CAR Based Safety Model in Automotive Software Engineering

CHANDRASEKARAN S, VIJAYA RAMAN P, VIJAYRAVIKUMARAN R.S
Department of Information Technology
Rajalakshmi Engineering College, Jeppiaar Engineering College, Valliammai Engineering College
Anna University of Technology, Chennai, TamilNadu
INDIA
chandrasekaran_s@msn.com, vijayaramanp@gmail.com, vijayravikumaran.rs@gmail.com

Abstract: - The objective of the paper is to propose a design for safety model in automotive software architecture focusing the context awareness features, user actions and unexpected reaction from the environment. 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. The safety feature proposed in the work is achieved through a V-W software development model and a context-action-reaction safety logic is used which is satisfied using sequent calculus. The hierarchical object oriented design (HOOD) methodology is adopted in the architecture of safe software. The safety due to functionality and behavior of the critical modules like airbag control and tire pressure monitoring are considered in the work and implemented through interacting finite state machines (FSM) as per the existing system and software safety standards. The software safety based on the operational context, user actions and the system reactions in the automotive applications are considered.

Key-Words: - Safety logic, safety satisfaction, context awareness, sequent calculus, HOOD, FSM

1 Introduction

Software safety is the systematic approach to identify, analyse and track software mitigation that prevents system hazards and mishaps. The non-occurrence of catastrophic consequences on the environment leads to safety of the system. The software safety controls the hazards and hazardous functions to ensure safer software operation within a system [1]. In general, automotive embedded software has been closely coupled not only with the electronic hardware but also with electromechanical sub systems. The functional boundary of the application software that may reside in different hardware chips is not clear. The earlier proposed automotive software architecture (AUTOSAR) is composed of three layers, such as application software component (SWCs), runtime environment (RTE) and basic software module (BSW) modules. The SWCs can encapsulate internal behavior of applications and their communications that is handled in a style of client-server through standard interfaces [2].The essential quality of the automotive system through the embedded software is safety that can be assured only through an extra cost. But safety and cost are conflicting as the safety level is elevated, the cost increases through mandatory high speed communication channels between critical components. That is the communication performance is required in order to guarantee a certain response time for safety-critical operations. The categorization of aspects such as safety along with collaboration trace with other components helps to understand the design goals of each component. Hence a timely action based on the context of operation either hidden or explicit determines the safety of the software system in the design stage. The main issue in assuring safety is the quicker response of the system components and expected reaction from the subsystems in a dynamic environment. The earlier safety model concentrates a real-time distributed architecture, which has to satisfy multiple interrelated non-functional requirements such as safety, performance and cost [3]. In all model based development strategies and automatic code generation technologies, there is a semantic gap between models and implementation code developed [4]. The AUTOSAR provides standard interfaces through virtual function bus (VFB) technique for their automotive embedded system and the efficient communication was established between software components[5]. The safety in the design of automotive software indicates that errors which could reduce the safety of the system have been eliminated or controlled. The safe design with minimum error can be achieved only through systematic design and validation not only in the architectural perspective but also in the timely action perspective. The correct data, minimum functions or actions in the given context and safe behavioral reactions of the software system assures the product and people safety.

The paper is organized as follows: Section 2 focuses the need for a safety oriented system development life cycle model for automotive software and its sub sections explains the proposed context-action-reaction (CAR) logic with temporal operators with their equivalences to satisfy safe design logic. Section 3 discusses safety
the functional software phase functions of airbag system, wiper control, tire pressure monitoring and seatbelt indication functions are carried out whenever it is needed.

2 Automotive Software System Safety

Automotive software safety is a subset of system safety and if there is an omission or commission fault in the design of the software, it might lead to system failure that may cause serious problem not only to the system but also to the user or passenger. Hence the architecture pattern used in such software bundle and the algorithm employed in each of the functional module should be failure-safe. The objective of the research work is to propose a safety enhancement model in the architecture of the distributed automotive software systems. The focus on the interoperability and computational complexity issues of heterogeneous software components running on different electronic hardware units like micro controllers and signal processors. Apart from safety the security issues like unauthorized access to system components like hardware, software and network elements can result in an accident. The conventional software development models are not integrating the safety aspects of system as well as software components based on the context of usage. The environmental issues, the driving experience and road traffic conditions are also to be integrated for a successful design of automotive software. The need of an integrated software development process model encompassing all the new requirements along with the existing ones is essential in the case of changing customer requirements not only to satisfy but also to save them. The V-W software safety model shown in Fig.1 proposes the software safety aspects in which requirement engineering, system design, software design, software verification, system verification and system validation are carried out. The V-W model has three phases namely, automotive environment, knowledge control software and functional software. Each phase is responsible for its functional and non-functional requirements taking into consideration applications including third party software components. In automotive environment phase user-level interactions, signal processing, and obstacles are addressed. Security aspect in the same environment is dealt in a careful manner. Knowledge control software phase deals with the interactions, interfaces, inhibit and interferences taking place in an automotive software environment. In

![Fig.1 Automotive software safety V-W Model](image)

2.1 Safety-Context Action Reaction (CAR) Logic

The proposed CAR Grammar is given by:

\[
\text{CAR} :: T\alpha|\beta|\gamma
\]

Where C represents the context or operational environment of the system, A represents Action by the automotive software and R represents reaction from the user \(\alpha, \beta, \gamma\) represent the various automata through which the safety related software components follow. For example, The bad road conditions or foggy weather conditions while driving is a typical context C or environment outside the system and the over speed driving and checking the fuel conditions are the actions A by the vehicle driver whereas the air bag release and backdoor opening indication are some of the reactions R of the system. The Grammar based on the context can be represented in the temporal logic as:

\[
\alpha :: p|E\alpha|U\alpha|Y\alpha
\]

where \(p \in \alpha\), E represents “Eventually”, U represents “Unlikely” and Y represents “Rarely”. p,q and r are propositional atoms. The Grammar based on the action in the temporal logic is given by:

\[
\beta :: q|I\beta|K\beta|O\beta
\]

where \(q \in \beta\) is true, I represents “Immediately”, K represents “Knowingly”, O represents “Occasionally”. The
Grammar based on the reaction for the temporal logic can be represented as:

\[ \gamma ::= r|S|B|Y|L|\gamma \]

Where \( r \in \gamma \), \( S \) represents “Suddenly”, \( B \) represents “Accidently” and \( L \) represents “Likely”. The three different grammars and the resulting automata are interacting to give a reaction based on a particular context and actions. For example, consider the statement “Eventually the tire pressure will reduce after four hours of travel”. This can be represented in the CAR logic as follows:

\[ \exists t(\text{travel}(t-4,t)) \rightarrow \forall w( E.\text{reduce}(\text{pressure}), w \in \text{Wheel}) \]

Considering another statement “Its likely that the anti-lock breaking system(A) will be activated when the driver applies a forceful brake(F)” . This is represented as:

\[ \forall t \exists F(\text{Apply}(F,t)) \rightarrow \exists t( L.\text{activate}(A,t1)), t1 > t. \]

The statement “The air bag(a) should blow immediately after a heavy hit occurs in the hood(h)” is represented as:

\[ \exists t(\forall h \text{heavyhit}(h,t)) \rightarrow I.\text{activate}(a,t)). \]

In case of either bad weather(b) or a dark road(d) the driver occasionally uses night view assist(N)

\[ \exists t( O.\text{Environment}(b,t) \lor O.\text{Environment}(d,t) \rightarrow \forall System.\text{Deploy}(N,t)) \]

Similarly the operators declared in the CAR grammar can be used to specify the syntax and semantics of the operations needed to run a safe drive. At the same time it is essential to find out the equivalent relations between the logical expressions to decide or not to decide an operation by the automata developed under this logic. Small letters represent terminal variables and capital letters represent non-terminal variables. The tuples of the interacting Finite State Machine (FSM) are represented as:

\[ \text{m} = \{C, Q, \Sigma, \delta, R\} \]

where,

\( C \) is the start state of the FSM
\( Q \) is the set of all states in the FSM such that \( Q = \{C \times A \times R\} \).
\( \Sigma = \{C_{in}, A_{in}, R_{in}\} \) is a set of input symbols.
\( \delta = Q^{*}\Sigma \) and it is the transition function of the machine.
\( R \) is the final state.

The set of states \( C, A, \) and \( R \) contain the following states as shown below:

\[ C = \{C_{1}, C_{2}, \ldots, C_{n}\} \text{ Where } n \text{ is the maximum number of states in } C. \]

\[ A = \{a_{1}, a_{2}, \ldots, a_{m}\} \text{ Where } m \text{ is the maximum number of states in } A. \]

\[ R = \{r_{1}, r_{2}, \ldots, r_{o}\} \text{ Where } o \text{ is the maximum number of states in } R. \]

The interacting FSMs shown in Fig.2 consist of states Context(C), Action (A), Reaction(R) and their respective sub states. The machine takes context as the input, and based on the context or environment at that instance a particular action is selected and executed by human. For every action a suitable reaction is chosen by the automaton and the machine releases the output. The CAR logic is explained as follows:

1. \( E\alpha \equiv \sim I\beta \) // For any eventual context, don’t do any immediate action.
2. \( I\beta \equiv S\gamma \) // Immediate action results in sudden reaction.
3. \( K\beta \equiv \sim B\gamma \) // An action happening knowingly implies that the reaction does not happen accidentally.
4. \( U\alpha \equiv \sim L\gamma \) // In an unlikely context, the reaction is not likely to take place.
5. \( \sim (B\gamma \lor K\beta) \equiv \sim B\gamma \land \sim K\beta \) // Not that the reactions occur accidentally or actions occur knowingly implies that the reactions don’t occur accidentally and actions don’t occur knowingly.
6. \( \sim (B\gamma \land S\gamma) \equiv \sim B\gamma \lor \sim S\gamma \) // Not that the reactions are accidental and sudden implies that the reactions are either not accidental or they are not sudden.
7. \( (I\beta \lor K\beta) \land L\gamma \equiv (I\beta \land L\gamma) \lor (K\beta \land L\gamma) \) // An action that occurs immediate or knowingly with a likely reaction implies that either the action is immediate and reaction is likely to occur or the action occurs knowingly and the reaction is likely to occur.
8. \( \sim \forall i O\beta(i) \equiv \exists i \sim I\beta(i) \) // Not all actions occur occasionally implies that there are few actions that occur immediately.
9. $\exists i K\beta(i) \equiv \forall i \sim B\beta(i)$ //Not some actions take place knowingly implies that all actions do not happen accidentally.

Table 1 CAR safety logic table

<table>
<thead>
<tr>
<th>Rule</th>
<th>Expression</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td>bSd→bNd</td>
<td>'S' indicates system/software, 'b' stands for bad weather, 'd' for darkness/</td>
</tr>
<tr>
<td>2.</td>
<td>bNd→bsd</td>
<td>'s' indicates safe, 'N' represents night view assist/</td>
</tr>
<tr>
<td>3.</td>
<td>eF→eA</td>
<td>'e' represents emergency, 'F' represents forceful brake, 'A' for antilock brake system/</td>
</tr>
<tr>
<td>4.</td>
<td>Ho→ao</td>
<td>'H' represents heavy hit, 'o' represents over speed, 'a' represents airbag controller/</td>
</tr>
<tr>
<td>5.</td>
<td>fT→Fr</td>
<td>'f' represents four hours, 'T' for travel, 'R' for reduced tire pressure/</td>
</tr>
<tr>
<td>6.</td>
<td>fR→fi</td>
<td>'i' stands for intimation/</td>
</tr>
</tbody>
</table>

For example, some of the premises in safety scenario are given and they are verified through satisfaction properties using sequent calculus.

1. Bad weather results in dark road. This is represented as $b \vdash d$.
2. When the road is dark and if the system travels with over speed a heavy hit might occur. This is represented as $d, o, S_2\vdash H$.
3. When a heavy hit occurs due to over speed an air bag will get released. $o \Rightarrow H \vdash a$.

Proof:

- $b \vdash d$
  - LW
  - $S_1, b \vdash d$
  - Cut rule
  - $S_1, b, o, S_2\vdash H$
  - $R\Rightarrow$
  - $S_1, b, S_2\vdash o \Rightarrow H$
  - $o \Rightarrow H \vdash a$
  - Cut rule
  - $S_1, b, S_2\vdash a$

In another scenario, the automotive system safety can be seen as follows: When going on a long journey ($L_2$) in a highway ($H$), check the tire pressure ($T$) and also fill fuel ($F$). 5. During night ($N_1$) journey the road will be dark and hence the headlight ($O_H$) should be switched on. 6. If the tire is checked, the fuel is filled and a back door indication ($I_{BD}$) exists then the system is in a safe(s) state.

Proof:

- $L_2, H \vdash T, F$
  - LW
  - $L_2, H, N_1 \vdash T, F$
  - Cut rule
  - $L_2, H, N_1 \vdash T, F, D_R$
  - $D_R \vdash O_H$
  - $L_2, H, N_1 \vdash T, F, O_H$
  - $R\Rightarrow$
  - $L_2, H \vdash T, F, N_1 \Rightarrow O_H$
  - $L_2, H \vdash F, N_1 \Rightarrow O_H \land I_{BD}$
  - $T, F, I_{BD} \vdash s$
  - $L_2, H \vdash N_1 \Rightarrow O_H, s$
  - Cut rule

3 Software Safety Standards

There are many standards that focus on the software safety in the automotive industry like DO-178B, DEF STAN 00-55 and ARP 4754\textsuperscript{12} inter alia. None of these standards identify the complexity added by implementing redundant modules that may introduce new hazards(such as Byzantine faults)related to this additional complexity. The problem of software modification is the situation in which software has been developed into a number of software product lines. The current software safety standards do not address the areas of new technology or emerging techniques like artificial intelligence that is almost needed in any safety-critical application [6]. The vital point in enhancing the safety of software systems is to avoid noncompliance components, dead codes and governing the risks through mitigation approach. The noncompliance, safety governance and safety constraints lead to risk analysis. Due to recent development in technologies the growth of hardware becomes faster. The hardware starts to degrade when they are created so the maintenance becomes an important criterion which restores the degraded quality to its original designed level. The hardware heterogeneity is making software architecture to adopt the rapidly changing hardware components such as sensors and actuators. One of the best ways to develop and deploy safety software is based on the software component modeling. The components can be designed with safe and secured interfaces to interact with all other components. The design of multi component activation helps the automotive system to coordinate in a synchronized way to solve typical problems of decision making. Apart from the reliability of the designed components, the functional safety is monitored based on
the software behavior. The multitude of requests from software components to the controller are prioritized and buffered to take an appropriate action based on the anticipated reaction from the system. The target environment is based on sensors and how the application software responds in a safer manner when more number of sensors are trying to enable for different functions like tire pressure indicator, automated seat belt sensor, automated wiper. The software must decide when to activate A.C, wiper, airbag and indication of fuel, tire pressure. Safety leads to performance degradation. Safety performance factor is defined as the average human-hour unit lost due to occupational accidents/incidents, including fatalities, first-aid incidents, bruises and cuts. Complexity is categorized into time complexity, spatial complexity, temporal complexity, dynamic complexity and differentiated complexity.

3.1 Software Safety Model
An automotive software safety system can be viewed in different levels. Basically the system is viewed as hardware, software, mechanical and environment domains in the safety domain perspective. In this software domain is focused, with safety element view and elements involved in this view are real-time operating system (RTOS), third-party components and application software. The application software element includes the context, action and reaction methods through which safety of the software is improved. The above safety model is designed with Hierarchical Object Oriented Design (HOOD) methodology which basically uses classes and objects to represent the model.

Fig.3 HOOD Model for CAR
The design includes MEM sensors, an Electronic Control Unit (ECU), a Context Analyzer, an Action Verifier and Reaction Checker modules. The MEM sensor takes the input to the Electronic Control Unit when it reaches threshold value, the corresponding automotive device is activated. The ECU analyzes the context that it experiences, verifies the action taken and checks for the corresponding reaction. The HOOD modules are triggered for their operations through the ‘input’ of the MEM sensor module as shown in Fig.3.

3.2 Safety Enhancement by CAR
In any software system model all the possible hazards have to be identified in the analysis stage and the probabilities of occurrence of hazards and the severity has to be considered in the enhanced safety specification. The design criteria that are to be followed to ensure safety may have many constraints that may create negative impact on safety performance. The parallel composition of software running on different platforms and software aging will be considered to design a safe system. The software safety analysis model must provide the necessary inputs to software safe design and development. The safety design requirements, implementation, recommendation or design changes can be incorporated with in to the software with minimal impact. The distributed computer system refers to a number of general and special purpose processing software modules connected by inter and intra process communication. The co-operative distributed problem solving is a network of semi autonomous processing software nodes working together to solve a problem typically in a multi agent systems. The model is for distributed, service oriented or component based system any incident may become an accident due to unsafe design invocation of remote methods. Stochastic or bio inspired model encompassing software action and reaction may also be proposed. The unused or remote dead code may act as a hazard that will create a mishap in the loosely coupled distributed system. The concurrency and interprocess communication in the system may induce safety related faults. The design requirements, functionally derived safety design requirements (based on hazard causes), test requirements to produce evidence for the elimination and/or control of the safety hazards, and the identification of safety requirements pertaining to operations and support of the product. The context based safety enhancement describes the situation that the customer experiences and task faced with when, where and what details whereas the action based safety enhancement describes how an action carried out in any context. The reaction based safety enhancement is about following rules to achieve the desired results and the actions to be performed under different scenarios as mentioned in Fig. 4. The credibility of software safety engineering activities within the hardware and software
development project depends on the credibility of individual(s) performing the managerial and technical safety tasks. It also depends on the identification of logical, practical, and cost effective process that produces the safety products to meet the safety objectives of the program. The primary safety process include hazard analyses, initial safety design requirements (based on hazard causes), test requirements to produce evidence for the elimination and/or control of the safety hazards, and the identification of safety requirements pertaining to operations and support of the product.

In the proposed safety model of automotive software, an order of precedence is followed in the design steps. The design model aims to reduce the minimum risk by the incorporation of more safety devices and alarm signals to the users for proactive safety. The procedures are to be followed to get the maximum usability of the proposed design model when it gets implemented in the automotive systems.

4 Conclusion

The research work not only aims to undertake an analysis of the existing safety standards and models but also propose new safety software development model. The various factors originating from the hazardous environment, lazy actions of the driver and the unexpected behavior of the software system are considered to design a safe design model for automotives. The limitations of the current safe design model are the challenges due to heterogeneity in the design of various brands and models of automotives and the unpredictable environmental conditions. Even an experienced user may perform some quick actions to which the unexpected reactions are really dangerous. In the future work, the driving patterns of different categories of users and the predictable traffic data will be included in the architectural model in order to give not only an action safety but also the system reaction safety.

References: