ISSN 2347-4289

Development Of Automatic Parser Tool For Safety Critical Automobile Software

R Monica, Dheeraj D, V. K. Agrawal

M-tech (SE), Asst Professor, Professor
Department of Information science Engineering
PES Institute of Technology
Bangalore-560085
Email: monica.ravishankar@gmail.com, dheerajd@pes.edu, vk.agrawal@pes.edu

ABSTRACT: In accession to the growing use and complexity of the software in safety critical automotive systems, the testing for its safety violation became more imperative. This necessitated the development of an automated parser tool for the software of the automotive system in order to ensure its safety. This tool is developed with intent to verify and validate whether the automobile system software which is developed using the C programming language have followed the coding rules and guidelines. These rules are drawn through the international standards which are in relation to the safety of the automobile system and the MISRA-C standards. This paper provides a concrete idea about the development of the parser tool with the supportive architecture, design of the coding rules and the test results.

Keywords: Automotive Systems, Automated parser tool, C programming language, International Standards, MISRA-C standards.

1. INTRODUCTION

With the growing dependency of the safety critical automotive systems on the computer software, it is anticipated that the need for testing its safety violation has become more crucial. Thus the safety and reliability of the vital software being embedded in the automotive systems started to become more significant as an important issue. In this way, for the safety and reliability of the automotive system software, the coding part which can have an effect on the safety at the coding level must be checked at the software assessment stage itself [1]. In the automotive field, the coding rules and standards which must be observed by software programmed in C language in the control units of automobile field are standardized in the form of "MISRA C" [2]. Similarly the safety requirements necessary for the software of automotive systems are standardized by IEC 62278 and IEC 62425 [3], [4]. Although there are myriad good reasons for C turning out to be the most prominent higher-level programming language—such as its support for high speed, low-level I/O operations, and compact compiler-generated code—programming in C does not guarantee problem- free code. C can also be written in a very condensed, hard-to-comprehend manner, which dramatically increases the likelihood of introducing errors. Additionally, a standard C compiler does not necessarily detect many small typing errors. Even if the developers are fairly knowledgeable about the language, not all C programmers are fully aware of the side effects of all possible constructs. Still for all the good reasons, including its broad availability across microprocessors ANSI/ISO C is a well accepted standard for higher level language programming of embedded systems. But as discussed, C is also considered to be unsuitable for programming safety-related applications. Thus industries developing safety-critical embedded applications have found ways to overcome the drawbacks of C—primarily by means of written guidelines with the intent of reducing the probability of coding errors by restricting the use of error-prone C constructs. Based on the experience and expertise in automotive applications, enhanced code-checking facilities that conform to the MISRA C guidelines have been developed to enforce best programming practices. One such is the development of the testing tools for the industrial embedded software coding rules, which is the main intent of this study. This automated testing tool for coding rules for vital software would verify and validate whether coding rules for testing the safety of railway signaling system software are suitable or not [5]. The remainder of this paper is structured as follows: Section II describes the architecture of the automated source code analysis tool of the automobile system. Followed by which the design of the coding rules of the prescribed tool is discussed in section III. Section IV deals with the implementation of the automated tool with supportive test results. Finally the paper concludes with future work in section V.

2. ARCHITECTURE OF THE SOFTWARE PARSER TOOL

The automated testing tool for coding rules of the automobile software which was suggested and implemented by this study is defined as the “Parser Tool for Automotive Software”. This module enables the automobile system software to judge if it is suitable for the required rule by checking whether it is developed by following the coding rules required by relevant safety standards. This section describes the architecture of the automated source code analysis tool of the automobile system. The configuration diagram of the parser tool for the automotive system software suggested through this study is depicted in figure 1. As shown in the figure, the tool is designed to present results through report generator. This is done after analyzing whether the object source code input through prescribed parser tool after processing it using a compiler is suitable for the required coding standard by analyzing the input source code. If the rules are found to be violated it would generate a warning indicating the user to take precautions. The source code is once again processed by the compiler even after being checked against the MISRA C to confirm that the syntax and the semantics of the c source code is unaffected by the MISRA C rule checking.

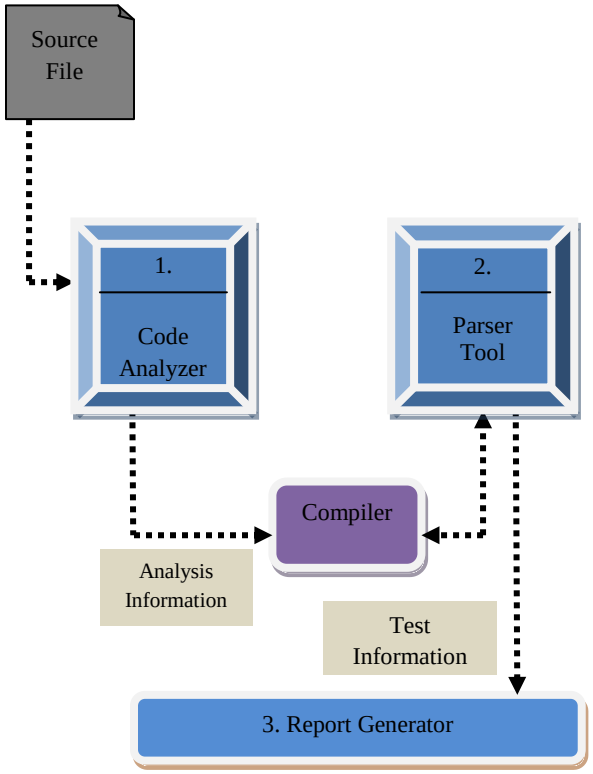

Figure 1: Configuration Diagram of the Developed Module

3. DESIGN OF THE CODING RULES

Though C is a well accepted higher level programming language of embedded systems it is proven to be unsuitable for programming safety-related applications. MISRA defined a total of 127 programming rules that are applicable when developing safety-related applications in C. Among the whole coding rules presented by this MISRA-C, the items possible to be automated were implemented in this development tool, whose small sample is highlighted in table1.

Table 1: Sample of the list of the rules implemented by the development tool

Rule	Type	Category
Comments shall not be nested.	Comments	Required
Identifiers in an inner scope shall not use the same name as an identifier in an outer scope, and therefore hide the identifier.	Declarations and Definitions	Required
All automatic variables shall be assigned a value before being used	Initialization	Required
The right-hand operand of an && or || operator shall not contain side effects.	Operators	Required
Floating-point variables shall not be tested for exact equality or inequality.	Expressions	Required

The parser tool checks to see if C source codes deviate from any of the MISRA C rules. If a deviation is detected, the tool outputs a report message. The figure below demonstrates a sample program which is to be inspected by the tool; a report message similar to the following will be output. It indicates that the use of the break statement on the appropriate line of the program deviates from the specified rule which is depicted in figure 3.

```
void func(void)
{
  while(port1 != 0)
  {
    if (port2 == 0)
    {
      break;
    }
  }
}
```

Figure 2: Sample program inspected by the tool

MISRA (58) warning: test.c, 10 - 'break' statement shall not be used except in a switch statement.

Figure 3: Report Message

ISSN 2347-4289

4. IMPLEMENTATION OF AUTOMATED PARSER TOOL

The parser statically checks C source codes to find the source codes that deviate from any of the rules listed in this document. Initially the parser tool is started from a compiler driver as a part of compile operation for syntactic and semantic verification which is projected in figure 4. The rules listed in this document are regarded as the guidelines and can be used as inspection items of source code review. Any deviations from these rules can be detected by parser tool which checks C source codes against each of the rules, and reports the locations where codes deviate from any of the rules which are known as violated locations. These are interpreted in the user interface diagrams where the login screen is shown in figure 5 followed by the screen for browsing the required input file depicted in figure 6 and 7. The output of the inspection result is directed to a report file, or to standard error as a report message which is portrayed in figure 8. Finally the source code is processed by the compiler even after being checked against the rules.

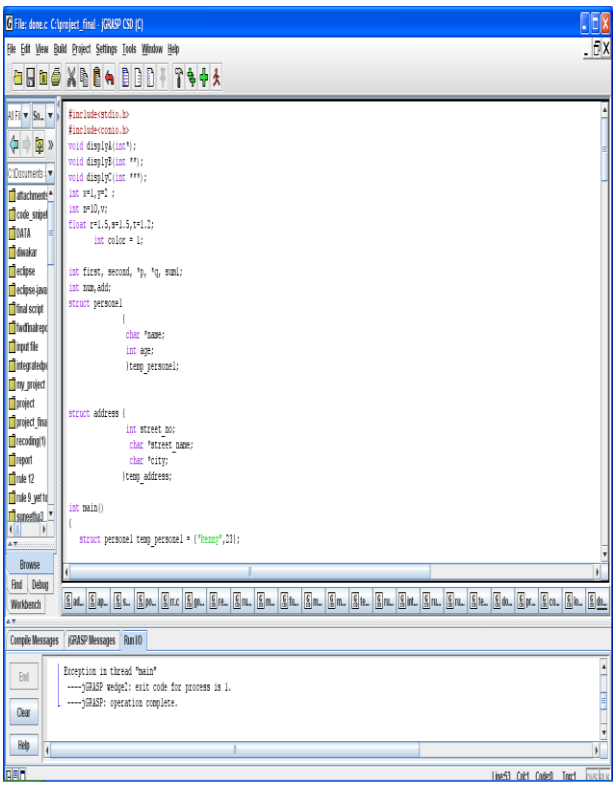

Figure 4: Screen which shows the production and usage of a compiled c source code as the input

Figure 5: Login Screen

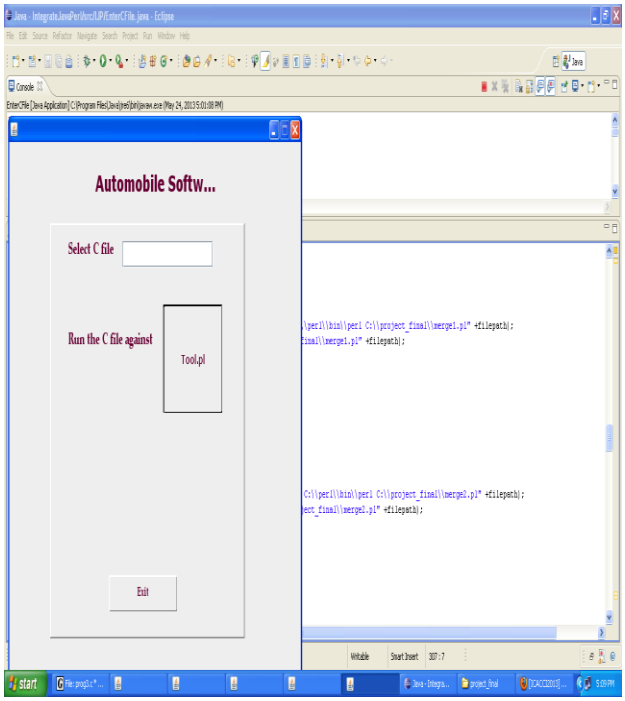

Figure 6: Browsing for the input

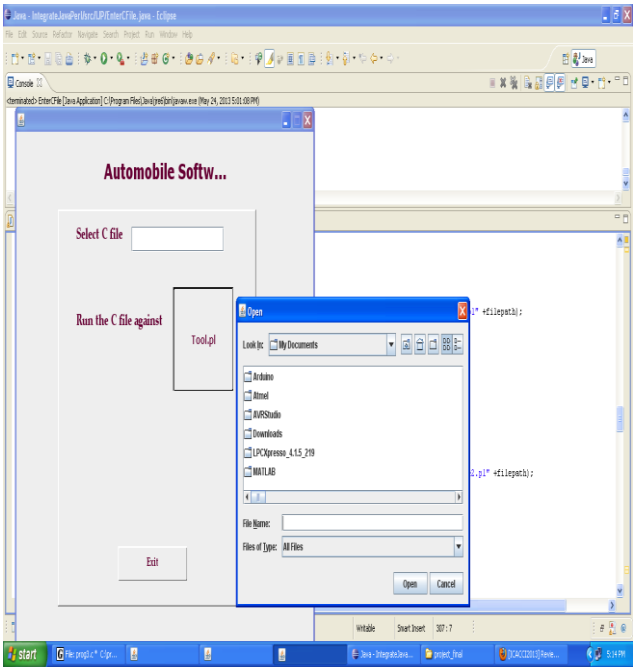

Figure 7: Selection of the c source code from the system

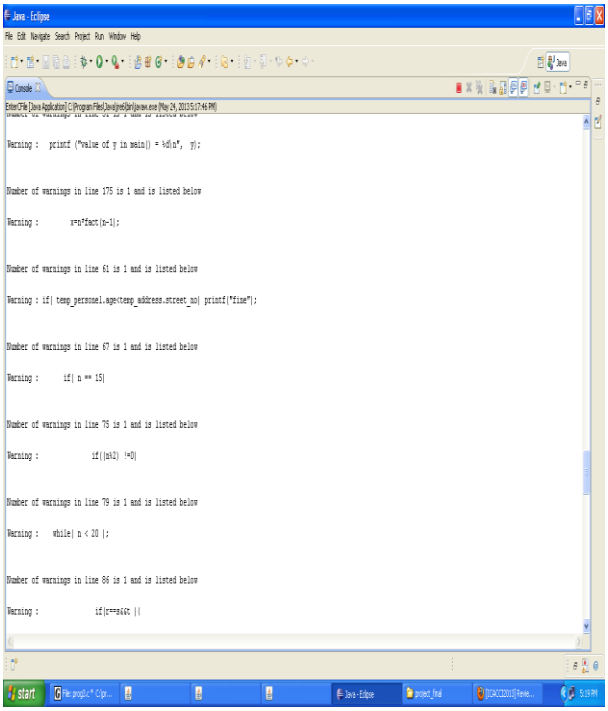

Figure 8: Screen shot which shows the violated locations

5. CONCLUSION

Recently, according to the development of computer technology, the dependency of automobile system on the computer software is being increased rapidly, and in accordance with this technical development, high level of safety and reliability in the automobile system software is required. This necessitated the development of the parser tool for software coding rules of the automotive system to secure the safety of software. This paper explained the contents of development for the automated parser tool along with the coding rules for automobile system software. Basically from the standpoint of assessor, the automated parser tool for design & coding standard for the automobile system software may be utilized to check if the software had observed coding rules required to the target software at the software assessment stage, and it can be utilized at the software development process. That is, if this automated parser tool is used in debugging process at the software development stage, the source code suitable for the coding standard can be developed, and it is anticipated that the software having higher safety level can be secured through it.

ACKNOWLEDGMENT

Our sincere thanks to Prof K N B Murthy, Principal and Prof Shylaja S S, HOD, Department of Information Science and Engineering, PESIT, Bangalore, for their constant encouragement.

REFERENCES:

[1]. IEC 62425," Railway Application: Communications, signaling and processing systems – Safety related electronic system for signaling", 2005.10.

[2]. MISRA-C Coding Standard, MISRA (Motor Industry Software Reliability Association), 2004.

[3]. IEC 62278," Railway Applications – The specification and demonstration of RAMS", 2002.

[4]. J.G Hwang, H.J.Jo, B.H.Kim, R.G.Jeong", Development of Automatic Testing for Software Coding Rules for Railway signaling", IEEE T&D Asia 2009.

[5]. About MISRA-C www.tasking.com/resources/TASKING_Misra_C_Softer_Side.pdf