IJARCCE

ISSN (Online) 2278-1021
ISSN (Print) 2319 5940

International Journal of Advanced Research in Computer and Communication Engineering
ISO 3297:2007 Certified
Vol. 5, Issue 9, September 2016

# Design of Software Safety Architecture and Software Safety Lifecycle of a Safety Critical System, Functionality and Model of Adaptive Cruise Control Safety Critical System

**P. Sowjanya**

M.Tech 2nd Year, Computer Science Engg, Vignan's Institute of Information Technology, Visakhapatnam, A.P, India

**Abstract:** There are many methods to ensure Safety in both Software Architecture and Software development Lifecycle. In this paper we have shown an overview of different types of models and standards used to develop Software Safety Architecture (SSA) and Software Safety Lifecycle (SSL) and mainly in Safety Critical Systems. In SSA we have discussed some types of strategies, approaches, design patterns and steps to analyse Safety Aspect in Software Architecture. In SSL we have shown three different types of implemented models such as: The model-driven software development (MDSD), CESAR domain (aerospace, automotive, rail and automation), and GTST-MLD based software development life cycle model. Safety Integrity and Fault Tolerance are the main important criteria of developing these SSA and SSL. A Safety Critical embedded System i.e. Adaptive Cruise Control System (ACCS) is taken for implementation. We have shown its basic Architecture and explained its Working technique using an Algorithm and based on that Algorithm its basic Functioning in Java and its MATLAB Simulink Model is shown in a brief way with its Screenshots.

**Keywords:** Adaptive Cruise Control System (ACCS), CESAR domain, GTST-MLD, MATLAB Simulink, model-driven software development (MDSD), Software Safety Architecture (SSA), Software Safety Lifecycle (SSL).

## I. INTRODUCTION

The role of Software in Safety Critical System is somewhat different compared to general type of Software's. Safety Critical System (SCS) is nothing but the system which has potentially destructive power. If such a system produced a failure at least once, the consequences that might be very serious, such as property loss, loss of human life and environmental damage etc. now a days, software application in SCS is more and more extensive, and the scale also increased drastically. From railway transit field to the aerospace field and from the power system to the medical system, this type of software plays a key role in command and control aspect for software safety. The core research in SCS safety is how to reduce the probability of unsafe system conditions that various SCS elements lead to, or weaken the SCS's consequences that failures produce, through using a variety of management, organization, technical measures [1].

Safety is thus achieved by deciding upon the appropriate design techniques to be employed in a specific system context. In general, current practice advocates two classes of design approaches:

- Process-based approaches. Industrial safety standards such as IEC 61508 prescribe a set of safety design techniques with respect to the classification of safety criticality. There is lack of practical guidance on demonstrating further how to exploit these techniques to tackle specific safety concerns. Moreover, most standards such as ARP 4761 and IEC 61508 dictate the allocation of safety functions over software and hardware but fail to explore the cost/benefit trade-offs behind allocation decisions.

- Architectural patterns. Architectural patterns have recently influenced the development of dependable systems. Yet the coarse-grained nature of design patterns makes it difficult to reason precisely about the achievement of desired safety properties and the design trade-offs involved [2].

As is commonly known, a software life-cycle is a sequence of steps describing how a development team specifies, designs, implements, tests, and maintains a piece of software. Each stage is described by its required inputs, performed activities and expected outputs, together with documentation, required properties, etc. In the case of critical embedded software, the cycle is usually a V-cycle, mainly decomposed into five (mandatory) phases: requirements specification, architecture design, implementation and low level testing, integration/ validation testing and the longest one, the maintenance phase [13].

"Adaptive cruise control is the first system in a network of sensors," said John Vaughan, vice president of business

IJARCCE
ISSN (Online) 2278-1021
ISSN (Print) 2319 5940
International Journal of Advanced Research in Computer and Communication Engineering
ISO 3297:2007 Certified
Vol. 5, Issue 9, September 2016

development at M/A-Com Inc. (Lowell, Mass.), which supplies sensors for the Mercedes system. "In time you will have a sensor field around the car which will be used by the vehicle's intelligence. It's the beginning of the microwave era in automotive electronics" [12].

## II. BACKGROUND

A critical system is a type of system such that 'failure' of that system could damage in human life, environment of the system or which can control entire equipment with its command in operation [3]. There are mainly three types of Critical Systems:

- Safety Critical Systems – Failure of this system may injure or kill people, damage the environment. Example: nuclear and chemical plants, aircraft – (Example: Weapon industry. People will be killed if the systems work.)
- Business Critical Systems – Failure of this system may cause great financial loss. Example: information system. Customer information cannot be lost, or hacked
- Mission critical system – Failure of this system may cause a mission to fail –Large values potentially wasted. Example: Space probe. Large sums of money, many years of waiting [4].

Safety is freedom from accidents or losses; software safety implies the contribution of software to safety in its system context. Another vital aspect of safety is risk. From an engineering standpoint, there is no such thing as absolute safety. Safety is often defined as the measure of the degree of the freedom of risk under all conditions [5].
Architectural design -the process of defining a collection of hardware and software components and their interfaces to establish the framework for the development of a computer system. [IEEE Std. 610.12-1990] ACCS is an embedded Safety critical system which has both hardware and software embedded in it. The Mercedes-Benz system uses a 77-GHz Doppler radar linked into the electronic control and braking systems to maintain a safe distance between a car with the system and the vehicle in front of it. Most of the new S-class vehicles are expected to ship with the radar, which carries a premium of about $1,500 [12].

## III.SAFETY IN SOFTWARE ARCHITECTURE

Fault tolerance strategies to balance the competing requirements for both reliability and safety, MIL-STD-1760 adopts two parallel fault tolerance strategies:

1. To assure reliable service by a redundant fault tolerant design, and to assure safe service by a 'fail silent' error recovery strategy.
2. It is important to note that requirements for fault tolerance may also introduce additional and complex asynchronous behavior which may exhibit even higher proportions of requirements related design faults than mission functions [6]

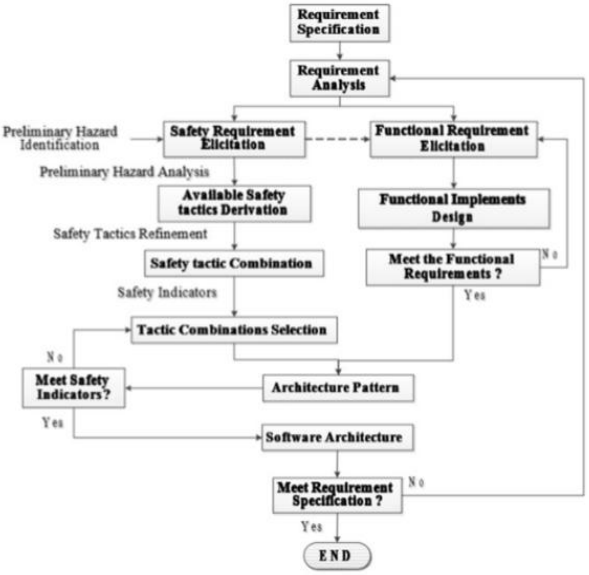

Fig.1. Safety-oriented SA design approach [7]

It is the key to design safe software that using the divide-and-conquer strategy when constructing safe architecture of software system. This approach is based on a hierarchical system model. The first step is to analyse the software requirement belong to one layer, and then according to the requirements specification, we should elicit functional requirements and safety requirements. However, not all the safety requirements can be individually extracted, some safety requirements are often reflected in the functional requirements. After obtaining the safety requirements through preliminary hazard identification, the second step is to conduct preliminary hazard analysis. At the same time, deriving and preliminarily selecting the feasible safety tactics. The third step is to further refine and classify these safety tactics. By analysing the severity caused by the system or architecture elements failure, safety metrics should be determined. The fourth step is through analysing the fundamental protection mechanism of safety tactics and the safety metrics, the feasible safety tactics will be filtered again, and the architecture should be constructed by choosing suitable safety tactics combination. At last, we should check whether the safety tactics combination can meet the safety requirements to control the software failure within an acceptable level of risk.

Safety architecture design patterns:

a) There are 9 different types of design patterns recommended by MIL-STD-1760 for different purposes : Inoperability design pattern, System level redundancy pattern, Homogenous redundant design pattern, Dissimilar redundancy design pattern, Monitor/actuator control channel pattern, Control & authority independence, Firewall (segregation) design pattern, Physical (spatial) proximity pattern, Signal complexity pattern [6]. The Architecture Analysis & Design Language (AADL) is a de-facto standard in the domain of avionics and automotive software systems.

IJARCCE
ISSN (Online) 2278-1021
ISSN (Print) 2319 5940
International Journal of Advanced Research in Computer and Communication Engineering
ISO 3297:2007 Certified
Vol. 5, Issue 9, September 2016

## IV. SAFETY IN SOFTWARE LIFECYCLE

For safety-critical systems it is often compulsory to perform various safety-related analyses as part of the software development lifecycle. The model-driven software development (MDSD) vision seems very promising in efficiently tackling the essential complexities (including safety concerns) of the software development process. Various standards, tools and techniques that are well-aligned with the MDSD vision are currently becoming widely accepted by the industry. RAM commander could be used to plug-in certain safety-related analyses, such as Failure Mode Effects Analysis (FMEA) and Fault Tree Analysis (FTA). The end-to-end development process leverages the V-model and the DSDM Atern agile framework.

The phases covered by the standard are as follows. Requirements Baseline corresponds to the complete specification provided by the end-user regarding the software product expectations. Technical Requirements correspond to all technical aspects that the software shall fulfil with respect to the end-user requirements. Software Architecture Design corresponds to the overall architecture that is created and refined based on the technical requirements. Software Component Design corresponds to a more detailed description of the elements described by the software architecture. Implementation corresponds to the development of the various software components described in the software component design phase. Verification corresponds to the testing of produced implementation in order to verify the correctness of the product performance. Validation corresponds to the testing of the software components as well as the complete software in order to validate the correctness of product performance [8].

The concept of the GTST-MLD-based software development life cycle model follows a hierarchical decomposition of software development life cycle activities. The main step to implement a GTST-MLD-based structural hierarchy is to decompose a function into sub-elements. The decomposition process is repeated until some lowest level of elementary activities is reached. Guidance on Software Review for Digital Computer-Based Instrumentation and Control System (BTP-14) is written by US Nuclear Regulatory Commission. BTP-14 was developed from IEEE Standards for different activities for the software life cycle and important design factors for safety critical software. Figure was derived from the information in NUREG/CR6101 and BTP-14. Planning a software development project can be a complex process involving a hierarchic set of many activities [9].

An analysis interaction between implementation phase elements is extremely important for safety, because, software requirement safety analysis is concerned with criticality analysis, system analysis, specification analysis and timing and sizing analysis. Also, all safety-related analyses should be performed in the design and the implementation phases. Safety-related functions of systems will be easily defined after the decomposition of the functional requirements.

The system design life-cycle in each CESAR domain (aerospace, automotive, rail and automation) is characterized by many commonalities, there are also inherent differences, prescribed by domain standards, which are usually reflected in the overall engineering activities.

The aim of the "CESAR-proposed" component-based development process is to provide a methodology that allows leveraging the productivity gains offered by uniting model-based development with component-based development.

This enhanced development process can be summarized as follows:

- Use of models as a basis for the development process,
- Definition of components as primary and mandatory artefacts throughout the development life-cycle,
- Traceability between development steps, requirements and various types of artefacts in general,
- Possibilities for early verification and validation (at design stage),
- An enhanced safety process based on models that are fully linked and/or synchronized with system design models.
- Adoption of product line principles during component design that can promote re-usability

CESAR design methodologies and suggested modelling approach also strongly promotes component-based engineering (CBE) principles during design and construction of safety-critical systems [10].

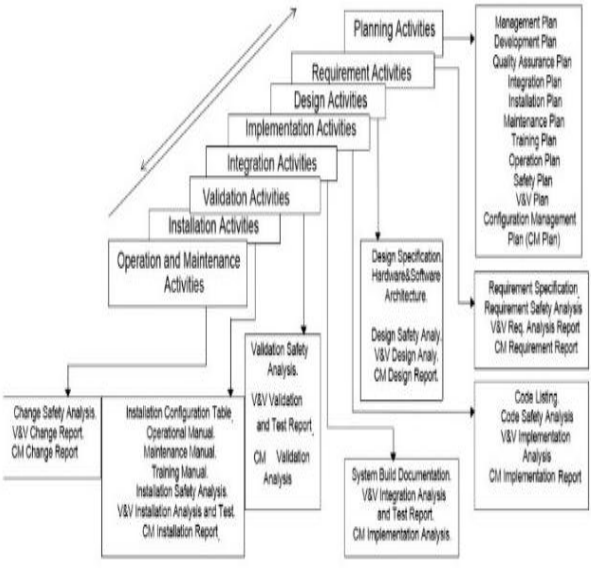


Fig. 2. Hierarchical Lifecycle of Safety Critical System.

IJARCCE
ISSN (Online) 2278-1021
ISSN (Print) 2319 5940
International Journal of Advanced Research in Computer and Communication Engineering
ISO 3297:2007 Certified
Vol. 5, Issue 9, September 2016

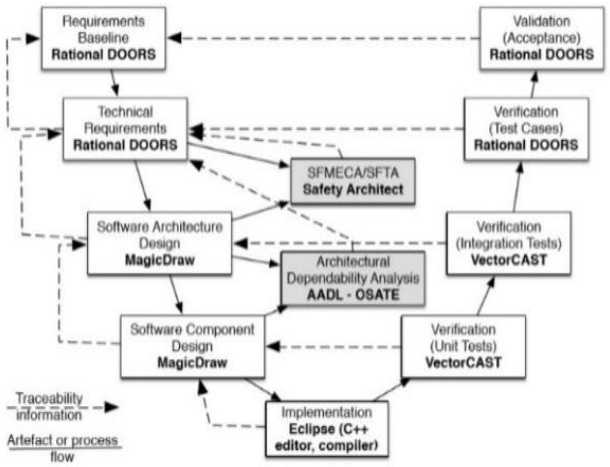


Fig. 3. Atern V-lifecycle for Safety critical systems

## V. A SAFETY CRITICAL EMBEDDED SYSTEM

**ADAPTIVE CRUISE CONTROL SYSTEM (ACCS):**

Adaptive cruise control is also called active cruise control, autonomous cruise control, intelligent cruise control, radar cruise control, or traffic-aware cruise control. Adaptive Cruise Control (ACCS) is an automotive feature that allows a vehicle's cruise control system to adapt and adjust the vehicle's speed to the traffic environment. A radar system attached to the front of the vehicle down the bumper is used to detect whether slower moving vehicles are in the ACCS vehicle's path.

If a slower moving vehicle is detected, the ACCS system will slow the vehicle down and reduces the time gap, between the ACCS vehicle and the opposite vehicle. If the system detects that the forward vehicle is no longer in the ACCS vehicle's path, the ACCS system will accelerate the vehicle back to its set cruise control speed. This operation allows the ACCS vehicle to autonomously slow down and speed up with traffic without intervention from the driver. The method by which the ACCS vehicle's speed is controlled is via engine throttle control and limited brake operation [11].

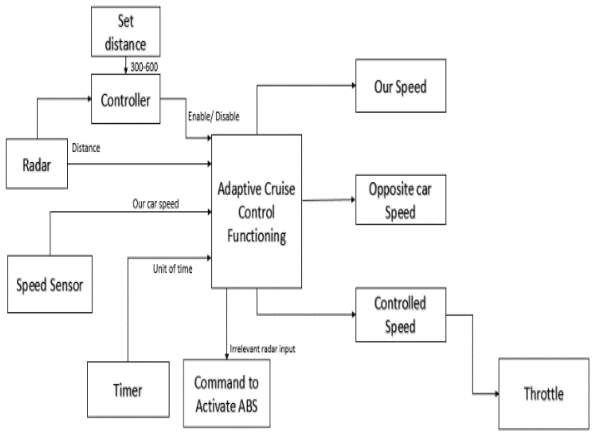


Fig. 4. Architecture of ACCS

A.SAFETY ALGORITHM

1. Start the vehicle and Car speed (S1) of our vehicle using a speed sensor.
2. Take two distances (D1 and D2) with difference of some period of time (eg.1 millisecond) using a RADAR.
3. Calculate the opp. car speed by using the speed of our vehicle and RADAR inputs.
4. Difference between the distances (DD = D2-D1), i.e. distance of the vehicle for 1st unit of time and 2nd unit of time.
5. Now opp. car speed (S2 = S1 + DD), i.e. Sum of Our Car Speed and Total Difference DD.
6. This is taken as the Controlled Speed of our vehicle CS.
7. Now we can Increase or Decrease the Throttle value and the Controlled Speed (CS) depends on the opp. car speed (S2).
8. Irrelevant data such as suddenly to null or huge difference of RADAR input, Switching on to ABS is done.
9. ACCS enabled, Repeat from Step1.

B. FUNCTIONING OF ACCS IN JAVA:

In java we have improved few user interface screens of Adaptive Cruise Control System (ACCS) such as Welcome Page, Login Page, Parameters, and Set Speed etc. We have done a Basic functioning ACCS in Java with few parameters such as speed of the vehicles, Radar input, Set speed of the vehicle distance, Distance between two cars and so on.

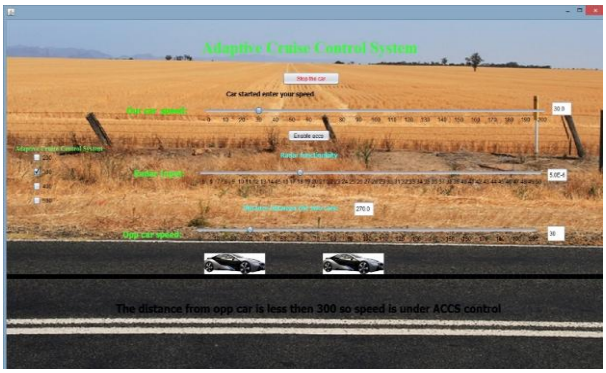


Fig. 5. Functionality of Adaptive Cruise Control System (ACCS)

By looking at this Interface we have to click start the car button and our car speed can be manually given or get adjusted through the Sliding bar. Later the input for our ACCS is taken by Radar which is used to calculate the distance between two cars and by using algorithm in functionality of ACCS we can get at what speed is the opposite vehicle travelling and at what speed our vehicle must get controlled in order to avoid collision or accident between the two vehicles. In the above interface the Opp. car speed need not be given manually, it is automatically calculated by the input of radar. If the radar input is below 300 units of distance then ACCS automatically gets

IJARCCE
ISSN (Online) 2278-1021
ISSN (Print) 2319 5940
International Journal of Advanced Research in Computer and Communication Engineering
ISO 3297:2007 Certified
Vol. 5, Issue 9, September 2016

activated. This set speed can be given and decided by us, it may be in between 300-600 units of distance. If the vehicle is moving below that distance, our ACCS will automatically get activated and slows down our vehicle to a certain speed as of the opposite vehicle's speed. This speed is given as Controlled speed in result. If the distance between two vehicles gets very closer or if the opposite vehicle is stopped then the ACCS equipped vehicle also must stop or should be in an idle state.

C. RESULTS

The result graph of this ACCS was done use java free charts and this shows the functioning of an entire ACCS. This is done using thread concept of java and the graphical representation shows in the graph.

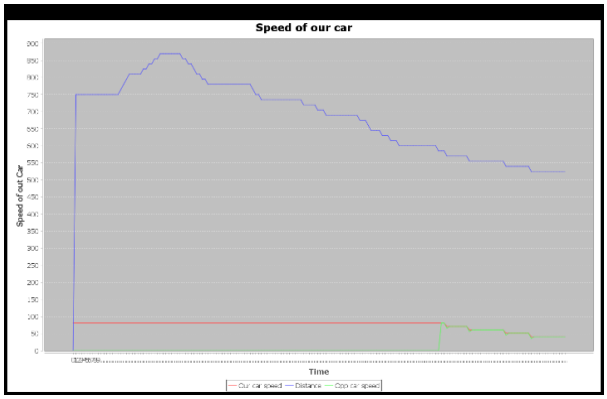


Fig. 6. Graph showing speed of the cars and distance between them.

In this graph we can show the speed of the vehicle (gradually increased or decreased) with equal to the opposite car speed. Accordingly our ACCS gives a result of controlled speed which must be done using a speed sensor. In the result we can clearly see that whenever the opposite vehicle is in the set speed area of ACCS within the range of the RADAR then our vehicle enables ACCS and moves in a controlled speed. If opposite vehicle is not in range of the set speed our vehicle sets to normal speed and disables ACCS.

## VI. CONCLUSION

Adaptive Cruise Control system was developed for the purposes of driving with both safety and comfort. It reduces the number of brake and switch operations that are required of the driver. As a result, the system reduces the driving burden so that the driver can drive in comfort. In this paper we have shown a basic functionality of an Adaptive Cruise Control system done in MATLAB Simulink Software. When the input value are given to the Radar and our speed, the calculation of front v speed and controlled speed came as a result. The effects and causes of these ACCS parts were identified by using Failure Mode Effective Analysis (FMEA) and root causes of these failures were analysed by using Fault Tree Analysis (FTA). The combined results of FMEA and FTA provide input for analysis of temporal or causal justification for prioritization of verification or validation test systematic approach from system down to subsystem.

## REFERENCES

[1] Springer, 1989, vol. 61.International Conference on Information Science and Computer Applications (ISCA 2013）Safety-Oriented Software Architecture Design Approach by Yuling Huang,
[2] M. Wegmuller, J. P. von der Weid, P. Oberson, and N. Gisin, in Managing Architectural Design Decisions for Safety Critical Software Systems Weihang Wu, Tim Kelly
[3] R. E. Sorace, V. S. Reinhardt, and S. A. Vaughn, "High-speed digital-to-RF converter," U.S. Patent 5 668 842, Sept. 16, 1997.https://courses.cs.washington.edu/courses/cse466/05sp/pdfs/.../L12-Critical_Systems.pdf
[4] Safety-Critical Systems by Rikard Land
[5] Leveson, N.G.: Safeware: System Safety and Computers. Addison-Wesley (1995)
[6] Safety, Software Architecture and MIL-STD-1760 by Matthew John Squair Senior Safety Consultant Jacobs Australia GPO Box 1976, Canberra, ACT 2601
[7] Safety-Oriented SoftwareArchitecture Design Approach Yuling Huang in International Conference on Information Science and Computer Applications (ISCA 2013）
[8] Model-Driven Software Development of Safety-Critical Avionics Systems: an Experience ReportAram Hovsepyan, Dimitri Van Landuyt, Steven Op de beeck, Sam Michiels, Wouter Joosen, Gustavo Rangel, Javier Fernandez Briones, Jan Depauw, iMinds-DistriNet, KULeuven Space Applications Services N.V./S.A.
[9] SOFTWARE DEVELOPMENT LIFE CYCLE MODEL TO ENSURE SOFTWARE QUALITY by Nihal Kececi, and Mohammad Modarres
[10] http://www.springer.com/978-3-7091-1386-8, The System Design Life Cycle by NikolaosPriggouris, Adeline Silva, Markus Shawky, Magnus Persson, Vincent Ibanez, Joseph Machrouh, Nicola Meledo, Philippe Baufreton, and Jason Mansell Rementeria
[11] Adaptive Cruise Control System Overview by 5th Meeting of the U.S. Software System Safety Working Group April 12th-14th 2005 @ Anaheim, California USA
[12] Adaptive cruise control takes to the highway BY Peter Clarke on 10/20/1998 06:06 PM EDT
[13] Development Life-cycle of Critical Software Under FoCaL1Philippe Ayrault Etersafe 43, Semantics, Proofs and Implementation Laboratoire Informatique de Paris 6 Pierre & Marie Curie University 104, Avenue du President Kennedy F-75016 Paris.

## BIOGRAPHY

**P. Sowjanya** is currently doing M.Tech 2year CSE in Vignan'sinstitute of information technology, Visakhapatnam. She completed her B. Tech CSE in Sri Chaitanya Engineering College, Visakhapatnam. She had published 2 papers for Internet of Things and Cloud Computing domains. Her areas of research includes Safety Critical Systems, Internet of Things, Cloud Computing and Big Data.