Requirement and design of safe medical device software architecture

Miklos Kozlovszky[1], Khulan Batbayar[2], Gernot Kronreif[3], Eszter Jósvai[4]

[1,2]*BioTech Knowledge center, Obuda University, Budapest, Hungary*

[3,4]*Austrian Center for Medical Innovation and Technology,Wiener Neustadt,Austria*

kozlovszky.miklos@nik.uni-obuda.hu[1], *batbayar.khulan@phd.uni-obuda.hu*[2], *gernot.kronreif@acmit.at*[3], *eszter.josvai@acmit.at*[4]

Abstract – Medical surgery robots must operate in contact with people. In an ideal world 100% safety can be achieved, the world is not ideal and errors in software and failures of hardware do occur. We have designed and build up a prototype safety framework for surgery robots and trying to implement in an eye surgery robot software. In this paper we provide description of its monitoring and inbuilt safety mechanisms.

INTRODUCTION

In a real world errors in software and failures of hardware do occur, despite duplication of systems and extensive testing. Every effort should be taken to ensure that the system is as safe as possible. The medical robot is designed to fail in a safe manner and come to a controlled halt so that it can be removed and the surgical procedure can be completed manually. Such systems are keen on controlled stop in the event of a failure and manual (mechanically assisted) tool retraction can be a solution in most of the cases. We are focusing on retinal vein occlusion and membrane peeling robot-assisted surgery robot. Our software system is developed in component based fashion, therefore, we are implementing component-based safety framework which checks continuously the availability of the components, collects error messages and reacts to errors in a pre-configured way. We have identified generic and component specific error groups. The source of errors in the system are categorized, and each error type has a severity level. One important component in the error handling is the HealthMonitor component, which monitors the action of all other components. Our error handling is built up from multiple levels; at the lowest level the errors captured by the component and the component is handling the occurred errors with its inbuilt functionalities, the highest error handling level is the system level. When an error occurs, the component tries to fix the problem. The error handling based on the type and severity level of the error. When the error handling at component level is not possible, the error is handled over to the HealthMonitor, which analyses the problem and initiate system level error handling. has to ensure that the control can be fully taken over by the surgeon. Another feature of the HealthMonitor component is detecting the error based on the incorrect or erroneous operation which will trigger automatic messaging to the HealthMonitor component and the HealthMonitor will take action according to the error.

MATERIALS AND METHODS

In component-based software system, one of the popular trend is to integrate traditional safety analysis techniques with a component model. Kaser et al proposed an idea to convert traditional fault tree analysis into components which allows partitioning fault tree into multiple components. Our safety framework follows Component Fault Tree Analysis to classify potential errors and estimate severity of the errors.
We are reusing ideas also from Jung et al's Safety Framework (SF) which provides runtime software safety platform for component-based medical and surgical robot systems. Thus their main idea is to decompose safety features or implementation into reusable safety mechanisms and safety specifications. Our error handling solution adapts, reusable and modular and extensively reuse features such as ease of development, error identification and error handling.

RESULTS

Main design requirements

Requirement 1. Robustness
The error handling should be available and able to serve during the whole system. It should support error identification, error handling and safe working environment of all the other components.
Requirement 2. Ease of development
The error handling solution should be easily configured and used for all system components. The lowest level of the multi-level error handling should be implemented at component level.
Requirement 3. Human factor
Since robot-assisted surgery robot software system is dealing with the sensitive information and patient safety, human-in-the-loop should be involved in order to comply with the safe working environment. Surgeon should be informed during the decision making process of the safety framework. System should allow full participation of the surgeon in order to handle critical condition of the system.
Requirement 4. Monitoring and event handling
The safety framework should be able to monitor each components failure information as well as its own failure information. Every failure in the software system

has to be notified to the Health Monitor and in parallel also logged.

Requirement 5. Early identification of possible failures

Based on the log data, if there is a suspicious event occurs, HealthMonitor should be able to catch the error and find the possible solution. For each failure, there should be a set of appropriate solutions and each solution should be ranked in order to give priority of execution.

Design of the safe medical device component based software architecture

System architecture consists of three main mechanisms which are shown in Figure 1. During the active running of the system, every action of the system will be monitored and logged. If there is an unusual pattern or erroneous action happens, then framework will detect the error and based on the error intensity, impact and the state of the error.

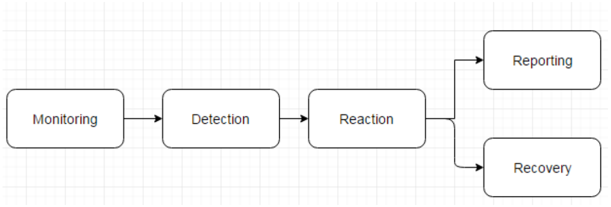

Figure. 1. Mechanism view of the safe component architecture

Safe medical software architecture shown in Figure 2 is a layered architecture, therefore, error handling is built in layered fashion. Each component has an error handling feature built inside and try to solve the problem inside using exception handling and report it to the HealthMonitor component. If the error handling in the component level is not possible, then component will report to the HealthMonitor to take an action. In this case, the HealthMonitor will search for the error information and take a reaction.

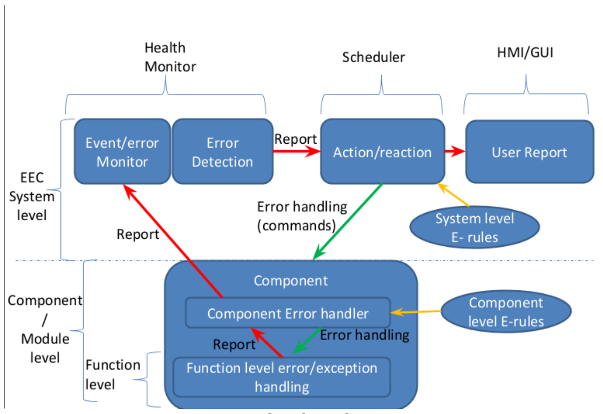

Figure. 2. Basic structure of safety framework

Compatibility with safety standards and regulation body

One of the major part of the medical robot system is that it has to be approved by regulatory authorities such as Food and Drug Administration (FDA) depend on the market of the medical device. Since the main goal of these standards are to ensure correct, precise working of the robot and avoid any harm for the patient as well as the surgeon. We are designing the software architecture in a safe manner, therefore we are trying to develop software as safe as possible. Further compatibility will be studied thoroughly.

CONCLUSION

Medical robot systems are safety critical system which are specially designed according to the environment, functionality and regulation requirement. The design of the requirement is based on the component-based software systems and safety engineering of the medical robot software. The fault analysis of the HealthMonitor component will use Component Fault Tree Analysis to estimate the failure and possible reaction of the system. The HealthMonitor component is responsible to fault detection, fault removal and ability to sustain safe running environment of the entire software system. The HealthMonitor component can be applicable to the other domain of the medical robot system which are built using component-based architecture.

REFERENCES

[1] 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.

[2] 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.

[3] 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.

[4] 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.