Asian Journal of Information Technology 15 (11): 1799-1815, 2016
ISSN: 1682-3915
© Medwell Journals, 2016

Safety Assessment of Distributed Automotive Software System Model with Design for Traceability

[1]S. Ravikumar, [2]S. Chandrasekaran and [3]S. Ramesh
[1]Department of Information Technology, Valliammai Engineering College, 603203 Chennai, Tamil Nadu, India
[2]Department of Computer Science and Engineering, Sri Ranganathar Institute of Engineering and Technology, 641110 Coimbatore, Tamil Nadu, India
[3]Department of Electronics and Communication Engineering, Valliammai Engineering College, 603203 Chennai, Tamil Nadu, India

Abstract: The objective of the research work is to propose a safety assessment model for an automotive software design as an interacting distributed software component towards traceability. The existing challenges in the automotive software are the composability of the distributed design components and the traceability of the fully assembled or partially assembled design software components. The safety of the composable design does not only depend on the individual software component behavior but also depends on the traceability of each of them in every possible level in both the directions. In such distributed design environment, the software design components can be classified into three types as influential, in-connection and in-absentia types. The proposed research explores a formal approach for the Safety Assessment Logic (SAL) with prevailing the Safety Integrity Levels (SIL). The set of distributed parameters which may cause the risk are studied using a Safety Assessment, Design document (SAD). The distributed SADs are reported to a central server as Document as a Service (DaaS) for carrying out the safe transactions and alert messages across the system. Safety components from each category are considered like air-bag release time, anti-lock braking system and parking distance parameters are tested under various environment conditions.

Key words: Distributed software, traceability, assessment logic, safety parameters, components safety, document services

INTRODUCTION

The software safety is not only a special case of safeguarding the application or system software from design flaws but also avoiding from the illegal utilization of the components. The software should be smart enough to predict certain risks and avoid or mitigate them in the initialization process. Many earlier software safety models had proposed a design for safety model in automotive software architecture focusing the context awareness features, user actions and unexpected reaction from the environment (Chandrasekaran *et al.*, 2011). The safety aspect in the design and development of automotive software is considered in the system level and also in the detailed software component level. Safety and reliability are often equated and the reliability is usually defined as the probability of failure free operation that a system will perform its intended function for a specified period of time under a set of specified computational environment conditions, whereas safety is the probability of those conditions that will not lead to a mishap or hazards whether the intended functions are performed or not (Nancy, 1995) .Safety can be defined as the freedom from exposure to danger and therefore it is a subjective measure which makes safety provision and measurement extremely difficult and contentious tasks. The Preliminary hazad analysis helps to identify the potential hazards to the automotive system which can later be eliminated. By analyzing all the hazards and classifying them, the system hazards can be translated into high level system safety design and constraints (Jesty *et al.*, 2007). Software safety analysis examines the consequences of faults or failures through the respective states on items considering their functions, behaviour and design (Wabmuth *et al.*, 2011). Their contribution towards the identificatio n of new functional or non-functional hazards was previously not considered during the hazard analysis and risk a ssessment (Schlummer *et al.*, 2010). Automotive safety integrity level is used as a specification for the risk management team and the requirements for risk reduction

Corresponding Author: S.Ravikumar, Department of Information Technology, Valliammai Engineering College, 603203 Chennai, Tamil Nadu, India

are assigned to each considered hazardous situation (Birch *et al*., 2013). The safety defects existed in General Motors X-body passenger cars such that at times when the brakes were applied, the rear brakes would lock up, sending the vehicles out of control (Pecht *et al*., 2005). The software safety risk is emanating from the family of Intellectual Property (IP) issues that seem to exceed that of hardware developments. The IP for a large piece of software is remarkable (Burton *et al*., 2012). The next generation of premium cars will exhibit hundreds of millions of lines of code. One of the central challenges for next generation automotive system development is due to heterogeneous logical and technical architectures (Chandrasekaran *et al*., 2010). The number of ECUs required to deliver the desired functionality and the dynamic reallocation of computing and communication resources makes the software design more and more complex and unsafe. The software testing cannot assure that the absence of safety critical errors. A key property of these models is that they abstract communication behaviour of hardware, middleware and application software in a stochastic way, focusing on size, frequency on size, frequency of occurrence and timing of the data to be sent (Pretschner *et al*., 2007). The safety assessment model for software services using the advanced collaborating technologies where the computational for the software safety is assessed not only on the functional and behavioural design but also on the non-compliances of the automotive software service standards (Wang and Zhang, 2013). It has described an approach to safety-driven design using a new hazard analysis method called STPA (Systems Theoretic Process Analysis) which is based on systems theory and on STAMP, a more comprehensive model of accident causality than the standard linear event-chain models. STPA considers the non-linear interrelationships among events and system components rather than just linear cause-effect chains, it looks at the processes behind the events and it includes the entire socio-technical system rather than just the hardware or physical process (Stringfellow *et al*., 2010). The probabilistic risk quantifications via Failure Rate are new for traditionally oriented safety experts. Considering the big mass of offered workshops to learn more regarding the Standard at the market place one can draw the conclusions backwards that the entire automotive industry worldwide is a bit puzzled to understand and to adopt the Standard adequately to their products (Gandhi and Trivedi, 2007). Two models were finally identified as fulfilling all three criteria. The first was the COCOM/ECOM (Contextual control model/extended control model) of Hollnagel and the other was the IVIS DEMAND software (in-vehicle information system design evaluation and model of attentional demand) which aims to help system designers to estimate the demand placed by their systems on drivers (Leveson, 2011). There are many Standards in Automotive Software Safety and Automotive Software reliability which deals with the dangerous failure that may arise from incorrect specification of the system either through hardware or software, random failure of hardware, systematic failure of hardware and software, common cause failure, human error and environmental influences (Cacciabue and Carsten, 2010). The IEC applies to any software forming part of a safety related system or used to develop a safety-related system within the scope of IEC 61508-1 and IEC 61508-2 (Haider and Nadeem, 2013). A programming language coding standard shall specify good programming practice, proscribes unsafe language features, promotes code understandability, facilitate verification and testing and specify procedures for source code documentation (Nancy, 1995). The DO-178B is the software development standards should enable some components of a given software product or related set of software products to be uniformly designed and implemented (Rierson, 2013). The ISO 26262 is functional safety standards which are a risk based safety standard that uses Automotive Safety Integrity Levels (ASILS) for assessment, avoiding and controlling systematic failure or controlling random hardware failures to fulfil the standard requirement (Bashir *et al*., 2013). The code itself is required to be developed in accordance with coding guidelines with MISRA C. Safety is one of the key issues of modern automobile development. New functionality is not only in the area of driver assistance, vehicle dynamics control and active and passive safety systems increasingly touches the domain of safety engineering. The development and integration of these functionalities will further strengthen the need to have safe system development processes and to provide evidence that all reasonable safety objectives are satisfied. A fundamental role of coding standards is to define a safer sub-set of the programming language by framing a set of rules which eliminate coding constructs said to be hazardous (Kafka, 2012). The organization of the paper is as follows:

MATERIALS AND METHODS

Proposed automotive insafe system model: The automotive software is designed in a highly subjective and optimistic manner in order to satisfy the technical and physical requirements of various hardware and software components in a distributed manner. The safety of multiple embedded components lies not only in the correct functionality as required but also the correctness from a live component's behaviour. The software requirements are isolated from the system and hardware requirements before designing the individual modules keeping the Design for Safety approach (DfS). The conceptual automotive software system to provide safety through the design phases of individual components are

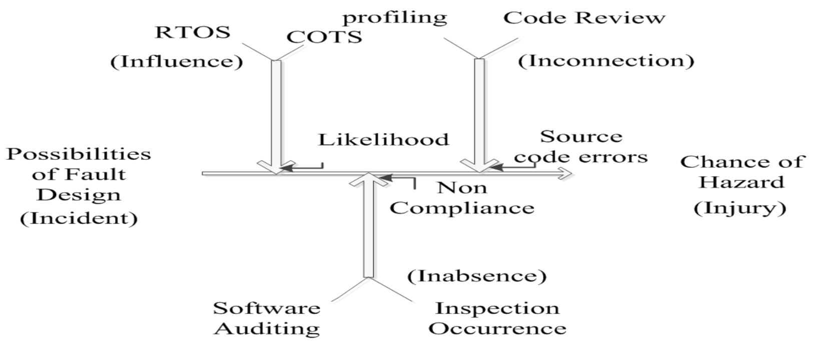


Fig. 1: IN-SAFE distributed software system safety model

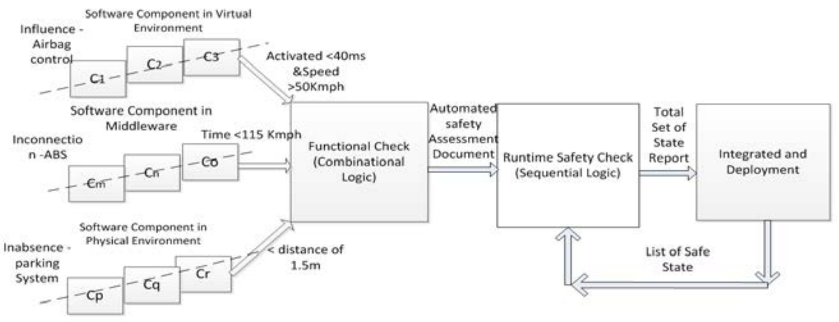

Software Component in Virtual Environment
Influence - Airbag control
C1
C2
C3
Activated <40ms &Speed >50Kmph
Software Component in Middleware
Inconnection -ABS
Time <115 Kmph
Cm
Cn
Co
Software Component in Physical Environment
Inabsence - parking System
Cp
Cq
Cr
< distance of 1.5m
Functional Check (Combinational Logic)
Automated safety Assessment Document
Runtime Safety Check (Sequential Logic)
Total Set of State Report
Integrated and Deployment
List of Safe State

Fig. 2: Safety enhancement model for dynamic distributed architecture

explored in its model, components, behaviour and algorithm to provide safety in exceptional circumstances and invisible triggers from the environment conditions.

The automotive software safety assumes that any incident within the system or outside the system can be turned into an injury or multiple injuries based on three different types of association of participating system components. These system components, including both hardware and software or even firmware even though from different vendors must work in an integrated manner to provide safety by the components. The heterogeneous software components can play their role and share their responsibilities as and when required when the system is put into operational mode. The system safety can be thought of an in presence or in the absence or in connection nature of many software design components to minimize the transformation of an incident to an accident. Sometimes, it is better and safer to make a set of software components absent during a specific scenario and similarly in some other cases, it is better to have different sets of components to be present and in connection with other counterparts. Hence, in the proposed model, the software design components are classified and grouped as Influential group, In-connection group and In-absence group across all software modules for the proper functioning of the system as shown in Fig. 1. The above mentioned software design components may be distributed in nature that is some of the software functions and behaviour which may be executed outside the system and some of them may be operated by the system by the operators. For example, the engine design software component of a car may be operated with a spark-plug control software component obtained from a third party and running on the brake control component from yet another designer. In all cases, the software design components can get identified based on the scenario and the mode of operation of the final system. The safety is influenced by an influence air bag control subsystem software component whose design is that it will be activated within a time period < 40 m sec and when the system vehicle speed is >50 km/h. In an another scenario, an in-connection antilock braking system should be activated in a time period that is <115km/h and at a different scenario, the safety will be realized once the in-absence parking aid software subsystems have to enabled less than a distance of 1.5 m. The proposed in-safe computational model can be extended for distributed safer software design components. The various inputs of software components from different environments like virtual, middleware and physical environment is checked and validated through the run time safety check before the software component from various environments are integrated and deployed which is described as a Safety enhancement model for dynamic distributed architecture in Fig 2. The software component

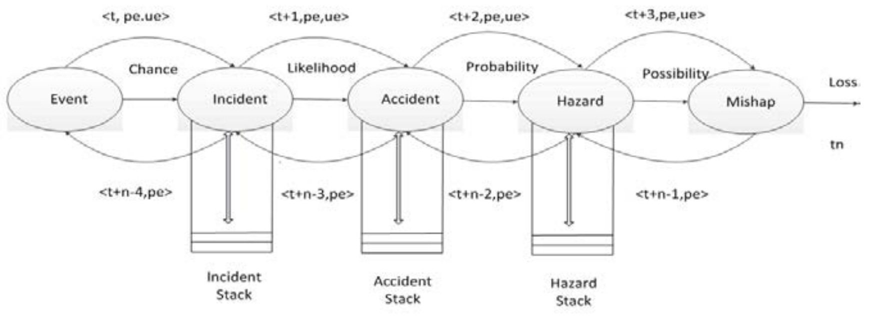


Fig. 3: Safety assessment behaviour design

Table 1: Safety assessment sample

No. of uncontrollable events(external)	No. of unsafe operations in a sample session (internal)	Session time period	Size trace stack
10	1	5	50
12	4	10	30
11	6	15	27.5
13	3	20	86.6
11	2	25	137.5
15	7	30	64.3
14	8	35	61.25
17	10	40	68
16	2	45	360
18	5	50	180

or control modules residing within various memory chips, including random and programmable read only memory devices are checked for primitive operations like read and write, reset are checked as per the hardware architecture. The signals cross over that may happen across various devices are checked through a functional checker as per their priorities. Similarly, all the components in the middleware and those created as virtual applications during on demand conditions are also checked for their functions. The static checker is deployed in each of the gateway points and then applied to a centralized control where the run time data, processes are checked for the transient behaviour of individual subsystems. These functionality checking towards safety assessment can be automated by an event and time triggered strategy where each and every event in the input and process, including memory are monitored and the chance of transferring the state of the system from incident to an accident and the accident to hazard and so on. The nondeterministic distributed automation can be considered as a parallel Turing machine with more number of input tapes and unlimited stacks. It is a timed automaton where the absence of an input value for a fixed duration may cause the system to change its current state and enter into a much safer state.

The chance of occurrence of planned event(pe) or unplanned event(ue) may be an incident which is acceptable or unacceptable that results a likelihood of accident. The safety controller can have any number of temporary storage based on the number of functional checks are considered. The safety design can be explored with the help of push down automata where the storage size or the stack memory size plays an important role. Each and every time the safety system checks its input, current state, permitted transition within the specified time interval once the conditions are satisfied. The probability of occurrence of incident to accident which in turn leads to hazard or mishap that causes injury or loss of life which is described as safety assessment behaviour design in Fig. 3. The stacks are used to trace the occurrence of unsafe events or transitions with their parameters like timing, identity or code of the task and the number of times the task was repeated in that sample session. There are hazard stack, incident stack and accident stacks for storing the hazard index value, number of occurrences and the time of occurrence of the incident and the Table 1 describes Safety assessment Sample for Size Trace Stack:

$$\text{Size Trace Stack} = \text{Max}\,[\text{Number of uncontrollable events (external)} \mid \text{Number of unsafe operations in a sample session (internal)}] \times \text{Session time period}$$

Safety assessment logic and rules in safe automotive software algorithm design: The automotive system safety is a unified way of identifying all possible opportunities from different risk items that lead to some form of risks and mitigating those through proper actions within the software components. The resultant of safety mechanisms is to minimize the risk or the risk free operation is to be a safe operation. But the risk is a function of frequency (or likelihood) of the hazardous event and the related degree of injury (severity). A safe operation will be a reliable one but not all functionally reliable operations are safe. The safety is a special kind of functionally reliable state. Especially in the case of distributed software components in the automotive software application, the reliability confidence interval of each component should be within the safety level as per the standards. Since, the safety is focused, the quality factor can be refined further as the system is ither reliably safe or safely reliable. The

Table 2: D-SAR based logic primitives

Logic	Codes
Each and Every	EAE
Sometimes	SOT
Always	ALW
Never	NEV
Whenever	WHN
Wherever	WHR
Whatever	WHT
Anyway	AWY
By the way	BWY
Totality	TOT
Individuality	IND
Not at all times	NAT

automotive applications need many such combinations or Functions of Functions (FoF) where the system safety has to be checked towards the component safety faults. The semantics of assessment rules that make the system to be understood and categorized as safety activities or reliability activities are discussed below and Table 2 describes the distributed safety assessment rules based logic that is D-SAR based logic. The set of proposed safety assessment rules as safety policies for software systems can be listed as follows.

Whenever there is system safety, there should not be any hazard due to system software, application and hardware whether physical or virtual. H(t) = ∑ Prob(Hazard). Software(t) U ∑ Prob(Hazard). Hardware(t) for all the states of hardware and software in operation at that time. Whn. Safe(system) ⊨ 1- Prob(H) if there is no transfer of data at that time across hardware and software .Whn. System(safety)↣hazard hardware U hazard. Software = ø. In an automotive system, the software that is embedded in all hardware in all forms and the services like transport signaling services should not be in safety failure like, in-absence signaling or hardware crash or air-conditioner control software failure.

Follow safer instructions when compared with least unsafe instruction always. In representation, Totality => TOT => totally is a spatial parameter, Always => ALW => always is a time parameter. Hence, totally. always [prob (safe. instructions)] >> prob (unsafe. instruction).Where safer instruction implies there is at least one unsafe instruction in the total system in the context of functional logic. In the context of functional safety of all the software modules , the automotive system can be operated in Different mechanical gear positions indicating that on all occasions, the driving the system at the top speed is not safer than driving the same with normal speed on typical road conditions.

In each and every case there is a minimum and maximum level of safety for all software components. Case is a set of system or modular states and the level being the safety level without causing any injury to the system or subsystem where the Each and Every case is represented as EAE. ∀ Component c , software minimum safety ≥ maximum injury and maximum safety functions., i.e., max [prob (injury)or prob (function].)

As an illustrative scenario, the various safety levels are different for different component based on the design and deployment. If the tyre pressure gets overvalued, then the software module monitors the pressure on the wheels with minimum and maximum values to avoid injury in rough road conditions.

Whenever (WHN) it is needed, the periodic checks should always be there for each software module. Whn.need$_m$(t) ≥check$_{m-1}$(t). check$_m$ (t). check$_{m+1}$ (t)
Where m indicates meet the software module in the software system architecture. The safety monitoring functions called Function of Functions are to be checked by the manager component like thermal, pressure, data management function are to be performed at the previous, current and future modules at any point of time. Fuel check, Brake check, Lighting check, Battery check is to be performed at any point of time when the safety manager module demands them. None of the system can go wrong and unsafe states.

By the way (BWY), the Influence and In-connection features are not supplementing the Incident. Bwy.prob(incident) ≥ ~ [prob (influence ^ In-connection) V prob (influence)v prob (In-connection).

Some of the software components are influencing or in-connecting the normal components through improper interfaces creates unwanted actions within the automotive software system. The short circuit of a wire from the battery may touch the body of the vehicle and influence a shock are damaging the other main functions of the control system modules. Anyway (AWY), In-absence of components can never affect safety. Awy prob (system.safety) ↣ nev [prob (inabsense. Component$_i$)].

In some cases, the absence of infotainment component and its interfaces may not be functioning due to the safe mode driving that can never affect the safety of the travel. At some of the times (SOT), the incidents may turn to be accidents but always with minimum injury. Sot, incidents → alw [accidents (minimum. injury)]

For example, at some of the time of travel, tiger Crossing may demand, the engine to be made ON Continuously but lights to be OFF, without any movement of the vehicle that leads the same module executed many times without any further change in data cause the temperature of the automotive System high and also leads to minor headache and delay. Always(ALW), the In-connection and In-absence components never lead to injury. Alw,Component (In-connection ^ In-absence) ↣ [Nev.prob(injury)]

The absence on essential data or alert status and the over speed indication component can cause the system to halt or stop without leading to system failure and physical injury to the traveling people. Always (ALW) The In-absence event lead to hazard. Alw, component(t).In-absence →prob(hazard)
The absence of the viper control section during raining season always leads to hazard of accidents on a highway.

Not at all times (NAT) the injury is the totality effect of an incidents and not all incidence cause injury at all times individually. Nat, the prob(injury) ↣ Tot. incidents(injury) ⊕ Ind.incidents(injury). The automotive software system causes major injury which is a an accumulations or aggregations of various causes through the respective events and not at all times these accumulations or individual safety failure leads to damage of the users. Many incidents leave a trace of full safety even though hazards in the operation phase.

Safety document and data design: The safety logic has been applied through the propositions wherever they are applicable and a consolidated document has to be generated on the fly across the distributed network. For example the safety logic, “Whenever (WHN) it is needed, the periodic checks should always be there for each software module can be stored in the individual node as a safety document and the data or association, namely, “Anyway (AWY)and In-absence” are stored as data where the In-absence of components can never affect safety. The other type of data is SAD no, date, safety design constraints, compliance manager, project manager, safety manager and network manager, safety goals are discussed here.

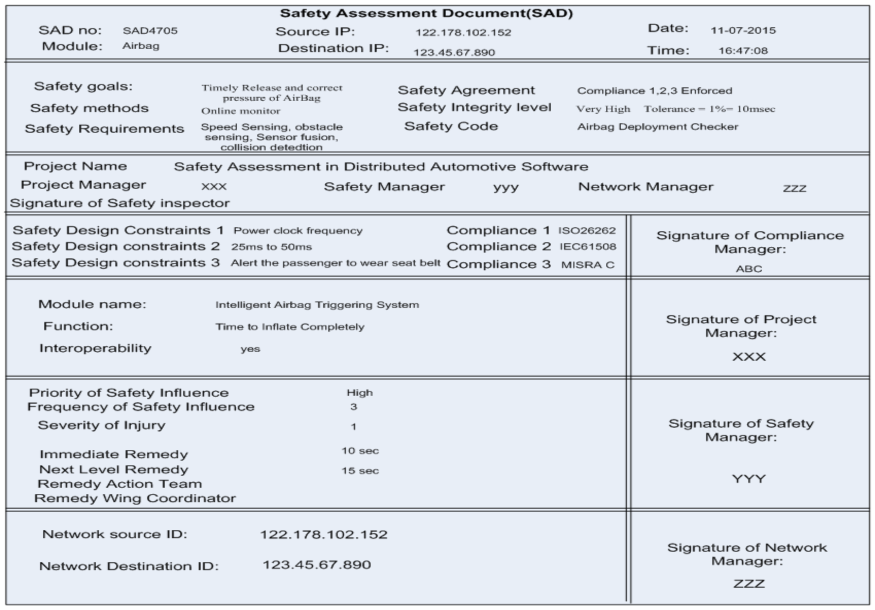
The safety document is to be portable one and at the same time the information and the data have to extracted by the computing node for further processing. The document model and the contents in a standard template have to be designed such that the same document can be reused for successive nodes with suitable modifications. The Safety Assessment Document (SAD) should carry the important field of safety goal at that instant of processing and the method with which the goal achieved. The needed fields are included as mentioned like Safety method, safety requirement, safety agreement, safety constraints, safety integrity level and safety code as shown in Table 3. As per the Safety assessment

Table 3: Qualitative safety assessment for automotive software

Embedded software Component	Safety goals	Safety requirement	Safety Methods	Safety Constraint
Air bag control	Timed release and correct pressure	Speed Sensing,Collision	Online monitor detection release	Power clock frequency and 25ms -50ms
Seat belt monitor	Seat belt maintains the passengers in their seats at the time of the collision	To prevent the passengers from impact at the time of the collision	Warning lights or a warning sound	Alert the passengers before the collision and activated
Tyre pressure checking	Providing appropriate traction to the road surface and it does not wear quickly.	Providing motion between the road and the vehicle.	Providing high friction in wet condition	Regular tyre pressure check up and loss of 3-6%pressure per month
Anti-Lock Braking	Maintain friction with the road surface	Prevent uncontrolled skidding and wheel lock	Electronic control to the front and rear brake bias	Faster rate with better control with 30 Sec Braking and collision control
Adaptive cruise control	Automatic adjustment of speed from ahead vehicle	Vehicle Distance information is sensed through sensors	Paired with a pre-crash system to begin braking	
Brake Assistance System	To increase braking pressure in critical situation	Immediate stop of the vehicle	Fast reaction and maximum braking input	Maximum braking effort within a millisecond and reduce stopping distance by 45%
Traction Control	Balance the loss of road grip.	To maintain fluid flow and engine torque .	Stabilize the power delivered to the wheels	Controlling the yawing movement of the wheels and Honda Tractio n control allowed me to control 500 HP through 4" tires, improving my 1/4 mile by 1.3 sec and speed at distance by 3-6 mph “.
Forward collision Warning	Alerts the driver to prevent accident	Reduce the severity of accident	Radar, laser and camera sensor	Up to 3 sec
Infra-red night vision activation system	Increases vision perception beyond head lamp range	Gives enough time to react	Reduces risk of accidents during night time.	Senses objects farther than 40m
Parking aid sub System	Making parking convenient and easier	Ultrasonic sensor mounted on the bumper	Monitored by visual or by sound effect	Makes parking safer without any damage and 45 degree turning-front wheel
Cornering Brake Control	Distributes brake pressure to all wheels	To improve wheel stability	To reduce the speed of the vehicles	Helps to maintain stability control on road at critical times and if a vehicle is cornering (with a transverse acceleration of over 0.6g-pak ground acceleration
Roll over detection subsystem	Rollover protection	Building electronic stability control	Control vertical movement of wheel	Activating active suspension system

Table 3: Continue

Embedded software component	Safety goals	Safety requirement	Safety Methods	Safety Constraint
Electronic stabilization on program	Improving vehicle stability	Reducing loss of traction	Automatically applies braking during loss of steering control	Minimize loss of control of the vehicle.
Brake by Wire program	Shorter stopping distance	It eliminates mechanical linkage	Controlling brake through the electronic control system	Improved crash worthiness
Lane Departure Warning module	To minimize accidents	Gives warning when the vehicle moves out of the lane	Camera based approach to monitoring and gives warning to the driver.	It automatically takes steps to ensure vehicle safety and Lane Departure Warning uses a second signal processor and software to filter lines and longitudinal patterns at speeds above 50 km/h


Safety Assessment Document(SAD)

SAD no:	SAD4705	Source IP:	122.178.102.152	Date:	11-07-2015
Module:	Airbag	Destination IP:	123.45.67.890	Time:	16:47:08

Safety goals:	Timely Release and correct pressure of AirBag	Safety Agreement	Compliance 1,2,3 Enforced
Safety methods	Online monitor	Safety Integrity level	Very High Tolerance = 1%= 10msec
Safety Requirements	Speed Sensing, obstacle sensing, Sensor fusion, collision detedtion	Safety Code	Airbag Deployment Checker

Project Name	Safety Assessment in Distributed Automotive Software				
Project Manager	xxx	Safety Manager	yyy	Network Manager	zzz
Signature of Safety inspector					

Safety Design Constraints 1	Power clock frequency	Compliance 1	ISO26262	Signature of Compliance Manager: ABC
Safety Design constraints 2	25ms to 50ms	Compliance 2	IEC61508	
Safety Design constraints 3	Alert the passenger to wear seat belt	Compliance 3	MISRA C	

Module name:	Intelligent Airbag Triggering System	Signature of Project Manager: XXX
Function:	Time to Inflate Completely	
Interoperability	yes	

Priority of Safety Influence	High	Signature of Safety Manager: YYY
Frequency of Safety Influence	3	
Severity of Injury	1	
Immediate Remedy	10 sec	
Next Level Remedy	15 sec	
Remedy Action Team		
Remedy Wing Coordinator		

Network source ID:	122.178.102.152	Signature of Network Manager: ZZZ
Network Destination ID:	123.45.67.890	

Fig. 4: Safety assessment document

Table 4: Safety function0: sfunc0 air bag release at speed 55km/h

S0	S1	S2	S3
Very low	Very low	Low	Low
Low	Low	Medium	High
Low	Medium	High	Very high
High	High	Very high	Very high

Table 5: Safety function1: sfunc1 air bag release at speed 80 km/h

S0	S1	S2	S3
Low	Medium	High	High
Medium	High	Very high	Very high
High	Very high	Very high	Very high
High	Very high	Very high	High very

Table 6: Safety Function0 :sfunc0 ABS system controlled at speed 95km/h

S0	S1	S2	S3
Very low	Very low	low	Medium
Low	Low	Medium	High
Medium	Medium	High	Very high
High	Very high	Very high	Very high

Table 7: Safety function1 :sfunc1 ABS system controlled at speed 115km/h

S0	S1	S2	S3
Medium	Medium	High	Very high
Medium	High	Very high	Very high
High	High	Very high	Very high
Very High	Very high	Very high	Very high

document shown in Fig. 4 the parsing and data retrieval from the respective nodes, It can be determined that the safety function , the Frequency(F) of the accident and its corresponding Severity(S) as shown in Table 4-7. The conditions for the satisfaction of the safety functions are that the severity Si and the frequency Fj of the needed events are uncertain and empirically calculated as per the trial and test drives. Figure 5 shows the distributed safety assessment process is integrated from different nodes for a safety check and given to the cloud safety checker in the deployment server. Generally any safety assessment is a function of Safety Integrity Level (SIL) which is the proportional to the total number of goals, R_s is the safety

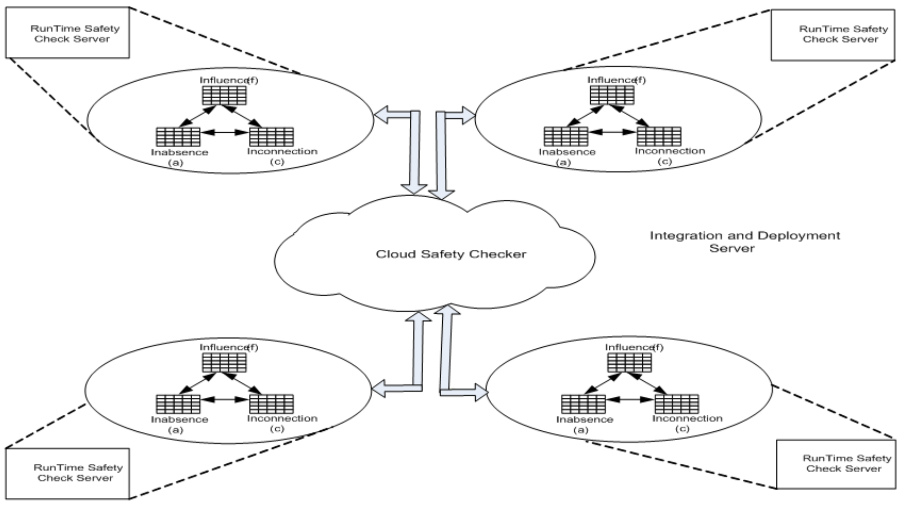


Fig. 5: Distributed safety assessment

requirement and m_s is safety method and is represented as:

F (SIL) £ {nsgcnrRsms/ncr}

It may also be represented in a probabilistic way as:

[1- p(hazard). Nsg Cnr/nc-1]

Where:

n_{sg} = The number of safety goals during the design stage
C_{nr} = The total number of safety constraints and
R_s = The safety requirement
m_s = The safety method

The Safety Integrity Level SIL can be calculated as 1-min {p (Frequency×Severity)} of each and every subsystem. The Safety of Automotive System can be evaluated by min {SIL^ number of Expected Safety levels ^ (number of Operational Safety levels/number of Safety Constraints levels(3))} or the safety for automotive software systems can be determined as the product of individual integrity level and the operational safety methods incorporated in the vehicle design as follows:

$$\text{Safety of Automotive System} = \min\{\text{SIL} \times \left(\frac{\text{Operational safety}}{\text{Expected sefety}}\right) \times \left(\frac{1}{\text{Safety constraints}}\right)\}$$

By assuming the value for very low = 0. 2, low = 0. 4, medium = 0. 6, high = 0. 8 and very high = 0.9 in the Tables 4-7. The safety of automotive system can be evaluated as below:

Max. Safety of Automotive System
(Air Bag Release at Speed 55km/h)
$= 0.8\times(5/6)\times(1/3) = 0.21$

Max. Safety of Automotive System
(Air Bag Release at Speed 80km/h)
$= 0.6\times(5/6)\times(1/3) = 0.16$

Max.Safety of Automotive System
(ABS Subsystem controlled at 95km/h
$= 0.8\times(5/6)\times(1/3) = 0.21$

Max.Safety of Automotive System
(ABS Subsystem controlled at 115km/h)
$= 0.4\times(5/6)\times(1/3) = 0.11$

The safety integrity level ,SIL can be calculated as [1-max {p(Frequency×Severity)}] of each and every subsystem and the safety of automotive system can be evaluated as:

$$\text{Safety of Automotive System} = \min\{\text{SIL} \times \left(\frac{\text{Operational safety}}{\text{Expected sefety}}\right) \times \left(\frac{1}{\text{Safety constraints}}\right)\}$$

Min. Safety of Automotive System
(Air BagRelease at Speed 55km h^{-1})
$= 0.6\times (5/6)\times(1/ 3) = 0.16$

Min.Safety of Automotive System
(Air Bag Release at Speed 80km/h)
$= 0.2\times(5/6)\times(1/3) = 0.05$

Table 8: Quantitative safety assessment for rules and relations

Type of safety component	Safety components	Safe Ranges	Safety assessment rule
Influence	Air bag control system	<40ms and >50km/h	WHN(Wherever),ALW(Always),BWY(By the way)
Influence	Seat belt monitor subsystem	50km/h	WHN(Wherever),BWY(By the way)
Influence	Tyre pressure checking	31/35psi	WHN(Wherever),EAE(Each and Every),BWY(By the way),SOT(Sometimes),NOT(Not at all times)
In-connection	Anti lock braking	<115km/h	WHN(Wherever),TOT(Totality),BWY(By the way)
In-connection	Adaptive cruise control	Increments of 5 or 10 km/h	WHN(Wherever),TOT(Totality),BWY(By the way)
In-connection	Traction control	120mph	WHN (Wherever), TOT(Totality),BWY (By the way)
In-absence	Forward collision warning	29kmph	WHN(Wherever),ALW(Always),AWY(Anyway)
In-absence	Infrared night vision activation	10m	WHN (Wherever), TOT (Totality), AWY (Anyway), NOT(Not at all times)
In-absence	Parking aid subsystems	>1.5m	WHN (Wherever), AWY (Anyway), SOT (Sometimes), NOT (Not at all times)
In-absence	Cornering brake control	5.88m/s2	WHN (Wherever), TOT (Totality), AWY (Anyway)
Influence	Roll over detection subsystem	134	WHN (Wherever), BWY (By the way)
Influence	Electronic stabilization on program	120	WHN (Wherever), ALW (Always), BWY (By the way)
In-connection	Brake by wire program	160bhp	WHN (Wherever), BWY (By the way)
In-absence	Lane departure warning module	50km/h	WHN (Wherever), TOT (Totality),AWY (Anyway)

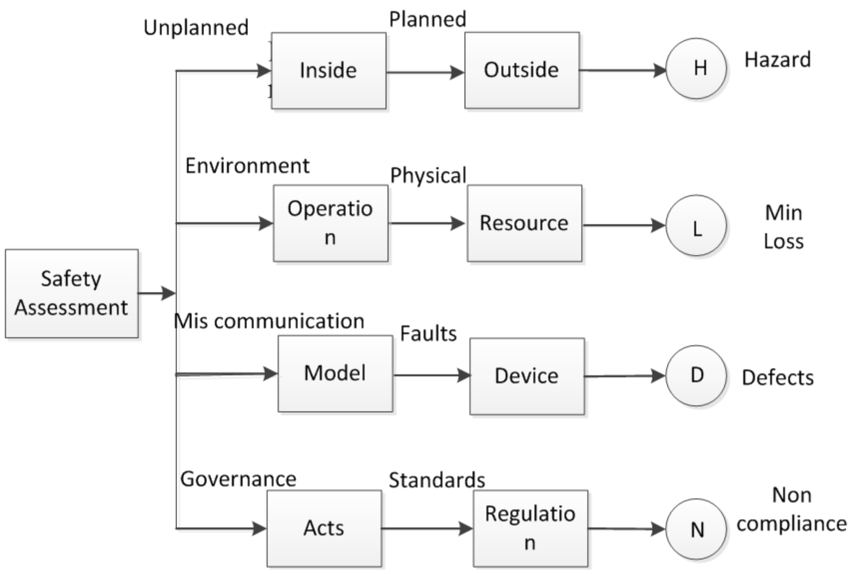


Fig. 6: Safety assessment distributed bounded tree

Min.Safety of Automotive System
(ABS Subsystem controlled at 95km/h
$= 0.4\times(5/6)\times(1/3) = 0.11$

Min.Safety of Automotive System
(ABS Subsystem ontrolled at 115km/h
$= 0.1\times(5/6)\times(1/3) = 0.02$

The Quantitative Safety assessment for Rules and Relation is shown in Table 8. In the case of a distributed automotive software systems, the safety assessment can be done by deciding the nature of operations, governance and the working environment with four major risks as Hazards, Loss, Defects and Non compliance as shown below in Fig 6.

Safety risk assessment by proposed insafe components: The risk is the probability of injury that may be as a simple expression but for a distributed application in the domain of automotive safety software, it can be the probability of injury is expressed as given by the equation as per the INSAFE model. The probability of injury can be determined as a complex quantity as per the role of insafe features like In-absence, In-connection, influential components as shown below.

$$\text{Probability of injury} = \sum_{i=1}^{A} op(\text{incidence})\Big[\sum_{j=1}^{B} po(\text{influence})\times \sum_{k=1}^{C} li(\text{inabsence}) + \sum_{k=1}^{C} li(\text{inabsence})\times\sum_{i=1}^{D} ch(\text{inconnection}) + \sum_{i=1}^{D} ch(\text{inconnection})\times\sum_{i=1}^{B} po(\text{influence})\Big]$$

Table 9: Computational factors in formal safety assessment model

Parameters	Factors	Conditions	Comparison
Assessment Parameter1	Likelihood(li)	Morning hours, cloudy and rainy days	Occasional = Low frequency component
Assessment Parameter2	Possibility(po)	Peak hours, bad road condition	Regular = Medium frequency component
Assessment Parameter3	Opportunity(op)	Over speed congestion	Often = High frequency Component
Assessment Parameter4	Probability(pr)	Collision , Engine failure	Sometimes = Pulsating
Assessment Parameter5	Chance(ch)	Hurt Blood injury	Casual = Composite frequency component

The incidence indicates the value of opportunity (incidence) where the value of opportunity is 0 to 1. The logical difference and conceptual differences between these factors are illustrated in the context of safety of the automotive system. For example, the opportunity of an incidence represented as op(incidence) needs to be maximum for system safety that implies the possibility represented as po(influence) needs to be minimum for safety. Similarly the likelihood of absence in a particular component needs to minimum for safety can be written as li(In-absence) needs to Min for safety implies the chance of an In-connection component ch(In-connection) needs to a minimum for safety. The concept can be logically arrived has been given a mathematical structure as:

$$\text{Safety} = 1\text{- pr(Injury)} = 1 - \left\{ \begin{array}{l} \text{Max}\left[\sum_{i=1}^{A} \text{op}(\text{incidence})\right] \times \\ \text{Min}\left[\sum_{j=1}^{B} \text{po}(\text{influence}) \times \right. \\ \qquad \left. \sum_{k=1}^{C} \text{li}(\text{inabsence})\right] \end{array} \right\} \tag{2}$$

The nature of the occurrence of any event may be associated with corresponding terms based on the context of the system, operational conditions and frequency of the events as shown in Table 9.

Traceability of safety and traceability for safety: The traceability of safety attributes due to a single safety requirement is a relation that connects the context, condition, action and reaction of each and every software component in all possible activation modes (Cacciabue, 2007; Farti and Albinet, 2010) (Chandrasekaran *et al.*, 2012). The traceability of safety components in an automotive application can be categorized into two major categories. They are Forward Expected Safety Traceability (FEST) and Reverse Expected Safety Traceability (REST). Forward Expected Safety Traceability helps to navigate the relationship chain from safety requirements to design a model or a code implementation model, whereas reverse expected safety traceability demands the identification of backward relationships of one attribute in the design model to the initial safety model. Both types of safety trace abilities are based on conditional execution, the action from specific components and its corresponding reaction of the system components under that particular mode. This top level relationship can be represented as an arrow"_ ". Thus the arrow represents traceability and when it is associated with the cause and effect sematics then, it may be called as sematic traceability that can be determined in either directions to debug and improvise the system. Semantic taceability implies that the actions of carrying over the events across the functionality of this system with its behaviour to track the sequence and consequence of the incidents, considering the context, condition, action, reaction and mode of operation and operator. The safety semantic traceability can be expressed as a set of implications shown below: The traceability arguments are based on the entities in the parenthesis. The reverse expected safety traceability is validating model changes or bugs for a new requirement: Given an original and changed version of a model, quality engineers restore the textual description of requirement, original and restored descriptions are compared. In forward traceability, the requirements are transformed into a quality model and can be represented as shown below: Forward Expected safety traceability→(leads) Requirement(safety, security, performance)→(leads) Design model(hazard,acess control, load balancing)→(leads) Execution model(interface, timing, valid jump)

For example, the safety requirements are transcripts into services or components with interfaces in the behavioural model and software modules in the safe design model. In the reverse expected safety traceability, the changed design models or modified code segments are translated into safety requirements and can be represented as shown below: Reverse Expected safety traceability→ (Trails) Execution model(interface, timing, valid jump)→ (Trails) Design model(hazard, access control, load balancing→ (Trails) requirement(safety, security, performance)

In forward expected safety traceability, the requirements are processed and added some weightages based on the primary quality objective of the application.

RESULTS AND DISCUSSION

The software safety assessment for the proposed model has been simulated using Reliasoft RENO, a simulation software for probabilistic event, safety and risk analysis for the results on the integration of insafe

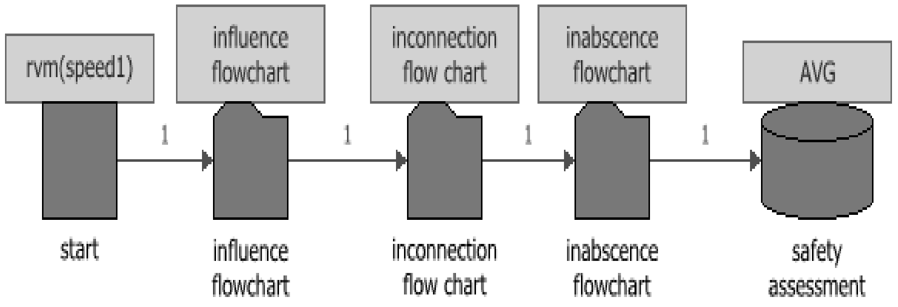


Fig. 7: In safe (influence,in-connection, in-absence) system model

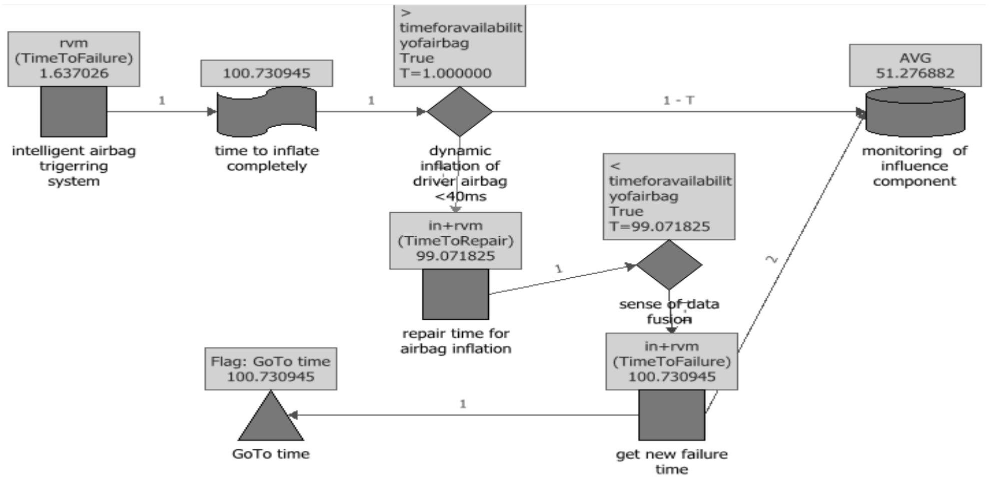

rvm (TimeToFailure) 1.637026
intelligent airbag trigerring system
100.730945
time to inflate completely
> timeforavailabilityofairbag True T=1.000000
dynamic inflation of driver airbag <40ms
1 - T
AVG 51.276882
monitoring of influence component
in+rvm (TimeToRepair) 99.071825
repair time for airbag inflation
< timeforavailabilityofairbag True T=99.071825
sense of data fusion
in+rvm (TimeToFailure) 100.730945
get new failure time
Flag: GoTo time 100.730945
GoTo time
1
1
1
1
~

Fig. 8: Airbag subsystem -Vehicle speed (influence monitoring component)

components approach where the variables like time for availability of air bag set to value 40 ms ,speed of limit of ABS set to 30 Km/hr, all led flashing variable set to 30 cm, green yellow and red variable set to 50 cm, green and yellow variable set to 100 cm, green variable set to 150 cm, speed variable set to 50 km/hr and parking speed variable set to 15km/hr. The four models where named like Time To Failure model which is created with weibull distribution of beta value = 40 and eata value = 100 sec, the second model Time To Repair is created with weibull distribution of beta value = 1 and eata value = 10 sec, the third model Speed1 is created with weibull distribution of beta value = 10 and eata value = 120 km/h and the fourth model Parking Speed1 is created with weibull distribution of beta value = 50 and eata value = 150 cm. In the distributed automotive software system, the software design components which are classified into three types as influence, in-connection and inabsence types and the services rendered by influential and in-connection components can transform an incident into an accident which in turn lead to hazards or mishaps are assessed by simulation as follows. Figure 7 describes Insafe system model which integrates the different components like influence, in-connection and in-abscence in the distributed automotive software system. The influence monitoring component shows the average time taken to deploy the airbag to inflate is <40 ms will be safe which is described in Fig. 8:

Traceability of influence by the air bag subsystem = intelligent airbag triggering time+time to inflate completely+ time for dynamic inflation of the driver

Traceability of influence by the air bag subsystem=Airbag is found to be< 40 ms

The in-connection monitoring component shows that the Anti lock Braking Control System (ABCS) applies >30 km h^{-1} and the speed of the vehicle is above 115 km is unsafe as described in Fig 9.

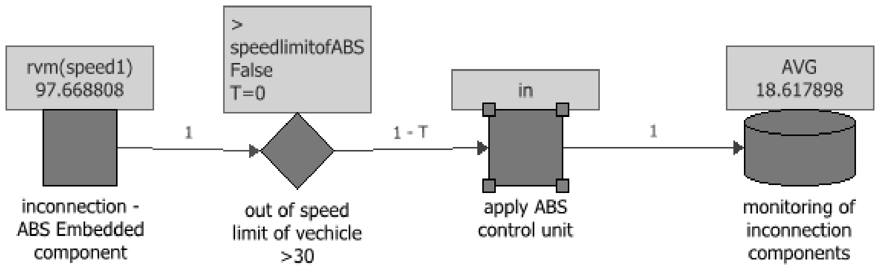

rvm(speed1)
97.668808
inconnection - ABS Embedded component
> speedlimitofABS
False
T=0
out of speed limit of vechicle >30
in
apply ABS control unit
AVG
18.617898
monitoring of inconnection components
1
1 - T
1

Fig. 9: ABS subsystem (In-connection monitoring component)

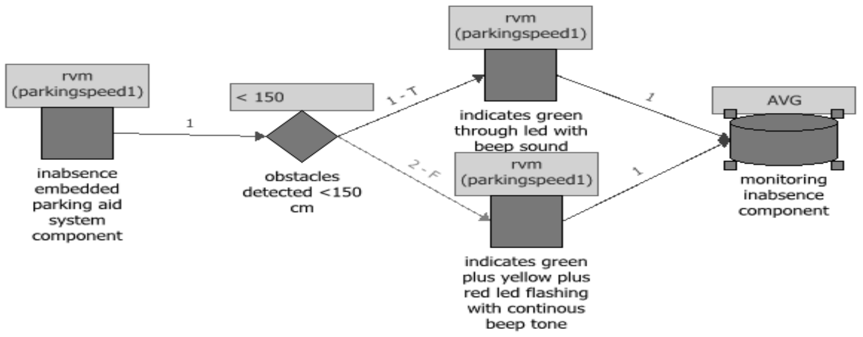

rvm (parkingspeed1)
inabsence embedded parking aid system component
< 150
obstacles detected <150 cm
rvm (parkingspeed1)
indicates green through led with beep sound
rvm (parkingspeed1)
indicates green plus yellow plus red led flashing with continous beep tone
AVG
monitoring inabsence component
1
1 - T
2 - F
1
1

Fig. 10: parking subsystem (in-absence monitoring component)

Table 10: Influence component-safety airbag subsystem

Vehicle speed in Kmph	Air bag release time in (ms)	Inflation of air bag component is SAFE or unrecoverable state
10	6.039	SAFE State
15	8.78	SAFE State
20	11.25	SAFE State
25	13.72	SAFE State
30	16.21	SAFE State
35	18.76	SAFE State
40	21.26	SAFE State
45	23.77	SAFE State
50	26.21	SAFE State
5	28.74	SAFE State
60	31.26	SAFE State
70	36.29	SAFE State
80	41.25	Unrecoverable State
90	46.26	Unrecoverable State
95	48.71	Unrecoverable State
100	51.27	Unrecoverable State

Traceability of in-connection by the
ABS subsystem = in-connection ABS
embedded component+out of a speed limit
of vehicle is >30kmph+apply ABS control unit
Parking subsystem (in-absence monitoring
component)-obstacles detected by distance

The in-abscence monitoring component shows that the obstacle detected less than the parking distance is 150 cm then green led glows with beep sound if the distance is 30 cm which is unsafe then green, yellow and red led flashes with continuous beep sound which is described in Fig. 10:

Traceability of in-absence by the parking system=
in-absence embedded parking aid system component+
Obstacles 150 cm (true condition)+
green LED with a beep sound=
in-absence embedded parking aid system component+
obstacles 30cm(true condition)+green and
yellow and red LED light with beep sound

The automotive software safety assessment of the design components and their local variations are identified with their distributions to influence to promote safety or avoid risk items.The distributed safety assessment documents are reported to a central server as

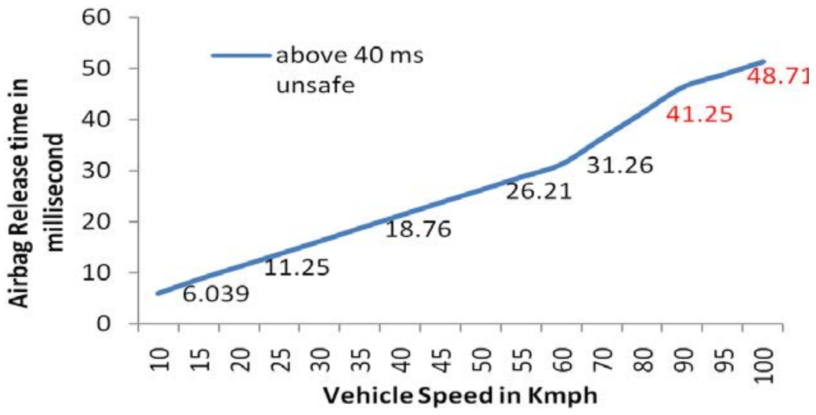


Fig. 11: Influence component- safety airbag subsystem

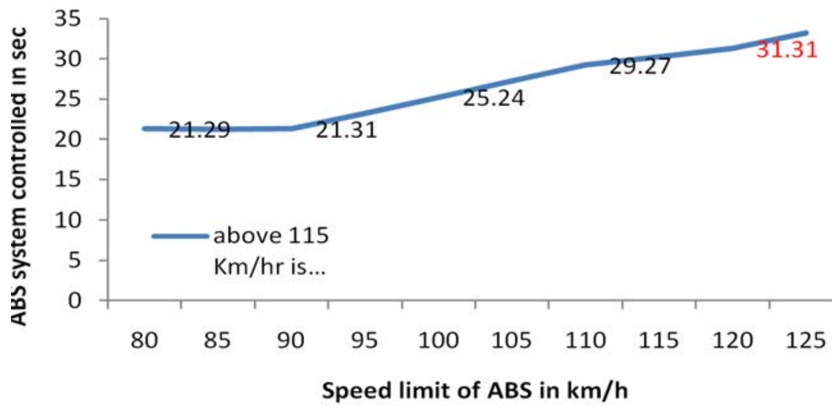


Fig. 12: In-connection component ABS subsystem

Table 11: In-connection component-ABS subsystem

Speed limit of ABS	ABS System controlled in Sec	ABS component activation is safe or unrecoverable state
80	21.29	Safe State
85	21.24	Safe State
90	21.31	Safe State
95	23.24	Safe State
100	25.24	Safe State
105	27.24	Safe State
110	29.27	Safe State
115	30.26	Safe State
120	31.31	Unrecoverable State
125	33.23	Unrecoverable State

Document as a Service (DaaS) for carrying out the safe transactions and alert messages across the system. The software safety assessment of the proposed model has been simulated using a tool, RENO from Reliasoft for the results of the integration of unsafe components approach. Safety components from each category are considered individually and the safety results due to air-bag release time, Anti lock braking system and parking distance parameters are tested under various environment conditions and the resultant graphs are discussed below. Table 10 describes the vehicle Speed (Influence Monitoring Component) – airbag time to inflate-average time to deploy <40 ms. If the inflation of Airbag component is in SAFE state means reliable otherwise unreliable. Figure 11 describes influence monitoring component say airbag , the relation between vehicle speed in kmph and airbag release time in millisecond was studied where the vehicle speed reaches till 70 kmph the airbag release time is linearly varying and then after 70 kmph the airbag time to deploy is faster which is safe.

From Table 11 describes the ABS speed (In-connection monitoring component) control unit

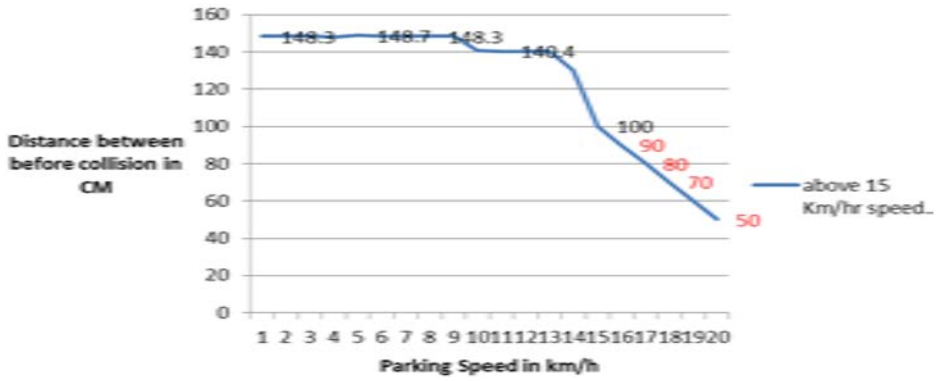


Fig. 13: In absence component – parking subsystem

Table 12: Parking subsystem (in- absence monitoring component)

Reverse Parking Speed in km/h	Distance between obstacle and Vehicle in centimetre	Indication of LED Green-150cm-SAFE Green and Yellow-100cm-MediumSAFE GreenyellowRed-50-UNSAFE
1	148.3	SAFE State green LED glow
2	148.3	SAFE State green LED glow
3	148.3	SAFE State green LED glow
4	148.2	SAFE State green LED glow
5	148.7	SAFE State green LED glow
6	148.6	SAFE State green LED glow
7	148.4	SAFE State green LED glow
8	148.3	SAFE State green LED glow
9	148.3	SAFE State green LED glow
10	140.7	SAFE State green LED glow
11	140.4	SAFE State green LED glow
12	140.4	SAFE State green LED glow
13	140	SAFE State green LED glow
14	130	SAFE State green LED glow
15	100	Recoverable state green yellow LED glow
16	90	Recoverable state green yellow LED glow
17	80	Recoverable State green yellow LED glow
18	70	Recoverable state green yellow LED glow
19	60	Recoverable state green yellow LED glow
20	50	Unrecoverable state Green YellowRed LED glow

average speed If the speed is above 115 kmph is safe. So, ABS component Activation is in Safe State and Reliable otherwise not reliable. Figure 12 describes In-connection monitoring component for ABS subsystem, the relation between speed limit of ABS in km/h and ABS subsystem controlled in seconds was studied. The speed limit of ABS subsystem is found to be constant till 90 kmph in the experimental studies and when the vehicle speed reaches 115 kmph, the ABS responds quicker than the earlier stages which is safer.

The Table 12 describes When reverse parking speed of the car is 5 kmph and the distance between the vehicle and obstacle is 148 cm, green LED glows to indicate the parking is safe (reliable). When the vehicle moves with 5kmph speed and the distance between the vehicle and the obstacle is reduced by 100 cm then the green and yellow led glows to indicate the parking of the vehicle is medium safe and reliable and it takes lesser time(in ms) to glow the GreenYellowRed LED when the vehicle moves with a speed of above 15kmph which is an unrecoverable state And an unreliable state. Figure 13 describes In-absence monitoring component, say parking subsystem , the relation between parking speed in kmph and distance between the obstacle and vehicles before collision measured in centimeters. In this scenario, the distance between the obstacle and vehicle till reaches 130cm is safe thereafter when speed is increased, the parking lighting subsystem indicates that it is unsafe.

CONCLUSION

A safety assessment model for an automotive software system design as an interacting distributed software components towards traceability is proposed as an In-Safe model and derived . The safety challenges in the automotive software is the composability of the distributed design components through safety assessment documents. The system safety can be thought of an in-presence or in-absence or in- connection nature of many software design components to minimize the transformation of an incident to an accident. The traceability of the fully assembled or partially assembled design software components after executing the software modules is derived as forwards encoded traceability and reverse encoded traceability in a real time software environment as a platform. The needed platform and the infrastructure components are assumed to be available on the test bed, the needed database for the testing component specifications are stored in the multiple data dictionary. The policies are incorporated to determine the safety of the platform and system to be tested as a composition of three categories of software components as In-absence, In-connection and influence categories. The research is substantiated with a proposed logic as a safety assessment logic with a set of decision rules. The set of proposed safety assessment rules as safety policies for software systems can be listed as a set of D-SAR primitives and their computational combinations. Safety assessment is done in the domain of automotive systems by checking the following properties in different subsystems or assemblies. They are Satisfaction property, Fairness property and Reachability property.

The Safety Assessment is achieved by evaluating the safety for automotive software system as the product of the individual integrity level of the components and the operational safety methods incorporated in the vehicle design. Thus the safety assessment documents are collected from different development centers and composed. The safety document is to designed to be portable one and at the same time the information and the data have to extracted by the computing node for further processing. The document model and the contents in a standard template have to be designed such that the same document can be reused for successive nodes with suitable modifications. The Safety Assessment Document (SAD) carries the important field of safety goal at that instant of processing and the method with which the goal achieved. The quantitative and qualitative safety assessments for an automotive system with distributive categorical and dynamic collection of safety documents for the three subsystems are AIRBAG subsystem, Antilock braking subsystem and Parking assistance subsystems are determined. The respective safety standards and compliances acts are cross checked and compared during the document verification phases .For the influence monitoring component say airbag, the relation between vehicle speed in kmph and airbag release time in millisecond was studied where the vehicle speed reaches till 70 kmph the airbag release time is linearly varying and then after 70 kmph the airbag time to deploy is faster which is safe. For the In-connection monitoring component for ABS subsystem, the relation between speed limit of ABS in km/h and ABS subsystem controlled in seconds was studied. The speed limit of ABS subsystem is found to be constant till 90kmph in the experimental studies and when the vehicle speed reaches 115 kmph, the ABS responds quicker than the earlier stages which is safer. For the In-absence monitoring component, say parking subsystem, the relation between parking speed in kmph and distance between the obstacle and vehicles before collision measured in centimeters. In this scenario, the distance between the obstacle and vehicle till reaches 130cm is safe thereafter when speed is increased, the parking lighting subsystem indicates that it is unsafe. Traceability of influence by the air bag subsystem is equal to the sum of intelligent airbag triggering time and time to inflate completely and time for dynamic inflation of the driver for the airbag is found to be <40ms. Traceability of in-connection by the ABS subsystemis equal to the sum of in-connection ABS embedded component and the time out of speed limit of vehicle is found to be <30 kmph and then apply the ABS control unit. Traceability of in-absence of the parking system is found when the distance in parking aid system component and the obstacles are at a distance of 150cm then green LED will glow with a beep sound and when obstacles are at distance of 30cm then all green , yellow and red LED light glowing with a beep sound. The automotive system safety is studied as a simulation using a standard tool RENO and the safety specifications are included in the tool, dictionary.

LIMITATIONS

The serious limitations of the above system model are the lack of performance assessment in terms of fuel, intertwined processes and time. A multicore programming model is very essential in solving the issues related to the parallel processing of all the data and safety documents on the fly. The optimized safety model for an automotive system with distributed components are studied with traceability. The safety requirements are transcripts fed into the services or components with standard interfaces in the behaviour of software modules in the safe design model. In the reverse expected safety, traceability, the changed design models or modified code segments are translated into safety requirements. The challenges in the safety assessment of an automotive system with distributed design models are due to heterogeneous requirements in terms of the system overall fuel efficiency, sub system minimum carbon emission rate, expense of optimal electrical power for air conditioner and infotainment, the aperiodic battery recharging, accurate tracking using GPS, in appropriate VANET interfaces. The limitations are also extened due to various types of the vehicles and their manufacturers and the system make of the year alongwith the country's dynamic driving rules and parking regulations.

REFERENCES

Bashir, K., F. Kanwal, A. Bashir and A.H. Ali, 2013. Software quality assurance for safety critical systems. Intl. J. Soft Comput. Software Eng., 3: 163-170.

Birch, J., R. Rivett, I. Habli, B. Bradshaw and J. Botham *et al*., 2013. Safety Cases and Their Role in ISO 26262 Functional Safety Assessment. In: Computer Safety, Reliability, and Security. Bitsch, F., J. Guiochet and M. Kaaniche (Eds.). Springer Berlin Heidelberg, Berlin, Germany, pp: 154-165.

Burton, S., J. Likkei, P. Vembar and M. Wolf, 2012. Automotive functional safety security. Proceedings of the 1st International ACM Conference on Security of Internet of Things, August 17-19, 2012, ACM, New York, USA., ISBN: 978-1-4503-1822-8, pp: 150-159.

Cacciabue, P.C. and O. Carsten, 2010. A simple model of driver behaviour to sustain design and safety assessment of automated systems in automotive environments. J. Appl. Ergon., 41: 187-197.

Cacciabue, P.C., 2007. Modelling Driver Behaviour in Automotive Environments: Critical Issues in Driver Interactions with Intelligent Transport Systems. Springer-Verlag, Berlin, Germany, ISBN: 978-1-84628-617-9, Pages: 428.

Chandrasekaran, S., F. Faizunnisa, H.H. Duba, N.S. Hemalathas and A. Abeetha, 2010. Safety assessment model and metrics for accidents in road transport system. Proceedings of the 5th International IET Conference on System Safety, October 18-20, 2010, IET, Manchester, England, pp: 1-6.

Chandrasekaran, S., F. Faizunnisa, H.H. Dubal and N.S. Hemalatha *et al*., 2012. Viewpoint based reverse semantic traceability in software quality requirement engineering. Eur. J. Sci. Res., 87: 66-81.

Chandrasekaran, S., P.V. Raman and R.S. Vijayravikumaran, 2011. CAR based safety model in automotive software engineering. Proceedings of the 10th WSEAS International Conference on Software Engineering, Parallel and Distributed Systems, February 20-22, 2011, WSEAS, Stevens Point, Wisconsin, USA., ISBN: 978-960-474-277-6, pp: 206-211.

Frati, M.A.P. and A. Albinet, 2010. Requirement traceability in safety critical systems. Proceedings of the 1st ACM Workshop on Critical Automotive Applications: Robustness & Safety, April 28-30, 2010, ACM, New York, USA., ISBN: 978-1-60558-915-2, pp: 11-14.

Gandhi, T. and M.M. Trivedi, 2007. Pedestrian protection systems: Issues, survey and challenges. Intell. Transp. Syst. IEEE. Trans., 8: 413-430.

Haider, A.A. and A. Nadeem, 2013. A survey of safety analysis techniques for safety critical systems. Intl. J. Future Comput. Commun., Vol. 2,

Jesty, P.H., D.D. Ward and R.S. Rivett, 2007. Hazard analysis for programmable automotive systems. Proceedings of the 2nd Institution of Engineering and Technology International Conference on System Safety, October 22-24, 2007, IET, Stevenage, England, pp: 106-111.

Kafka, P., 2012. The automotive standard ISO 26262, the innovative driver for enhanced safety assessment & technology for motor cars. Procedia Eng., 45: 2-10.

Leveson, N., 2011. Engineering a Safer World: Systems Thinking Applied to Safety. Mit Press, Cambridge, Massachusetts, USA, ISBN: 978-0-262-01662-9, Pages: 531.

Nancy, G.L., 1995. Safeware System Safety and Computers", A Guide to Preventing Accidents and Losses Caused by Technology. Addision-Wesley Publishing Company, Boston, Massachusetts, ISBN: 9780201119725, Pages: 680.

Pecht, M., A. Ramakrishnan, J. Fazio and C.E. Nash, 2005. The role of the US national highway traffic safety administration in automotive electronics reliability and safety assessment. Components Packaging Technol. IEEE. Trans., 28: 571-580.

Pretschner, A., M. Broy, I.H. Kruger and T. Stauner, 2007. Software engineering for automotive systems: A roadmap. Proceedings of the IEEE Conference on Future of Software Engineering, May 23, 2007, IEEE Computer Society, Washington, USA., ISBN:0-7695-2829-5, pp: 55-71.

Rierson, L., 2013. Developing Safety-Critical Software. A Practical Guide for Aviation Software and DO-178C Compliance. CRC Press, Boca Raton, Florida, Pages: 567.

Schlummer, M., D. Althaus, A. Braasch and A. Meyna, 2010. ISO 26262-the relevance and importance of qualitative and quantitative methods for safety and reliability issues regarding the automotive industry. J. KONBiN., 14: 165-176.

Stringfellow, M.V., N.G. Leveson and B.D. Owens, 2010. Safety-driven design for software-intensive aerospace and automotive systems. Inst. Electr. Electr. Eng., 98: 515-524.

Wabmuth, M., S.C. Stilkerich and E. Lubbers, 2011. Distributed safety assessment for airborne systems: An industrial relevant approach for automated safety analysis and reporting. Proceedings of the International ACM Workshop on Security and Dependability for Resource Constrained Embedded Systems, September 22, 2011, ACM, New York, USA., ISBN: 978-1-4503-0882-3, pp: 1-5.

Wang, Y. and Q. Zhang, 2013. Design of an active automotive safety system. J. Eng. Sci. Technol. Rev., 6: 155-159.