Software Reliability Assessment of Safety Critical System Using Computational Intelligence

Bharathi R, PES University, Bengaluru, India
Selvarani R, Alliance University, Bengaluru, India

ABSTRACT

In the recent past, automotive industries are concentrating on software controlled automatic functions for its safety operations. The automotive safety and reliability lie in its design, construction, and software implementation. To assess the software reliability, the hidden design errors are classified and quantified. The temporal characteristic of numerical error is analyzed and its probabilistic behavior is explored using a novel framework called software failure estimation with numerical error (SFENE). Here, a model is devised to assess the probability of occurrence of the numerical error and its propagations from the initial to various other states using a Hidden Markov Model. It is seen that the framework SFENE supports classifying and quantifying the behavior of numerical errors while interacting across its system components and aids in the assessment on software reliability at design stage. The sensitivity and precision are found to be satisfactory. This attempt will support in the development of cost effective and error free safety critical software system.

KEYWORDS

Failure, Fault, Design Error, HMM, Propagation, Safety Critical

1. INTRODUCTION

The software controlled automotive systems comes under the category of safety critical system and in the recent past it has gained lot of attention due to the emerging challenges in its design. These systems uses up to 100 electronic control units (ECUs) communicating more than 2500 signals by using up to 5 different bus systems (Eyisi et al., 2013). These ECU’s control and monitor many vehicle subsystems such as anti-lock braking system (ABS), chassis control, vehicle stability, and engine control. A system is called a safety critical system when the human safety is at stake and dependent upon mechanical/electrical/automatic systems’ operation. In this paper the concentration is on the controlling software, which controls the safety critical system, but not on the any associated element meant for system operation.

The design and development of software module responsible for operational control of any system is a challenge for its precision. The high precision is expected in terms of reliability, functional safety, real-time behavior, resource sharing in operation etc. in order to ensure, zero defect system in any critical operations (Mössinger, 2010). Software failures in automotive systems may cause severe damages or loss of life. In recent past one-third of the car recalls was witnessed and understood that the failure is due to error in control software, which lead to system failure (Boulanger & Dao, 2008). Here our analysis focused on reliability of the safety critical software system.

DOI: 10.4018/IJSSCI.2019070101

Copyright © 2019, IGI Global. Copying or distributing in print or electronic forms without written permission of IGI Global is prohibited.
Software reliability is defined as the probability that the system will perform its intended functionality under specified design limits. Software reliability techniques are aimed at reducing or eliminating failures of software systems (Roshandel, 2004). In spite of rigorous validation and testing procedures, there are many accidents occurred leading to catastrophic failures and life-threatening situations. To circumvent such incidents research on failure analysis of safety critical system is of high importance to ascertain its reliability.

Designing for reliability is a systematic approach, which must be established in the early design stages with the sole objective of reducing the number of failures (Garg, 2015; Garg, Rani, & Sharma, 2014). This can be accomplished by the extensive implementation of design analyses, evaluations, testing and simulation techniques that can optimize and assess reliability. The objectives of the reliability assessment are: to identify potential hazards, to translate reliability requirements as functional requirements and to provide design assessment in the on-going design. Lutz et al have recommended that include validity checking for input, error handling, overflow protection and behavior of the system under unexpected conditions as requirements for defensive design (Lutz, 1993). In pursuing towards this, we have proposed a framework to study the probability of occurrence of software failures. This helps to assess the reliability and the number of errors associated in the controlling software. This leads to understand the relationship between the reliability and safety of the controlling software. Software failure occurs due to various reasons, one of the reason is due to missteps made during design phase, that are often challenging to visualize, categorize and debug (Xie & Lyu, 1996). The software reliability is estimated using the failure statistics observed in various situations of system operation. The observations are made at various critical points of occurrence for devising a more accurate failure prediction system.

In automotive safety-critical systems, the software controls and monitors critical vehicle functions namely steering and braking. Though these systems are designed with safety features, unpredicted interactions among the software, hardware, and environment might lead to potentially hazardous situations. The potential causes for software failures are software errors, and support software errors and hardware failures (Czerny, D’Ambrosio, Murray, & Sundaram, 2010). Software errors might be due to either inconsistent or incomplete requirements, design or coding.

Many articles have been published on the predictions and estimation of software impairments. But only few have detailed on the link between these impairments. The objective of this work is to fulfill the gap by analyzing the types of error that causes software failures by injecting fault into the safety critical system. In general, faults are local to one component; whereas errors and failure propagate from one component to another. Further to analyze how errors are propagated and transformed across system components. This analysis helps to explore the association between the errors and the corresponding failures. SCSS invariably uses floating-point variables, which are prone to round off errors, truncation errors etc., which are, categorized as numerical error. These numerical errors have propagation characteristics (Darulova & Kuncak, 2014). Altinger et al. have listed the most common type of bugs discovered in automotive systems as Boolean logic error, missing condition, variable type cast error including wrong variable declarations and mismatch in Boolean and bit wise logic (Altinger, Dajsuren, Siegl, Vinju, & Wotawa, 2016). In this paper experimental studies are carried out to explore the probabilistic nature of the software behavior when there is a numerical error and to understand component behavior as well. The objective of our work is to build a model to capture the behavior of the controlling software with impairments, by observing the execution sequence of critical system operations. Typically, we want to explore and find answers on the exact type of software failure that is most likely to occur and to find the corresponding error propagations. The uncertainty associated with the execution of the software with impairments can be illustrated using Hidden Markov Model (HMM), to facilitate probabilistic modeling.

HMM is used to describe the evolution of observable events, which depends on certain internal factors that are not directly observable. It is well known that the operation of a safety critical system is controlled by software, which are hidden. We call the observable event as “observation symbols”
Modernization of Healthcare and Medical Diagnosis System Using Multi-Agent System (MAS): A Comparative Study
www.igi-global.com/chapter/modernization-healthcare-medical-diagnosis-system/72509?camid=4v1a

A Formal Knowledge Representation System (FKRS) for the Intelligent Knowledge Base of a Cognitive Learning Engine
www.igi-global.com/chapter/formal-knowledge-representation-system-fkrs/72769?camid=4v1a