Automatic Translation Form Requirements Model into Use Cases Modeling on UML

Haeng-Kon Kim[1] and Youn-Ky Chung[2]

[1] Department of Computer Information & Communication Engineering, Catholic University of Daegu, Korea
hangkon@cu.ac.kr
[2] Department of Computer Engineering, Kyung Il University, Republic of Korea
ykchung@kiu.ac.kr

Abstract. Given the investment organizations are making in use cases and the increasing use of computers to control safety-critical applications, the research on integrating use cases and safety consideration is highly demanded. However, currently the approaches for incorporating safety analysis into use case modeling are scarce. In this paper, we present an approach to integrating safety requirements into use case modeling. We demonstrate how deductive and inductive safety techniques can be used hand-in-hand with use cases. An application of the proposed approach facilitates early hazard identification and assessment, as well as elicitation and tracing of safety requirement though the entire development process. The proposed approach is illustrated by a realistic case study – a liquid handling workstation.

Keywords: UML, use cases, safety analysis, safety requirements, CBD.

1 Introduction

The Unifying Modeling Language (UML) [1] is gaining increasing popularity and has become de facto industry standard for modeling various systems many of which are safety-critical. UML promotes use case driven development process [1] meaning that use cases are the primary artifacts for establishing the desired behavior of the system, verifying and validating it. Elicitation and integration of safety requirements play a paramount role in development of safety-critical systems. Hence there is a high demand on methods for addressing safety in use case modeling. In this paper we propose an approach to integrating safety requirements in use cases.

It is widely accepted that building in safety early in the development process is more cost-effective and results in more robust design [2,3,4]. Safety requirements result from safety analysis – a range of techniques devoted to identifying *hazards* associated with a system and techniques to eliminate or mitigate them [3,4]. Safety analysis is conducted throughout the whole development process from the requirements conception phase to decommissioning. However, currently the approaches for incorporating safety analysis into use case modeling are scarce.

To make an integration of safety analysis and use cases complete we need to investigate the use of not only deductive techniques but also inductive safety techniques.

Indeed, to conduct safety analysis at the early stages of the development we need deductive techniques able to assess hazards at functional level. We describe a general structure and behavior of control systems which employ manual reconfiguration for error recovery. We give a brief overview of the case study – the liquid handling workstation. We finally present our approach to conducting hazard analysis at the early stages of development by embedding the method into use cases.

2 Dependable Control Systems

2.1 Control Systems

In this paper we focus on modeling of dependable [2] control systems. The basic building blocks of a control system are presented in Fig.1. An *application* is a physical entity whose operation is being monitored and controlled by a *computer-based controller*. The behaviour of the application evolves according to the involved physical processes and control signals provided by the controller. The controller observes behaviour of the application by means of *sensors* – the devices that convert application's physical quantities into electric signals to be input into the computer. On the basis of data obtained from the sensors the controller manipulates states of *actuators* – devices that convert electrical signals from the output of the computer to physical quantities, which control the function of the application.

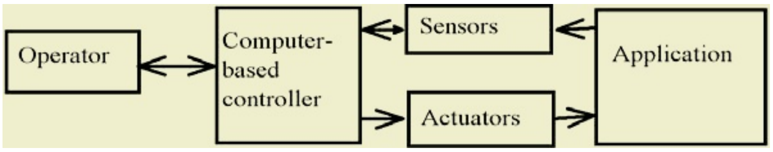


Fig. 1. A general structure of a control system

2.2 Manual Reconfiguration

As we discussed above, system's components are susceptible to various kinds of faults. A fault manifests itself as *error* – an incorrect system's state [11]. The controller should detect these errors and initiate error recovery. In this paper we investigate manual reconfiguration, as it is the most commonly used error recovery mechanism [11]. The manual reconfiguration is performed by an operator. The behaviour of the control systems with manual reconfiguration for error recovery follows a general pattern graphically represented in Fig.2. Initially the system is assumed to be fault free. The operator starts the system functioning and upon successful initialization the system enters an automatic operating mode. Within the automatic mode the system executes a standard control loop provided no error is detected.

Upon detection of an error the system leaves the automatic mode and reverts to the operator's control. An operator assesses damage caused by an error and executes a required error recovery procedure. As a result of error recovery the operator might resume

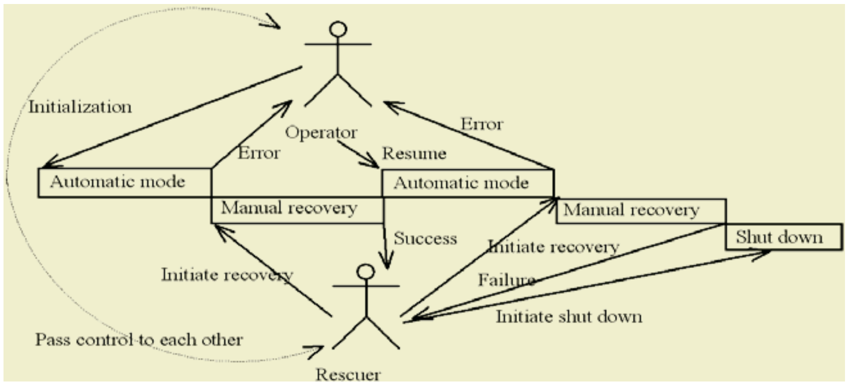


Fig. 2. Illustration of manual reconfiguration

the automatic mode or execute shutdown. The human operator plays two different roles while controlling a system with manual reconfiguration. On one hand, the operator initiates and monitors system functioning in the automatic mode, on the other hand s/he is a trouble-shooter performing error recovery in the manual mode. Hence, the operator should be represented by two actors in the use case diagram: the first is *operator* and the second is *rescuer*. The general form of the use case diagram is given in Fig.3.

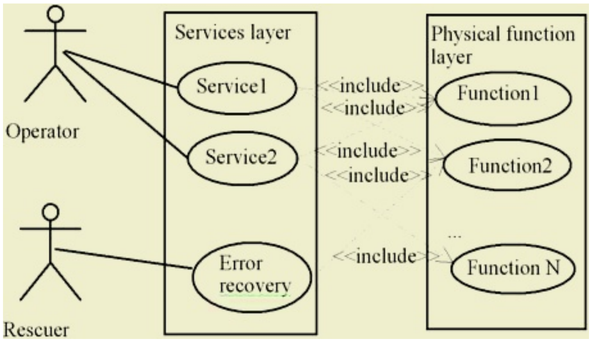


Fig. 3. Schematic representation of use case diagram

3 Use Case Modeling and the Requirement Risk Assessment Method

3.1 The Functional Hazard Assessment Method

Essentially, a use case is a coherent unit of system's functionality [1]. Hence to smoothly incorporate safety analysis in use case modeling we need a technique, which is able to identify hazards on a function level. In this paper we investigate how the Requirement Risk Assessment (RRAM) method can be applied to use cases to identify hazards and assess them. The RRAM allows us to identify hazards at a functional level and correspondingly derive safety requirements. The RRAM process consists of five steps [5]:

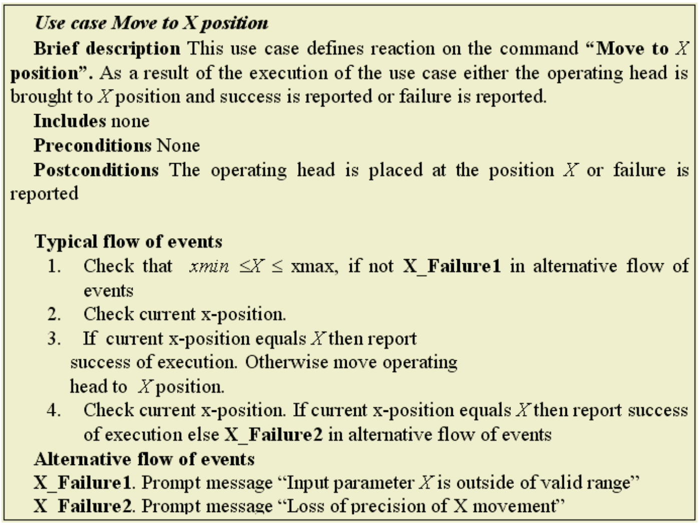
1. Identification of all functions associated with the level under study.
2. Identification and description of failure conditions associated with these functions.
3. Determination of the effects of the failure condition.
4. Classification of failure effects on the system.
5. Assignment of requirements to the failure conditions to be considered at the lower level.

An identification of failure conditions can be done systematically by applying the following guidewords [7]:

- *Loss of function*
- *Function provided when not required*
- *Incorrect operation of function.*

We argue that use case model constitutes a suitable description of system's functionality. Indeed

- use cases clearly define system's functions
- use case diagrams explicitly show interdependencies between use cases by means of associations
- the proposed layered structure of use case diagram clearly identifies levels of abstraction and therefore, allows us to apply the RRAM method iteratively, layer after layer.



Use case Move to X position

Brief description This use case defines reaction on the command "**Move to** *X* **position**". As a result of the execution of the use case either the operating head is brought to *X* position and success is reported or failure is reported.

Includes none

Preconditions None

Postconditions The operating head is placed at the position *X* or failure is reported

Typical flow of events

1. Check that $xmin \leq X \leq$ xmax, if not **X_Failure1** in alternative flow of events
2. Check current x-position.
3. If current x-position equals *X* then report success of execution. Otherwise move operating head to *X* position.
4. Check current x-position. If current x-position equals *X* then report success of execution else **X_Failure2** in alternative flow of events

Alternative flow of events

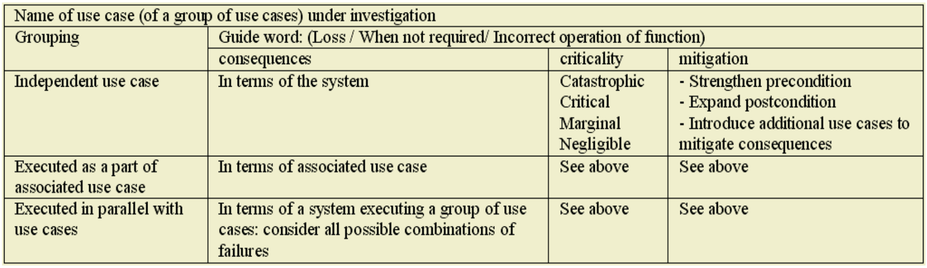
X_Failure1. Prompt message "Input parameter *X* is outside of valid range"

X_Failure2. Prompt message "Loss of precision of X movement"

Fig. 4. Example of physical-layer use case description

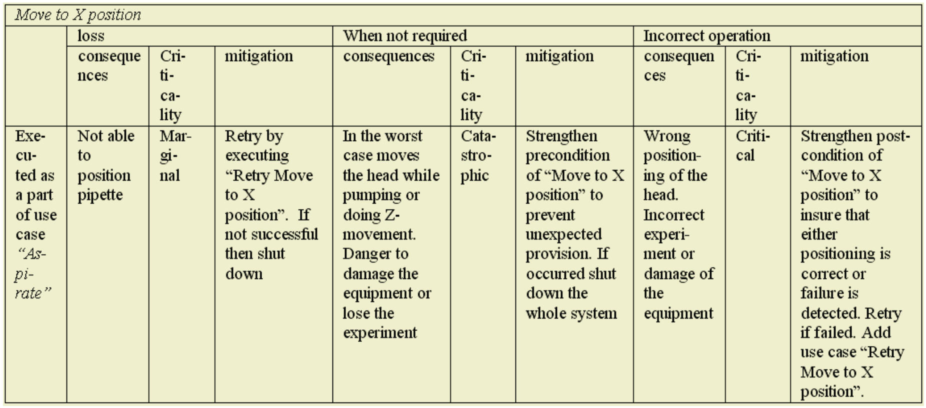
For the control systems with manual reconfiguration discussed above the RRAM starts from analyzing use cases at the server's layer. By systematically applying guidewords we describe consequences, criticality and mitigation of the corresponding failures for each use case. Such an analysis of individual use cases suffices if the use cases are independent. However, if there are use cases associated with each other or the use cases, which might be executed in parallel, then we also need to consider the effect of multiple failures. In this case various combinations of failures should be explored and hazards induced by them assessed.

Table 1. The template for conducting the RRAM



Name of use case (of a group of use cases) under investigation			
Grouping	Guide word: (Loss / When not required/ Incorrect operation of function)		
	consequences	criticality	mitigation
Independent use case	In terms of the system	Catastrophic Critical Marginal Negligible	- Strengthen precondition - Expand postcondition - Introduce additional use cases to mitigate consequences
Executed as a part of associated use case	In terms of associated use case	See above	See above
Executed in parallel with use cases	In terms of a system executing a group of use cases: consider all possible combinations of failures	See above	See above

Table 2. An excerpt from an application of the RRAM to a use case



Move to X position									
	loss			When not required			Incorrect operation		
	consequences	Criticality	mitigation	consequences	Criticality	mitigation	consequences	Criticality	mitigation
Executed as a part of use case *"Aspirate"*	Not able to position pipette	Marginal	Retry by executing "Retry Move to X position". If not successful then shut down	In the worst case moves the head while pumping or doing Z-movement. Danger to damage the equipment or lose the experiment	Catastrophic	Strengthen precondition of "Move to X position" to prevent unexpected provision. If occurred shut down the whole system	Wrong positioning of the head. Incorrect experiment or damage of the equipment	Critical	Strengthen postcondition of "Move to X position" to insure that either positioning is correct or failure is detected. Retry if failed. Add use case "Retry Move to X position".

3.2 Example of an Application of the RRAM

The use cases at server's layer of Fillwell are independent of each other. An application of the RRAM has shown that the most serious consequences are induced by the incorrect provision of the use cases "Aspirate" and "Dispense". The loss of them deadlocks the system. Finally, an inadvertent provision of them is assumed to be unfeasible. The next iteration of the RRAM focuses of the analysis of use cases of the component's layer. An excerpt from the analysis of the use case *"Move to X position"* is presented in Table 2. The RRAM conducted at the component's layer is more complex, because we need to consider various dependencies and associations. However, an application of the template presented in Table 1 significantly simplifies the analysis: it allows us to link directly failures of "Aspirate" and "Dispense" to the failures of component's layer use cases.

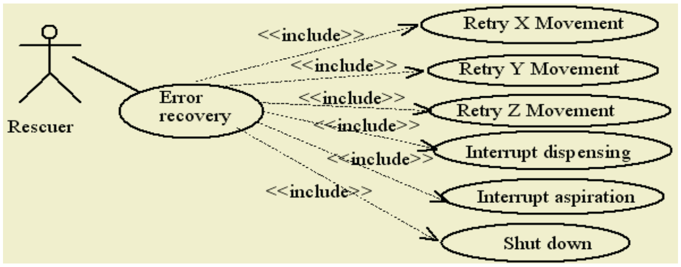


Fig. 5. Elaborated representation of "Error recovery"

3.3 Validating Proposed Approach

An application of the RRAM as proposed above allowed us to significantly improve requirements of Fillwell. For instance, as a result of preliminary hazards identification only safety requirements *SR1* and *SR2* have been stated. The additional safety requirements *SR3* and *SR4* are discovered as a result of the proposed application of the RRAM:

SR1. The operating head must stay inside allowed margins, i.e. it should not be moved outside of minimal and maximal allowed positions.
SR2. The amount of liquid in the pipette must not exceed the volume of the pipette
SR3.While the operating had is moving along Z-axes movement along X,Y-axes is prohibited
SR4. Any movement of operating head while the pump is operating is prohibited

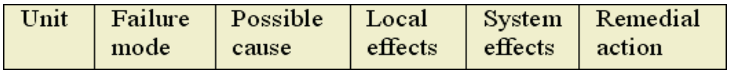
The proposed approach encouraged us to produce an explicit description of hazards mitigation. As a result new functional requirements have been added. Namely, we elaborated on the use case *"Error recovery"* to define the physical functions to be executed for error recovery. The refined part of the use case diagram is shown in Fig. 9. Moreover, we refined the existing description of the use cases. For instance, we added the precondition "Ensure that the request to move to position X is placed and neither Z-movement nor pumping is performed" to the specification of the use case "Move to X position" and expanded the postcondition to model possibility of failure of the use case. Furthermore, the proposed approach allowed us to establish a link between the use cases and safety requirements. This is a valuable asset since it facilitates a straightforward tracing of requirements through the whole development process.

4 Incorporating RMEA into Use Case Modeling

In the previous section we have shown how to incorporate a deductive safety technique in use case modeling. We demonstrated how an application of the RRAM method facilitated derivation of system's requirements at functional level. However, as system development progresses, we learn about behavior of components, which are executing these functions as well as contributing to their failures. This information is usually obtained by an application of inductive safety techniques such as Risk Modes and Effect Analysis (RMEA). Hence, to incorporate safety analysis in the entire process of requirements engineering, we also need to merge use case modeling and an inductive safety technique. Next we present our approach to incorporating results of RMEA in use cases.

4.1 Extending RMEA

RMEA is an inductive analysis method, which allows us to systematically study the causes of components faults, their effects and means to cope with these faults [3]. RMEA step-by-step selects the individual components of the system, identifies possible causes of each failure mode, assesses consequences and suggests remedial actions. The results of RMEA are usually represented by tables of the following form:



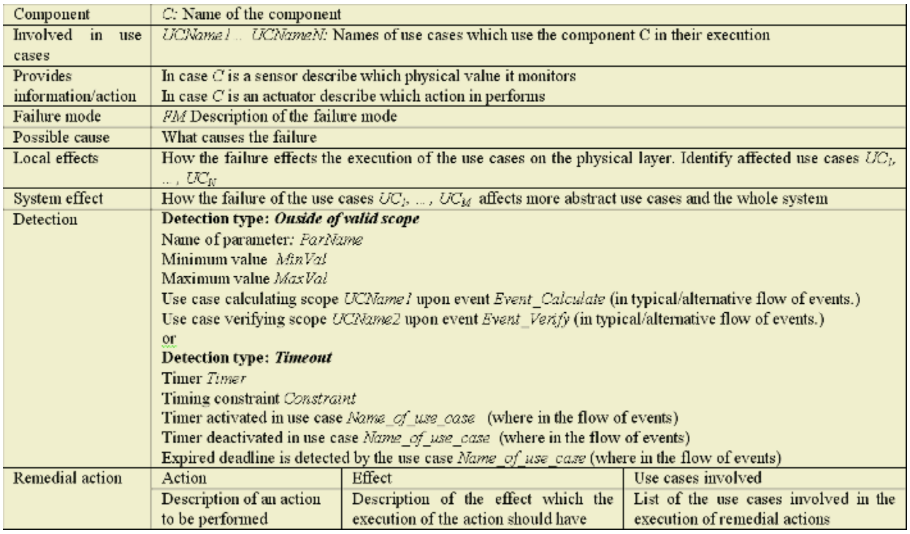
Unit	Failure mode	Possible cause	Local effects	System effects	Remedial action

RMEA provides us with important information about failure modes of components and gives guidance on how to cope with these failures. Usually this information is hardware-oriented. Hence its translation into software requirements demands a good understanding of software-hardware interaction and physical processes evolving in the system. This task is challenging and therefore, needs to be assisted by software designers and safety engineers. Observe that use cases constitute an effective tool for interdisciplinary communication. Therefore, use cases can be used for establishing common communication framework between software designers and safety engineers. Next we propose a structured approach to extending a traditional representation of the results of RMEA to facilitate discovery of software requirements. The main idea behind the proposed approach is to encourage safety engineers to explicitly express how controlling software should detect errors caused by component's faults and which remedial actions software should initiate for error recovery. Unfortunately currently such a description is often confined by a phrase "modify software to detect error and initiate error recovery". It is rather unlikely that such a description will result in the traceable software requirements unless additional clarifications will be provided by a safety engineer. Therefore, while analyzing failure modes of each component we encourage safety engineer to describe

- which use cases employ the component under investigation
- what kind of information the component supplies to the use cases or which action it provides
- which use cases are involved in detection of failure of the component
- which use cases are involved in recovery from component's failure.

In Table 3 we demonstrate how the proposed extension can be incorporated in the traditional representation of the results of RMEA. Our previous experience shows that usually there are two types of most frequently used detection mechanisms: *timeout* and *a value outside of the valid scope*. For each detection mechanism we incorporate a description, which directly allows us to generate traceable software requirements. For instance, for the detection by capturing incorrect parameter range, called *outside of valid scope,* we generate the following requirements:

Table 3. Template for representing results of extended RMEA



Component	*C:* Name of the component		
Involved in use cases	*UCName1 .. UCNameN:* Names of use cases which use the component C in their execution		
Provides information/action	In case *C* is a sensor describe which physical value it monitors In case *C* is an actuator describe which action in performs		
Failure mode	*FM* Description of the failure mode		
Possible cause	What causes the failure		
Local effects	How the failure effects the execution of the use cases on the physical layer. Identify affected use cases UC_1, .. , UC_N		
System effect	How the failure of the use cases UC_1, ... , UC_M affects more abstract use cases and the whole system		
Detection	**Detection type: *Ouside of valid scope*** Name of parameter: *ParName* Minimum value *MinVal* Maximum value *MaxVal* Use case calculating scope *UCName1* upon event *Event_Calculate* (in typical/alternative flow of events.) Use case verifying scope *UCName2* upon event *Event_Verify* (in typical/alternative flow of events.) or **Detection type: *Timeout*** Timer *Timer* Timing constraint *Constraint* Timer activated in use case *Name_of_use_case* (where in the flow of events) Timer deactivated in use case *Name_of_use_case* (where in the flow of events) Expired deadline is detected by the use case *Name_of_use_case* (where in the flow of events)		
Remedial action	Action	Effect	Use cases involved
	Description of an action to be performed	Description of the effect which the execution of the action should have	List of the use cases involved in the execution of remedial actions

5 Conclusions and Future Work

Given the investment organizations are making in use cases and increasing use of computers to control safety-critical applications, the research on integrating use cases and safety consideration is highly demanded. In this paper we presented an approach to incorporating safety requirements into use cases. We demonstrated how the results of deductive and inductive safety techniques can be used to discover and strengthen functional and safety requirements. The proposed approach has been illustrated by a realistic case study – a liquid handling workstation Fillwell. In our previous work we demonstrated how to incorporate various safety techniques into formal approaches to system specification and development [12,14,15]. Moreover, we succeeded in formalizing UML class and state diagrams [8,10]. In our future work it would be interesting to integrate formal design techniques with use case driven approach proposed in this paper.

References

[1] John Cheesman, John Daniels, UML Components A Simple Process for Specifying Component-Based Software, Addision-Wesley, 2001

[2] J.-C. Laprie. "*Dependability: Basic Concepts and Terminology*". Vienna, Springer-Verlag, 1991.

[3] MG, OMG Unified Modeling Language Specification Version 1.5, http://www.omg.org/uml , 2003.
[4] K. Allenby, T. P. Kelly. ”Deriving Safety Requirements using Scenarios”. In *Proc. of the 5th IEEE International Symposium on Requirements Engineering (RE'01)*, Toronto, Canada, 2001, pp.228-235.
[5] Gunter Preuner, Michael Scherfl, "Integration of Web Services into Workflow through a Multi-Level Schema Architecture," Proceeding of Advanced Issues of E- Commerce and Web-Based Information Systems, Fourth IEEE International Workshop on, pp. 51-60, Jun. 2002.
[6] Ruben Prieto-Diaz, "The Common Criteria Evaluation Process," Commonwealth Information Security Center Technical Report, 2002.
[7] PMI, A guide to the Project Management Body of Knowledge(PMBOK Guide 2000 Edition), at URL: http://www.pmi.org , 2000.
[8] Troubitsyna. “Integrating Safety Analysis into Formal Specification of Dependable Systems”, in *Proc. of Annual IEEE Workshop on Fault-Tolerant Parallel and Distributed Systems.* Nice, France, April 2003.