An Integrated Approach for Software Safety Analysis

Fayokemi Ojo [1], Tangee Beverly[2], and W. Eric Wong[3]

[1]Department of Computer Science and Electrical Engineering, University of Maryland, Baltimore County,
[2]Department of Computer Science, Elizabeth City State University
[3]Department of Computer Science, University of Texas at Dallas

Abstract - **Software safety analysis as we know it does not look at functional and safety requirements simultaneously. Functional specifications are typically modeled with a UML state diagram and a Fault Tree Analysis (FTA) is generally used to delineate the causes of hazards. By using an integrated approach, a more thorough look into possible system failures is produced. In other words, creating a model that shows how a system is designed to work, while also describing where issues can occur, leads to a better, safer system. A case study is done using a stair-climbing wheelchair. Functionality and safety hazards are modeled after the case study. The immediate research goal is to create a new model that combines the case study diagrams together. This goal was met and the new model gave more insight into possible failures than the FTA and UML state diagram did individually. The next step is to use this method on a different system. Eventually, a model will be created that can be used with any safety-critical system.**

Keywords **–Software Safety, Software Reliability, Fault Tree Analysis, UML Statechart Diagram**

I. INTRODUCTION

Software has been built into more and more products and systems used by the public. In most cases, the integration of software helps make tasks easier for human beings. In some cases, however, software can cause unforeseen problems and possibly even put the user in danger. There are some safety processes to help prevent a risky event. One of them is the fault tree concept, which a fault tree has the capability of providing useful information concerning the probability of a failure and by which a failure can occur. State machines are a formalism that has been widely applied to the functional specification of software. Operational or intended behavior of the system often focuses with the modeling statecharts[2]. Engineers with a variety professional background use fault trees and state machines.

II. METHODOLOGY

We started our research by looking into FTA and UML statechart diagrams. Once we had a basic understanding of how these diagrams worked, we started researching how the two could be combined. We found amalgamation was possible by integrating gates and events from the fault tree into the statechart. The next step was finding what these diagrams could be modeled after and a bulk of this research was spent looking for a case-study. This entire project is centered around software safety, so we had to find a safety-critical system. The goal was to use newer technology that could still be in development. This way we could see if the modified state machine actually made the system safer. However, time limitations kept us from having a contract with a manufacturer in order to get all the information required for the modified state machine. We altered our initial goal to include newer technology that was available to the public, so that more information on the system would be available. Eventually, the stair-climbing wheelchair was selected as our case study.

III. Fault Tree Analysis

In 1962, Bell Telephone Laboratories developed fault tree analysis for the U.S. Air Force. Every major failure is represented in a safety critical system. It is another technique for reliability and safety analysis [3]. Fault tree describes how the individual fault components combine to result in an undesirable system behavior or catastrophic failure [2]. A fault tree is composed of nodes, edges, and gates. A gate is a logical connective, nodes are events that are considered as gate inputs, and lastly edges connect nodes to gates [2]. There are various types of gates to be used in the fault tree, but in this paper, we focused on a few in our research. Fault trees has many benefits it creates a visual record of a system that shows the logical relationships between events and causes lead to potential failures [1]. It advises others to immediately understand the result of your analysis and pinpoint the weakness in the design and identify errors [1]. Fault tree diagram is used to help design quality tests and maintenance procedures [1]. Fault tree analysis is definitely useful in engineering, especially in industries where a potential failure can have some catastrophic consequences such as nuclear power, automotive, and aeronautics [1]. However, in a complex system, a fault tree analysis can also be used during software development to debug [1]. Below is a chart of the different gate symbols (figure 1). There are a lot more tree blocks for a fault tree than the ones down below and we did use all the blocks in our research [4].

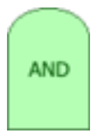
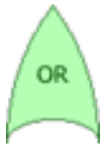
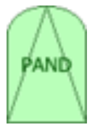
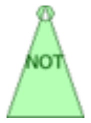
	AND	All input events TRUE
	OR	At least one input event TRUE
	PRIORITY AND	The output event occurs if all input events occur in a specific sequence
	NOT	The output event occurs if the input event does not occur.

Figure 1. Fault Tree Blocks

IV. UML Statechart Diagram

A UML statechart diagram is an interactive model that depicts the functional specifications of a system and it is composed of states, transitions, and events [2]. A statechart diagram defines the states; it is used to model the lifetime of an object's existence. States are components and subcomponents of a system. Transitions is the movement between states from one state to another state. Lastly, events trigger these transitions [2]. Statechart diagram defines the flow of control from one state to another state. They are also used for forward and reverse engineering of a system. The following points should be clarified before drawing a statechart diagram: identify the objects, states and events. The first initial state is a small black circle in the diagram which where the process starts. Next couple states are arrived for events and responsible for the state changes of order object. Harel introduced statecharts as an extension to state machines. The goal was to represent the behavior of complex systems, in a comprehensible form without suffering from explosion in the number of states and edges [2]. Statecharts were intended to be formalism or a language that can be compiled and executed not a specification tool. Harel initial did not define semantics for the statecharts but as they became more popular and useful, semantics were introduced [2].

V. Case Study

As mentioned in a previous section, our case study is a stair-climbing wheelchair. There are many types of stair climbing wheel chairs, but this case study looks specifically at battery-powered wheelchairs with a conveyor belt. The first question we asked ourselves while analyzing this wheelchair is, what are something's that can wrong? In other words, how can the wheelchair fail? To answer that question, we realized that we also had ask ourselves, "how does this wheelchair function?" That is when we realized that there is a real benefit from putting functional and safety requirements together. We researched different brands of stair-climbing wheelchairs, but gather most of information from the Scewo brand. We watched their videos to see how the wheelchair worked. We constructed a fault tree and a state diagram based on our observations.

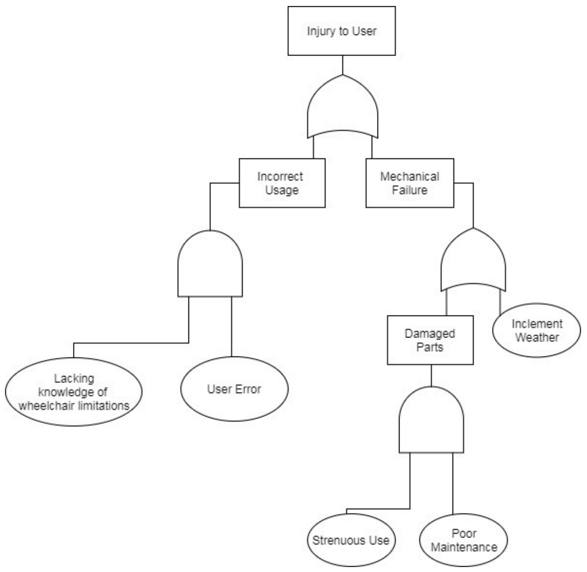


Figure 2. The Fault Tree model for stair climbing wheelchair

We kept the fault tree relatively simple and we only use "and" and "or" gates. We found that many stair-climbing wheelchairs are not meant to climb all staircases. There is typically a slope limit and a maximum step height. If the user is not careful it can cause safety hazards. The user also needs to make sure that the wheelchair is properly charged and maintained in order to avoid safety hazards. The fault tree we created portrays a general idea of possible undesired behavior that can occur when using this wheelchair. The statechart we constructed only shows the desired behavior when the chair is in climbing mode.

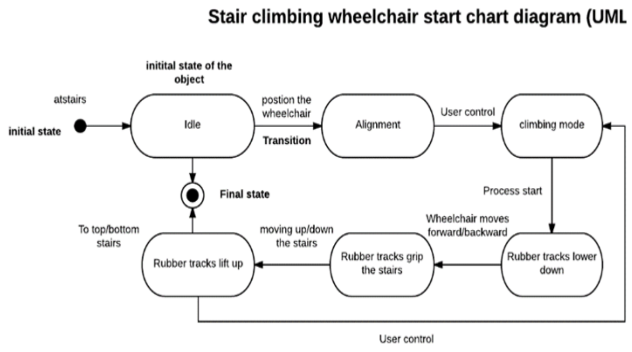


Figure 3. The first state chart diagram of stair climbing wheelchair.

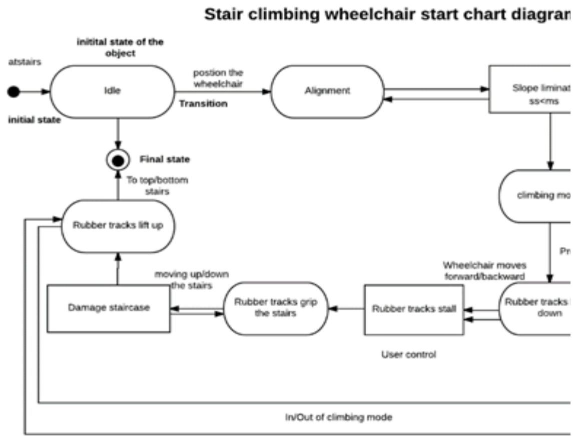


Figure 4. The modified statechart stair-climbing wheelchair.

We modified the first statechart by integrating some gates and events from the fault tree into the statechart. Due to the fact that we did not have access to how software was used in the wheelchair, the result is not as detailed as we would have liked. And because of the contrasting ways we created the fault tree and the state chart, we could not integrate every event and gate. However, the modified statechart shows that desirable and undesirable traits can be integrated and that was one of the biggest goals of this project.

VI. RESULT & CONCLUSION

In this paper, we analyzed how functional and safety requirements could be integrated. We found not only is it possible, but doing this gives a more in-depth look into possible hazards that by looking at these two things separately. If systems are designed with safety features in mind rather than just tested for problems later in the system development lifecycle, then money, time and even lives could be saved.

VII. Acknowledgment

Fayokemi Ojo and Tangee Beverly conducted this work during the 2017 REU (Research Experiences for Undergraduates) summer program under the supervision of Professor W. Eric Wong at the University of Texas at Dallas. The National Science Foundation funds the program. More details of our REU program can be found at http://paris.utdallas.edu/reu.

VIII. REFERENCE

[1] "Fault Tree." Fault Tree Diagram - What is a Fault Tree and Fault Tree Analysis? N.p., n.d. Web. 21 July 2017.
[2] Ariss, Omar El, Dianxiang Xu, and W. Eric Wong. "Integrating Safety Analysis With Functional Modeling." IEEE Transactions on Systems, Man, and Cybernetics - Part A: Systems and Humans 41.4 (2011): 610-24. Web.
[3] Publishing, ReliaSoft. "Fault Tree Analysis." Weibull.com -- Free Data Analysis and Modeling Resources for Reliability Engineering. N.p., n.d. Web. 21 July 2017
[4] *Fault Tree Block Types*. N.p., n.d. Web. 27 July 2017.