Analysis of Software Reusablity Concepts Used In Automotive Software Development Using Model Based Design and Testing Tools

Dr. P. Sivakumar[*], Mrs. R.S. Sandhya Devi [**], Dr. B. Vinoth Kumar [***], Mr. R. Balaji [*], Mr. Dominic savari raj X[*]

[*] Department of EEE, PSG College of Technology, Coimbatore, India
[**] Department of EEE, Kumaraguru College of Technology Coimbatore, India
[***] Department of Information Technology, PSG College of Technology, Coimbatore, India

Abstract- Automotive electronics has gained very high significance over the past three decades as electronics in today’s cars has already exceeded 20% of the total vehicle value, and is expected to increase further in the forthcoming years. Hence is an increasing demand to reduce the automotive Electronic Control Unit (ECU) development cost, enhance reliabilty, and shorten the development cycle time. Modularity facilitates code reuse, integration of third-party components is likely to reduce the development cost by introducing new features. In this paper, MATLAB is used to generate Application layer and RTE layer which in compliance with AUTOSAR. The generated AUTOSAR compliance RTE layer in MATLAB is completely independent of hardware, so the generated RTE is portable to any different hardware platform. Therefore the reuse of SWCs in automotive system is achieved using AUTOSAR standard. Model Based Design (MBD) is a promising way to meet the above demands. Most of the future innovations in the car industry will happen in the area of electronics, where the role of software is going to be very significant. Further research work in this field facilitates automobile manufacturers to develop vehicles that satisfy new government norms that focus on specific parameters like road safety and pollution

Index Terms- Reuse, Model Based Design, Model Based Testing, AUTOSAR.

I. Introduction

Ever increasing market demand and newer technology affecting today’s competitive business environment globally. To enhance customization process several strategies such as; modular design concept, common platform, parts standardization, components commonality etc. can be adopted [1].

Automobiles in the last three to four decades have seen an unprecedented rise in use and importance of software [2]. Being lightweight is important when automobile weight is reduced then fuel consmpation is also reduced by 0.5 litres [3]. Therefore, another solution is necessary. Because software has no unit cost, the idea to reduce complexity by using standardized software suggests itself. Prominent standardized automotive software architecture is the AUTOSAR.
Model Based development under real-time constraint helps to launch new vehicles by reducing the valuable time required for testing. The Model Based methodology approach proposes the use of an architecture exploration tool to facilitate the rapid exploration of various CPUs, memories, and peripherals [16].

II. Reuseablity of Software in Automotive

Typically, functionalities changes only to a small amount from one vehicle generation to the next. Most of the old functionalities remains remain same can be found in the new car generation. Today the process of software reuse is not systematically planned between OEMs and Suppliers, as required, say, for software product lines [4].

In the end, a lot of the code could be generated from high-level models, which can be reused in product line approaches [5]. The organizational structure of the development process, with its interplay between OEMs and suppliers and the resulting conflicting desires for reuse, must also be taken into account, and could, in a second step, possibly be reshaped [6].

A. *Reuse of Software: A Challenge for Automotive OEMs*

More and more highly connected functions must be developed to series-production readiness, while at the same time development cycles become shorter and shorter. The importance of software in the automotive industry is shown in Figure 1. According to this study, in the year 2010, 13% of the production costs of a vehicle will go in software.

The main prerequisite for reusing software in the automotive domain is to separate the hardware of an ECU from the embedded software that runs on it. In order to fulfill the increased communication needs of these electronic systems, the ECUs communicate via different bus systems [7]. In requirements for the development of safety critical functions are stated for example, verification process, time-triggered architectures, and software architecture models [8].

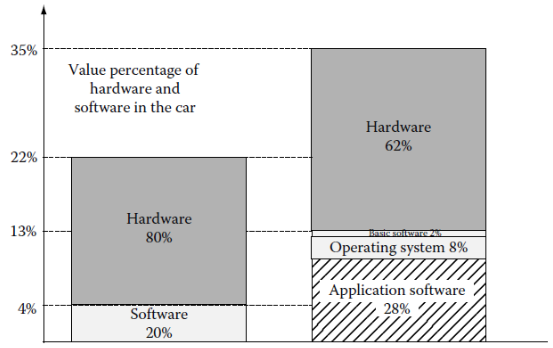

Figure.1 Rise of importance of software in the car

The increasing complexity of automotive electronics, the platform, distributed real-time embedded software, and the need for continuous evolution from one generation to the next has necessitated highly productive design approaches [9].

One of the most valuable aspects of Model-Based Design is the availability of executable models to perform Verification, Validation, and Test (VV&T) throughout the development process, especially its earliest stages [17].

III. MODEL BASED SOFTWARE DEVELOPMENT PROCESS FOR AUTOMTOIVE SOFTWARE SYSTEMS

Using Model Based Design the overall of cost- and time-consumption is heavily reduced because errors can be detected and corrected in early design phases [11]. Today, more than 3000 functions are realised by software running on ECUs, intelligent sensors and actuators [12] [13].

The reason for building those intermediate levels is the fact that it is much cheaper and faster to modify a model than to change the final product. The entire process is called MBD [14]. Additionally, methods such as Correct-by-Construction (CbyC) exist [15]. This method enables automatic code generation and ensures that the parameters that are verified on the model level can also be verified on the embedded code level [32].

MBD can be defined as a visual way to create complex control systems. The main target is to move a designer's focus from code programming to system design. An example on how to define it as a process is the V-model form, shown in Figure. 2 [31].

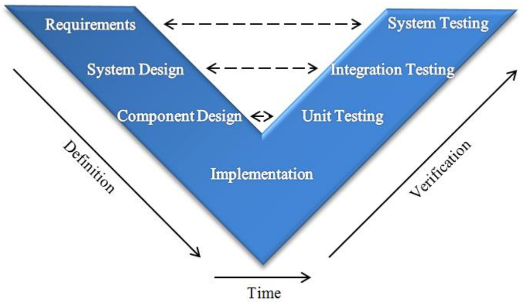

Figure.2 A V-model of the Model Based Design process

The big challenge for today's engineers is to manage complex design requirements. This is why the system design phase is often divided into two parts. The first part is to create an overall architecture consisting all the input, output and inner parameters of the designed system. The second part is to design the individual components. Dividing the system design into smaller subsystems gives an ability to divide the workload amongst a design team [18].

IV. MODEL BASED TESTING APPROACH FOR SOFTWARE DEVELOPMENT IN AUTOMOTIVE

Here comes the most crucial step for yourresearch publication. Ensure the drafted paper is critically reviewed by your peers or any subject matter experts. Always try to get maximum review comments even if you are well confident about your paper.

Model based Testing (MBT) aids to link the process of creating test oracles [19]. MBT relates to a process of test generation from models related to a SUT by applying a number of sophisticated methods. MBT is defined as testing in which test cases are derived in their entirety or in part from a model that describes some aspects of the SUT based on selected criteria [19]. In the automotive industry, MBT describes all testing activities in the context of MBD.

Commercial tools such as Conformiq, Reactis have been developed to support MBT which can be used as an integrated part of the testing process [20] [33]. Currently the challenges present in automotive software development are to meet the brand survival demands, competition, quicker time to market, increased complexity advanced functionality, tight performance constraints and high reliability demands [21].

A. *Reuablity & Implementation of Reusablity through AUTOSAR*

The applications are decoupled from the hardware as shown in Figure.3. With a standardized interface it is possible to buy software and hardware from different manufacturers, and all work together [24].

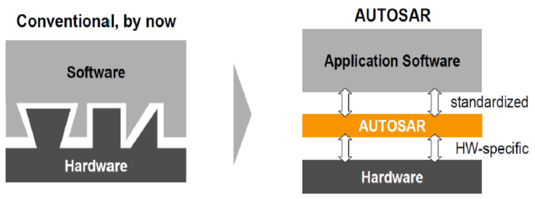

Figure. 3 Application software independent from the hardware using AUTOSAR

The complexity of automotive software systems continues to increase [21]. Modularity facilitates code reuse and integration of third-party components which is likely to reduce the development cost by introducing new features. ISO-26262 requires robust testing of the software units of a system [23] [31] [32]. The industry requires the need for tools to perform the robust testing of AUTOSAR components [10].

V. Reuse of Automotive Software

The MathWorks approach based upon MATLAB®, Simulink, and Real-Time Workshop Embedded Coder for generating AUTOSAR-compliant code follows a transparent and intuitive process [25] [32].

In the top-down workflow use an architecture design tool to design the architecture ECU network then export an XML-description of the corresponding components: the AUTOSAR Software Component Description. AUTOSAR System Target in Real-Time Workshop Embedded Coder needs to be selected [26]. The bottom-up workflow requires the same AUTOSAR configurations as described in the top-down workflow.

A. *Software Environments for Automotive System*

OSEK was developed to provide standard Real-Time OS and software architecture for various automotive ECUs [27]. The application software can be described in the form of a hardware independent model; advantage of this approach is the easy migration of proven function models from one vehicle to the next.

B. AUTOSAR Architecture

AUTOSAR is a development partnership of Automobile manufacturer's world wide, companies from the electronics, semiconductor, suppliers and software industry formed in 2003. This partnership has developed an industry wide standard for the automobile electronics, which is headed by the core partners – BMW Group, Bosch, and Continental [22].

In AUTOSAR, the ECU software is sub-classified as application layer, software BSW layer, and run time environment RTE. The layer of the architecture is shown in Figure.4.

The ECU Abstraction Layer offers consistent access to all features of an ECU like communication, memory or I/O. The drivers for such outside peripheral components reside in this layer.

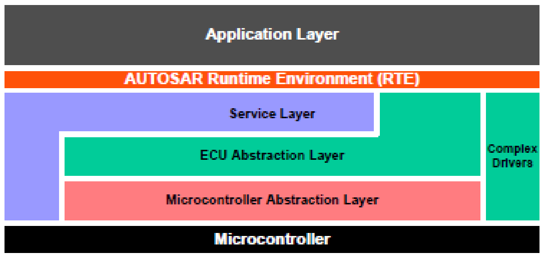

Figure. 4 Different layers of AUTOSAR

The Service Layer provides various types of background services such as vehicle network communication and memory management, diagnostic services, management services, ECU state management [35].

The RTE implements the data exchange and controls the integration between the application software section and BSW [35].

The AUTOSAR Software Component SWCs is a fundamental device concept of AUTOSAR, which is the fundamental structure of an AUTOSAR application Each SWCs is deployed, or mapped, on one ECU. For example, an automatic light adaptive application may consist of three AUTOSAR SWCs which are outside brightness detective component, a light request component and a light control master component [28]. The drivers do not follow the normal layered AUTOSAR architecture, but instead access both the microcontroller and RTE layers directly [29]. Detailed Comparsion metric with common software architecture based on different paramets as shown in Table1.

Table 1. Quality Attribute table for AUTOSAR.

Quality Attributes	**Architecture Metric**	**AUTOSAR / Common Architeture**	**Existing Architecture**
Complexity	Software component Complexity	1.688	2.164
Portablity	Dependency on Basic Software	0.249	0.3579
Testablity	Average Input Interface Complexity	1.123	3.183
Maintanablity	Average output Interface Complexity	0.734	2.367

From the table infered that, there is clear statisical differentiation between Existing Architecture and the Common Software Architecture [34]. Fig. 5 Shows the Statistical comparision of Existing Architecture and the Common Software Architecture.

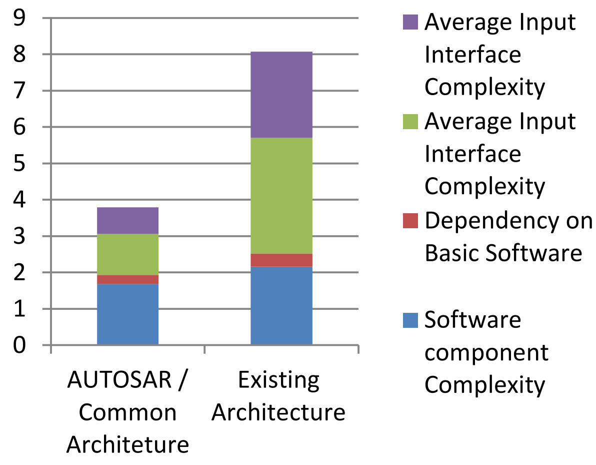

Figure.5 Statistical comparision of Architecture Metric

VI. Kernel Development Process

The AUTOSAR porting procedure on Raspberry Pi is described [36] [31]. Kernel devlopment of AUTOSAR OS consider this Raspberry Pi board as Hardware for the kernel development. This kernel expansion process includes initialization, exception handling, memory modeling and context switch. Initialization is the primary code that is executed when the OS starts up.

A Raspberry Pi's boot process starts with a small ROM based primary boot loader copying a file containing a secondary boot loader from SDHC card into the L2 cache of the microcontroller. The boot loaders compute SDRAM and load a third boot loader into SDRAM which starts up the OS [30]. The bootloader of Raspberry Pi consist of bootcode.bin, loader.bin and start.elf. It boots at the ARM address space from the 0x00008000, which is the ARM entry point and location of the first ARM instruction.

A. Cross Compilation Results

After the completion of cross compiling a kernel image file along with bootloader files are generated. This kernel image corresponds to AUTOSAR OS. Kernel.img is the heart of the AUTOSAR OS which is be used for data communication through CAN protocol and also for sensor elements. Bootloader.bin and start.elf files are necessary for booting ARM processor from 0x00008000 of ARM address space and that is the ARM opening point and location for the initial ARM instruction.

B. Reusability of Software Components

In order to provide these reusabilty functionalities, a set of 6 SWCs is defined as shown in Figure.6 showing the SteerSensor and WheelSensor sensor SWCs; the SteerManager and WheelManager application SWCs; and the SteerActuator and WheelActuator actuator SWCs. The type of the ports used is sender-receiver. Data is acquired at the sensor SWCs level, transformed and then yielded to the 2 application SWCs. Data is also exchanged between application SWCs. Finally the data is sent to the actuator SWCs from the application SWCs. Each SWCs is composed of one or more Runnables.

The SteerManager SWCs is composed of 3 Runnables: Today car manufacturers produce large number of Customized Cars to meet customer demands. The objectives are to increase the ability to reuse software and to meet customization requirements.

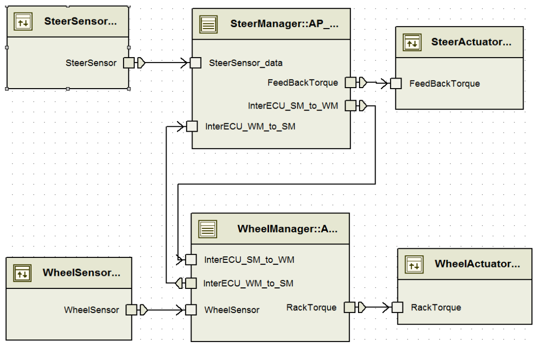

Figure.6 Typical steer-by-wire systems

This system contains two Actuator SWCs for the two direction indicators. The system also contains two Sensor SWCs to read the state of the two switches. The Direction Indicator Switch component sends the state of the switch to a service, while the Warning Light Switch component uses a Sender-Receiver communication to send the state of the switch to another component. The communication between the components is designed only virtual by using the VFB.

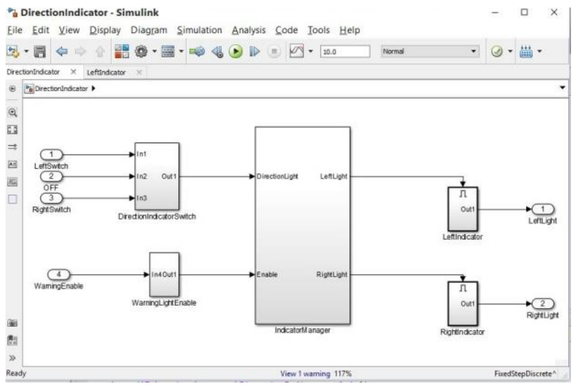

Figure.7 Design of a car direction indicator system

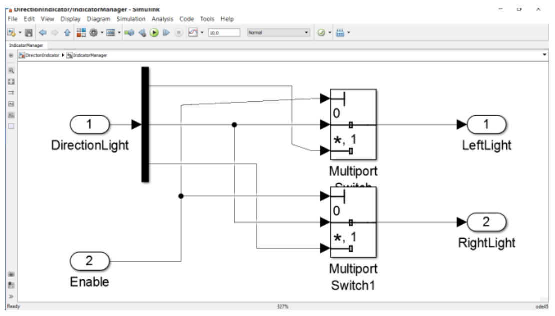

Fig.8 Internal component of indicator manger

The model is developed using Simulink block as shown in Figure.7. DirectionIndicatorSwitch to process the signal from the switch is used to select the direction Light to turn ON by means of input switch. Light blinks only if WarningLightEnable is enabled. IndicatorManager is used to issue the signal to blink the Indicator based on the input. Left/Right Indicator contains pulse generator and based on the logic issued to the manager it blinks the light. The internal component of indicator manager is shown in Figure.8. Baesd on the logic issued by Manager blinks the indicator.

C. *Integration*

MATLAB generates Application layer and RTE layer with AUTOSAR compliance code. Driver files for Raspberry Pi are readily available. RTE interface should be manually written, which interfaces the MATLAB generated RTE and Driver file. Finally integrate the RTE which is generated by MATLAB, for Raspberry Pi board to form a kernel.img file. Copy that kernel.img into the SD card and insert into the Raspberry Pi board and turn it ON. Creating more than two runnables i.e. tasks and integrating these runnables with OS service i.e. Real – Time OS is also possiable. The MATLAB generated RTE i.e. runnables and OSEK OS are integrated with the Hardware driver file and then converted it into kernel image. The kernel image is then loaded into SD card of the Rraspberry pi.

AUTOSAR uses its own notation for modeling applications. An application consists of one or more SWCs based in the Application layer. In order for SWCs to communicate with each other the Virtual Functional Bus (VFB) is used from SWC's point of view all it sees is the VFB and not the hardware dependent BSW and the hardware itself; in reality this is provided by the RTE.

In this paper, resuse of automotive software component is implemented using hardware and software tools it follows AUTOSAR-Compliant Code, AUTOSAR architecture it support both MBD and MBT development.

VII. CONCLUSION

One of the advantages of modularity is the ability to focus specialized engineering talents that results a higher quality product or services. Modularization can be applied as a framework that splits the activities of end products among OEMs and suppliers. It is a concept of design and develops smaller sub-systems independently which are able to function properly after assembled and tested with an endproduct. In auto industries, expanded use of modularity concept could create ample opportunities for the production of customized vehicles. In shorter term, modularity may not be very much effective but in longer run this has been proven to be beneficial for industries. The paper deals with the reuse of components, or subsystems, which is one way to achieve time to market. Thus AUTOSAR is developed for Raspberry Pi. Therefore AUTOSAR act as a platform for different embedded systems to speed up the development process and also provide an open hardware platform for experimentation on advanced automotive ECUs. In this paper, AUTOSAR compliance OS has been developed and the generated AUTOSAR compliance RTE layer using MATLAB, is completely independent of the hardware, which makes it portable to any different hardware platform. Therefore the reuse of SWCs in automotive system is achieved using AUTOSAR standard. The generated RTE is completely independent of hardware however the driver file depends on the specific hardware used in the design.

REFERENCES

[1] Liggesmeyer, P & Trapp, M, “ Trends in embedded software engineering,” *IEEE software*, vol. 26, no. 3.2009.

[2] Schlegel, J, “Technical foundations for the development of automotive embedded systems,” *Technical report, Hasso-Plattner-Institut für Softwaresystemtechnik GmbH*. 2009

[3] Voget, S & Becker, M, “Establishing a software product line in an immature domain,” *In International Conference on Software Product Lines*, Springer Berlin Heidelberg, 2002, pp. 60-67.

[4] Ahluwalia, J, Krüger, IH, Phillips, W & Meisinger, M, “Model-based run-time monitoring of end-to end deadlines,” *In Proceedings of the 5th ACM international conference on Embedded software*, 2005, pp. 100-109.

[5] Dannenberg, J, & Kleinhans, C, “The coming age of collaboration in the automotive industry,” *Mercer Management Journal*, vol. 17, no. 66, 2004, pp. 88-94.

[6] Krüger, A, Hardung, B & Kölzow, T, “Reuse of software in automotive electronics,” *In Automotive Embedded Systems Handbook*, CRC Press , 2008, pp. 214-233.

[7] Simonot-Lion, F, “In car embedded electronic architectures: How to ensure their safety,” *In 5th IFAC International Conference on Fieldbus Systems and their Applications*-FeT, 2003, pp. 1-8.

[8] Broy, M, Kirstan, S, Krcmar, H, Schätz, B & Zimmermann, J, “What is the benefit of a model-based design of embedded software systems in the car industry?,” *Software Design and Development: Concepts, Methodologies, Tools, and Applications*, IGI Global,USA, 2013, pp. 310-334.

[9] Yu, H, Joshi, P, Talpin, JP, Shukla, S & Shiraishi, S, “The challenge of interoperability: Model-

based integration for automotive control software," *In Proceedings of the 52nd Annual Design Automation Conference*, ACM, Article 58, 2015, pp. 58:1- 58:6.

[10] Mitschang, J, "Evaluation of a Model-Based Development Process for Automotive Embedded Systems," *Doctoral dissertation, Kaiserslautern, Technische Universität Kaiserslautern, Diplomarbeit*, 2009.

[11] Kugele, S, "Model-Based Development of Software-intensive Automotive Systems," *Doctoral dissertation, München, Technische Universität München, Diss*., 2012

[12] Ebert, C, & Jones, C, "Embedded software: Facts, figures, and future," *Computer*, vol. 4, no. 4, 2009, pp. 42-52

[13] Yoon, H, "A Multi-layered Statechart Diagram Including Mitigation Behavior*," International Journal of Computer Science and Network Security (IJCSNS)*, vol.15, no. 4, 2015, pp. 44-47.

[14] Baleani, M, Ferrari, A, Mangeruca, L, Sangiovanni-Vincentelli, AL, Freund, U, Schlenker, E, & Wolff, HJ, "Correct-by-construction transformations across design environments for model-based embedded software development," *In IEEE Design, Automation and Test in Europe, Proceedings*, 2005, pp. 1044-1049.

[15] Conrad, M, "Verification and validation according to ISO 26262: A workflow to facilitate the development of high-integrity software," *Embedded Real Time Software and Systems. MathWorks*, Inc. Natick, MA, USA, 2012.

[16] Lumpkin, E & Gabrick, M, "Hardware/software design and development process," (No. 2006-01-0170). *SAE Technical Paper*, 2006

[17] Murphy, B, Wakefield, A, & Friedman, J, "Best practices for verification, validation, and test in model-based design," (No. 2008-01-1469). *SAE Technical Paper*.2008.

[18] Skruch, P & Buchala, G, "Model-Based Real-Time Testing of Embedded Automotive Systems*," SAE International Journal of Passenger Cars-Electronic and Electrical Systems*, vol. 7, no. 2, 2014, pp. 337-344.

[19] Bringmann, E & Kramer, A, "Model based testing of Automotive Systems," *International Conference on Software Testing, Verification and Validation*, 2008, pp. 485-493.

[20] Broy, M, "Challenges in automotive software engineering," *Proceedings of the 28th international conference on Software engineering*, ACM, 2008, pp. 33-42.

[21] Marinescu, R, Saadatmand, M, Bucaioni, A, Seceleanu, C, & Pettersson, P, "A model-based testing framework for automotive embedded systems," *In Software Engineering and Advanced Applications (SEAA)*, 40th EUROMICRO Conference. 2014, pp. 38-47.

[22] Mössinger, J, "Software in Automotive Systems," *IEEE Software*, vol. 27, no. 2, 2010, pp. 92-94.

[23] Melin, J & Bostrom, D, "Applying AUTOSAR in Practice Available Development Tools and Migration Paths," *Master Thesis, Computer Science,* School of Innovation Design and Engineering, Mälardalen University., 2010.

[24] Köhler, A & Volkswagen, AG, "AUTOSAR-Compliant Functional Modeling with MATLAB®, Simulink®, Stateflow® and Real-Time Workshop® Embedded Coder of a Serial Comfort Body

Controller," *In MathWorks Automotive Conference*, 2007.

[25] Sandmann, G & Thompson, R, ":Development of AUTOSAR software components within model-based design," (No. 2008-01-0383). *SAE Technical Paper*. 2008.

[26] Cho, J, "Software Development Environment for Automotive SoC. In Algorithm & SoC Design for Automotive Vision Systems," *Springer Netherlands*, 2014, pp. 231-261.

[27] Park, G, Ku, D, Lee, S, Won, WJ, & Jung, W, "Test methods of the AUTOSAR application software components," *In IEEE ICCAS-SICE,* 2009, pp. 2601-2606.

[28] Wu, R, Li, H, Yao, M, Wang, J & Yang, Y, "A hierarchical modeling method for AUTOSAR software components," *In IEEE Computer Engineering and Technology (ICCET), 2nd International Conference*, vol. 4, 2010, pp. V4-144 –V4-184.

[29] Bo, H, Hui, D, Dafang, W & Guifan, Z, "Basic concepts on AUTOSAR development," *In IEEE Intelligent Computation Technology and Automation (ICICTA), International Conference,* vol. 1, 2010, pp. 871-873.

[30] Voget, S & Becker, M, "Establishing a software product line in an immature domain," *In International Conference on Software Product Lines*, Springer Berlin Heidelberg, 2002, pp. 60-67.

[31] Sivakumar, P. and Vinod, B. and Devi, R.S. Sandhya and Divya, R, "Novelty Testing Measures and Defect Management in Automotive Software Development," *Australian Journal of Basic and Applied Sciences*, 2016, Vol. 10(1), Pages: 607-613.

[32] Sivakumar, P , Vinod, B. , Sandhya Devi, R. and S.Poorani Nithilavallee, "Model Based Design Approach In Automotive Software and Systems," *International Journal of Applied Engineering Research*, 2016, ISSN 0973-4562 Volume 10, Number 11,pp. 29857-29865.

[33] Sivakumar, P, Vinod, B, Sandhys Devi, R. and Divya, R, "Deployment of Effective Testing Methodology in Automotive Software Development," *Circuits and Systems,* 2016, 7, 2568-2577. doi: 10.4236/cs.2016.79222

[34] Sreekanth, A., Srikanth, K., Aditya, C., Satish, T. and Ramchandran, R, " Deploying common software system for hybrid electric vehicles in AUTOSAR way," *In Transportation Electrification Conference (ITEC-India)*, 2017 IEEE, pp. 1-6.

[35] Sandhya Devi R.S., Sivakumar P., Balaji R, "AUTOSAR Architecture Based Kernel Development for Automotive Application," *In: Hemanth J., Fernando X., Lafata P., Baig Z. (eds) International Conference on Intelligent Data Communication Technologies and Internet of Things (ICICI)* 2018. Lecture Notes on Data Engineering and Communications Technologies, vol 26. Springer, Cham.

[36] M. M. Srihari and P. Sivakumar, "Implementation of Multi-Function Printer for Professional Institutions," *International Conference on Inventive Research in Computing Applications* (ICIRCA), Coimbatore, 2018, pp. 1-3.doi: 10.1109/ICIRCA.2018.8597283