, Peter Kazanzides; Runtime safety framework for component-based medical robots; Medical Cyber Physical Systems Workshop (formerly known as HCMDSS (High Confidence Medical De- vices, Software, and Systems)), CPSWeek 2013, Philadelphia, PA, USA, 2013.

[5] EurEyeCase Project Homepage, www.eureyecase.eu , (accessed 20.05.2017)