iJRASET
International Journal For Research in Applied Science and Engineering Technology

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Volume: 13 **Issue:** IV **Month of publication:** April 2025

DOI: https://doi.org/10.22214/ijraset.2025.69213

www.ijraset.com

Call: 08813907089 | E-mail ID: ijraset@gmail.com

International Journal for Research in Applied Science & Engineering Technology (IJRASET)
ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538
Volume 13 Issue IV Apr 2025- Available at www.ijraset.com

Components for Implementing Tool Interface Requirements in Automotive Domain

Anand Wanjari
USA

Abstract: ***The modern automotive ecosystem demands seamless interaction between electronic control units (ECUs) and diagnostic tools. Tool interface requirements play a pivotal role in enabling effective diagnostics, repair, and maintenance of vehicle systems. This paper presents a comprehensive view of the environment required for implementing tool interface requirements for diagnostic tools. It covers essential components including the Application Controls Systems Team, Tool Integration Strategies, Global Tool Interface Standard (GTIS) Specification, validation by the Central Tools Validation team, and the final implementation and testing processes. The paper highlights each component's function, discusses integration strategies, and evaluates the pros and cons of this structured development environment.***

Keywords: Electronic Control Modules (ECU), Global Tool Interface Standard (GTIS), Unified Diagnostic System (UDS), Original Equipment Manufacturer (OEM), Failure Modes and Effect Analysis (FMEA), Functional Safety (FuSa), Tool Integration, Requirement Analysis, Diagnostic Tools.

I. INTRODUCTION

With increasing complexity in vehicle control systems, the ability to diagnose, configure, and service vehicles efficiently has become critical. Diagnostic tools, serving as the bridge between service personnel and onboard ECUs (Electronic Control Units), must adhere to robust tool interface requirements to ensure reliability and standardization. The development of these interfaces involves cross-functional collaboration, stringent guidelines, and a structured workflow. This paper explores each facet of this environment, showcasing how standardization and modular strategies contribute to scalable and efficient diagnostic solutions. The development and production of modern vehicles are affected by a strong relationship between OEM (original equipment manufacturer) and its suppliers. Due to the increasing importance of software in the automotive industry, both OEMs and suppliers have established a requirement engineering process to stay abreast of changes. The OEM's requirements engineering process is tailored to the management and the integration of certain supplier products. Therefore, the OEM's goal is to provide information about the system environment into which the supplier product must be integrated **[9]**

II. BACKGROUND

Automotive diagnostics rely heavily on precise communication protocols and tools capable of interacting with vehicle subsystems. Traditionally, each vehicle model or subsystem may have had proprietary diagnostic approaches, but the demand for standardized tool interfaces led to the establishment of structured environments and specification guidelines such as GTIS. A centralized workflow involving control systems engineers, integration teams, validation bodies, and service tool developers ensures that diagnostic interfaces are consistent, maintainable, and production ready. While implementing the tool interfaces we need to consider the following quality attributes Rendering Scalability: Tools must handle varying data sizes efficiently, especially for interactive systems. Information Scalability: Tools should manage cognitive overload and allow selective information display. Interoperability: Tools need mechanisms for seamless integration and data exchange among different systems. Customizability: Tools should allow user-specific configurations and extensions for diverse contexts. Interactivity: Tools must enable user manipulation while balancing automation to avoid excessive interactions. Usability: Tools should be easy to use, with intuitive interfaces and minimal cognitive overhead. Adoptability: Tools must integrate smoothly into existing processes to encourage user adoption **[1]**

III. COMPONENTS EXPLAINATION

A. Controls System Team

The Controls Team consists of control system engineers responsible for translating functional service requirements into software behavior. They typically use Matlab Simulink models to design the embedded control logic, including diagnostics.

Their input serves as the foundation for tool interface development, defining what information and capabilities the service engineers will need from the diagnostic tools. The output created by the control systems team would be the technical profile document. This document consists of the functional requirements based on the wiring diagram of the product. Functional requirements consist of the range and operating environment of the physical entity that need to be implemented in the system. This technical profile also should include the calibratable parameters, trimmable/adjustable parameters and override parameters. Specially overrides parameters are defined for simulation/testing purpose only. FMEA and System functional requirement flow diagrams specially for the diagnostic tests are valuable outputs we get in this process.

The initial requirements come from OEM, Tier 1 or Tier 2 suppliers depending upon the product implementation/integration. Control systems engineer ensures the new component to align with vehicle architecture, auto domain regulations/compliance and performance criteria. The developed technical profile document includes a high-level boundary diagram which defines the scope of the component in the system, Functional and non-functional requirements, communication protocol requirements, Error handling, Performance requirements. If the component falls under functional safety, then ISO26262 related requirements shall be included. ISO 26262 prescribes a systems engineering process for safety engineering. It recognizes that safety is a system attribute and can be addressed using a systems engineering approach. It also emphasizes the importance of fostering a safety culture and implementing safety engineering management **[2]** It is essential to perform the FMEA activity as soon as the system and component requirements are defined. FMEA provides an easy tool to determine which risk has the greatest concern and therefore an action is needed to prevent a problem before it arises.**[6]** The inputs from FMEA helps design the safety goals and ASIL categorization. FMEA is an iterative process and not a onetime activity which ensures the changes in requirements are captured, new failure modes are identified. FMEA are conducted in the product design or process development stages, although conducting an FMEA on existing products or processes may also yield benefits. The FMEA team determines, by failure mode analysis, theeffect of each failure and identifies single failure points that are crucial. It may also rank each failure according to the criticality of a failure effect and its probability of occurring **[3]**

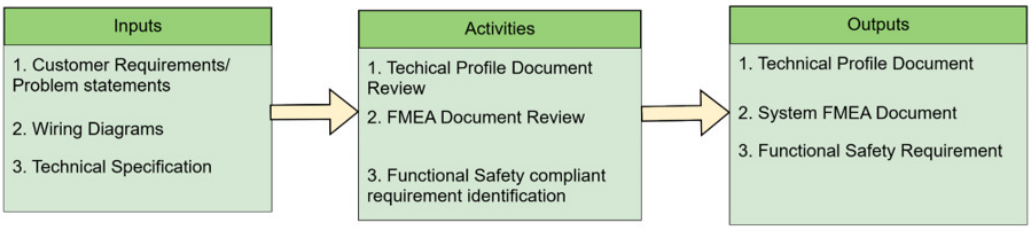


Figure 1 - Controls system team requirement workflow

B. Tool Integration Strategies

Tool integration is not a property of a single tool, but of its relationships with other elements in the environment, chiefly other tools, a platform, and a process. Tool integration is about the extent to which tools agree. The subject of these agreements may include data format, user-interface conventions, use of common functions, or other aspects of tool construction. A framework that determines how well tools are integrated into an environment and that defines integration independently of the mechanisms and approaches used to support integration is proposed. Process, data, control, and presentation integration properties are described separately to identify them as clearly and independently as possible **[4].** Tool Integration is a crucial step in developing tool interface requirements. This component defines how requirements are captured, structured, and translated into a standardized format for further processing. The input requirement for tool integrator is the technical profile document created by the control systems team. It is essential to get approval on I/P concerns before creating any tool interface parameters which will be available to users. Tool integrators work as a bridge between controls team and tools team as they have understanding of the controlsstrategies, and they have better understanding of the diagnostic tools functionalities. Following key strategies helps Tool integrator to simplify the requirements:

B.1) Requirement Templates: Customized templates help in structuring diagnostic service requirements, ensuring consistency and traceability. These templates are used while gathering the requirements from application controls and service teams.
B.2) Version Control Mechanisms: Track changes in requirements across development cycles.
B.3) Traceability Mapping: Linking tool interface requirements back to control model elements or service engineer use cases.

Integration strategies aim to bridge the gap between control logic design and tool implementation, reducing ambiguity and enhancing reusability. In the tool integration process, we need to assign unique IDs to all the requirements/failures defined under technical profile document, FMEA document. These activities are performed based on the guidelines under the GTIS document. The outputs of the integration are traceability matrix, impact analysis report, audit trails, verification planning document. The tools like IBM DOORS, Polarian, Integrity are used to write and maintain the version-controlled requirements **[10].** As shown in figure below the output we get by following the tool integration activities are the structured tool interface templates which are a set of predefined questionnaires having key details on implementing the tool interfaces. Also, the failure mode use cases which are considered to design the failsafe functional requirements.

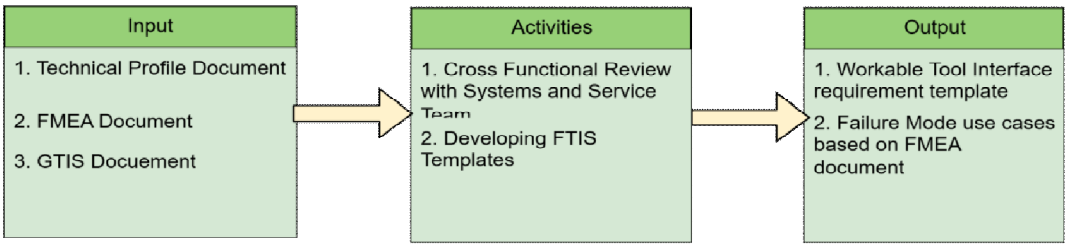


Figure 2: Tool Integration team requirement workflow

C. Global tool interface standard (gtis) specification

GTIS is a Tool - ECM interface comprehensive requirement document to perform Diagnostic functionalities over the automotive communication protocols like UDS (ISO 14229-1 standard), XCPetc. The UDS protocol provides a number of necessary functionalities for repairers, developers and testers so that they can, for example, read or write data in ECU memory, program the flash memory and create specific behavior for an ECU such as provide a response on a timer interrupt event. As a result, the UDS protocol is very capable but also comprehensive and therefore complex in terms of implementing all of the ISO standard's protocol functionalities **[7].** This document standardizes the structure, semantics, and formatting of tool interface requirements. Typically, versions of these documents are provided to suppliers to align them with OEM standards. This document helps define the integration boundaries. This is critical for system, integration and functional testing. It helps to clarify interface communication errors during implementation.By following GTIS, teams can produce diagnostic interfaces that are consistent across vehicle lines and tools, enabling global reuse and reducing validation overhead.

D. Central Tool Interface Validation

Risks steadily increase, as we see with the current fast growth of cybersecurity threats and insufficient usability. Thus, our verification and validation methods must evolve even faster. At the same time, we need to prepare on a dual track for failure. Quality strategies thus not only mean finding defects but also hardening systems to make them robust. Graceful degradation and fail-operational scenarios need to be designed and deployed **[8]** the central tool interface validation team serves as a centralized validation body that reviews tool interface requirements based on GTIS guidelines and latest industry trends to make the tool interface modular, reusable and failsafe. Their role includes:

D.1) Compliance Checks: Ensuring adherence to GTIS formatting and structure.

D.2) Template Creation: Providing standard templates which can be used across all the platforms for requirement detailing and maintenance

D.3) Feasibility Analysis: Assessing whether the described requirements are implementable.

D.4) Feedback Loops: Collaborating with integration teams to refine and finalize requirements. This validation step mitigates risks related to misinterpretation or incompatibility during tool development.

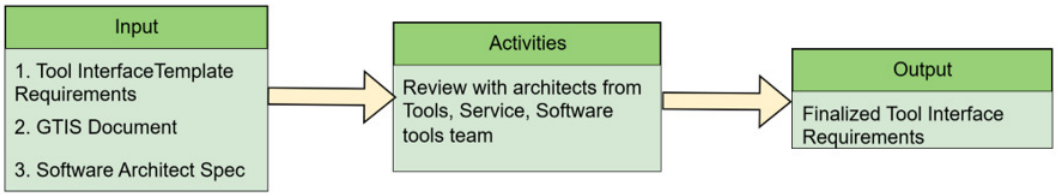


Figure 3 - Central Team Requirement Workflow

E. Tool Interface Implementation And Testing

Once validated, the tool interface requirements move to the **service tools domain**, involving:

E.1) Service Tool Development Teams: Implement the tool features using validated requirements.

E.2) Systems and Testing Teams: Ensure that the implementation meets expected performance, functionality, and reliability standards.

E.3) End-to-End Testing: Including both simulated environments and physical ECUs to confirm real-world behavior.

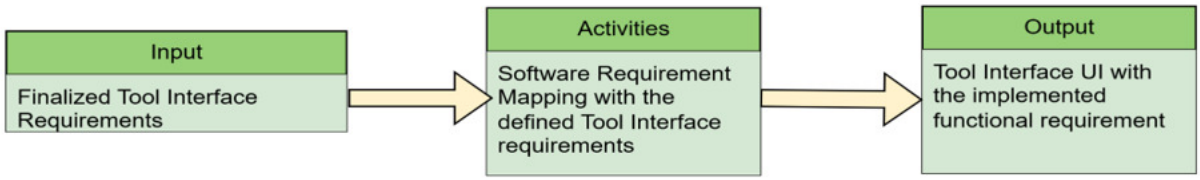


Figure 4 - Development Team Requirement Workflow

IV. COMPONENTS INTERACTION AND OUTPUTS

The development of a diagnostic tool interface in the automotive domain begins with the creation of the GTIS by system architects from the Controls, Software, and Tools teams, based on standards like UDS, XCP, ISO 26262, and FMEA. When a new business use case or feature requirement arises, it is analyzed by the Application Controls and Service teams, who define the technical profile and system-level requirements. These are passed to the Tool Integration team to develop system and functional requirement specifications, which are maintained with proper version control and traceability. The Tool Interface Validation team then reviews the requirements to ensure compliance with GTIS and tool adaptability before the finalized requirements are implemented by the Tool Development team. GTIS serves as a central reference throughout the process, ensuring consistency, integration, and compliance across all development activities.

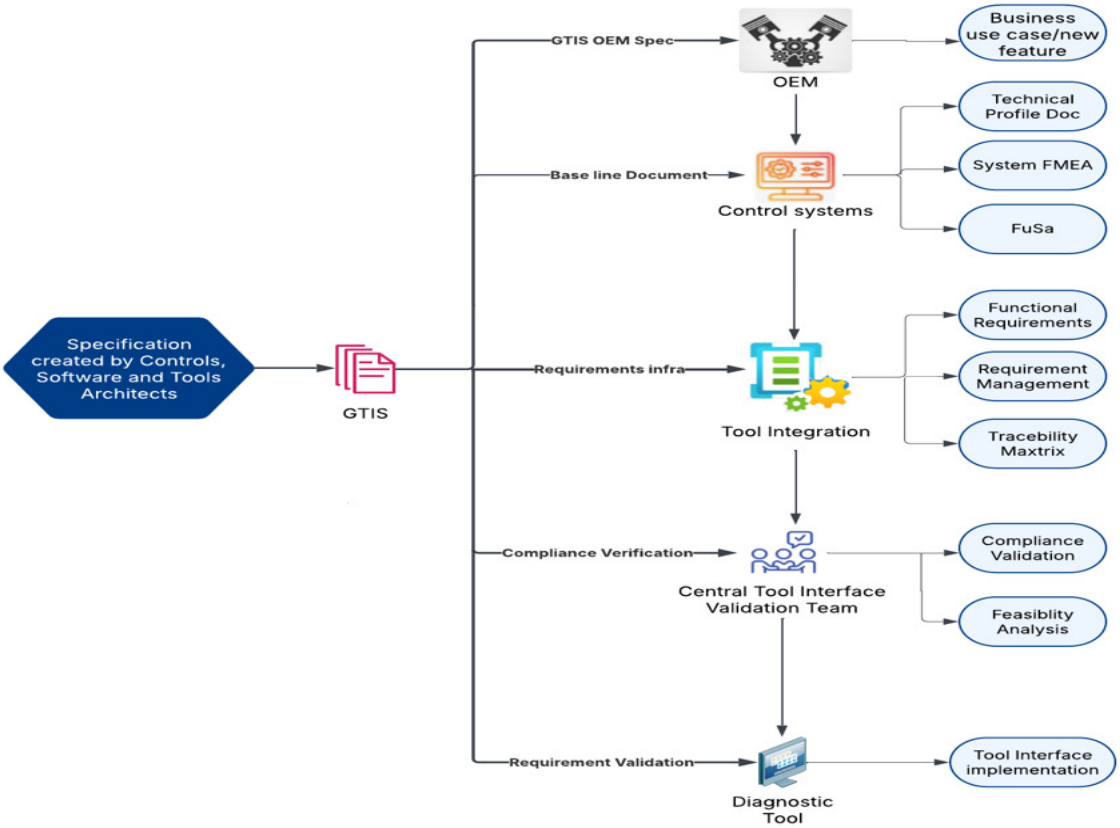


Figure 5 -Components Interaction and Outputs

V. PROS AND CONS

A. *Pros*

Benefit	Description
Standardization	GTIS ensures consistent interfaces across systems, making it easier for OEMs and suppliers to integrate their products.
Cross-Team Alignment	Provides a common reference for Controls, Software, and Tools teams, reducing miscommunication and rework.
Traceability	Enables full traceability from requirements to implementation and testing, which is critical for audits and compliance
Reusability	Promotes reuse of interface specifications across vehicle platforms, saving time and cost in future projects.
Compliance Ready	Ensures that diagnostic implementations align with industry protocols like UDS and standards such as FMEA and ISO 26262.
Early Issue Detection	Interfaces are clearly defined early, reducing integration issues and allowing for better planning of validation and diagnostics.

B. *Cons*

Limitation	Description
High Initial Effort	Developing and maintaining GTIS documents requires significant upfront effort and coordination across teams.
Complex Change Management	Updates to GTIS may impact multiple systems and suppliers, requiring careful impact analysis and stakeholder alignment.
Learning Curve	New team members or suppliers may face a steep learning curve understanding the GTIS framework and associated standards.
Tool Dependency	Effective implementation relies on mature requirement management and integration tools (e.g., DOORS, Polarian), which may not always be readily available or fully adopted.
Overhead for Small Changes	Even minor updates might require formal review cycles, making the process less agile for rapid iterations.

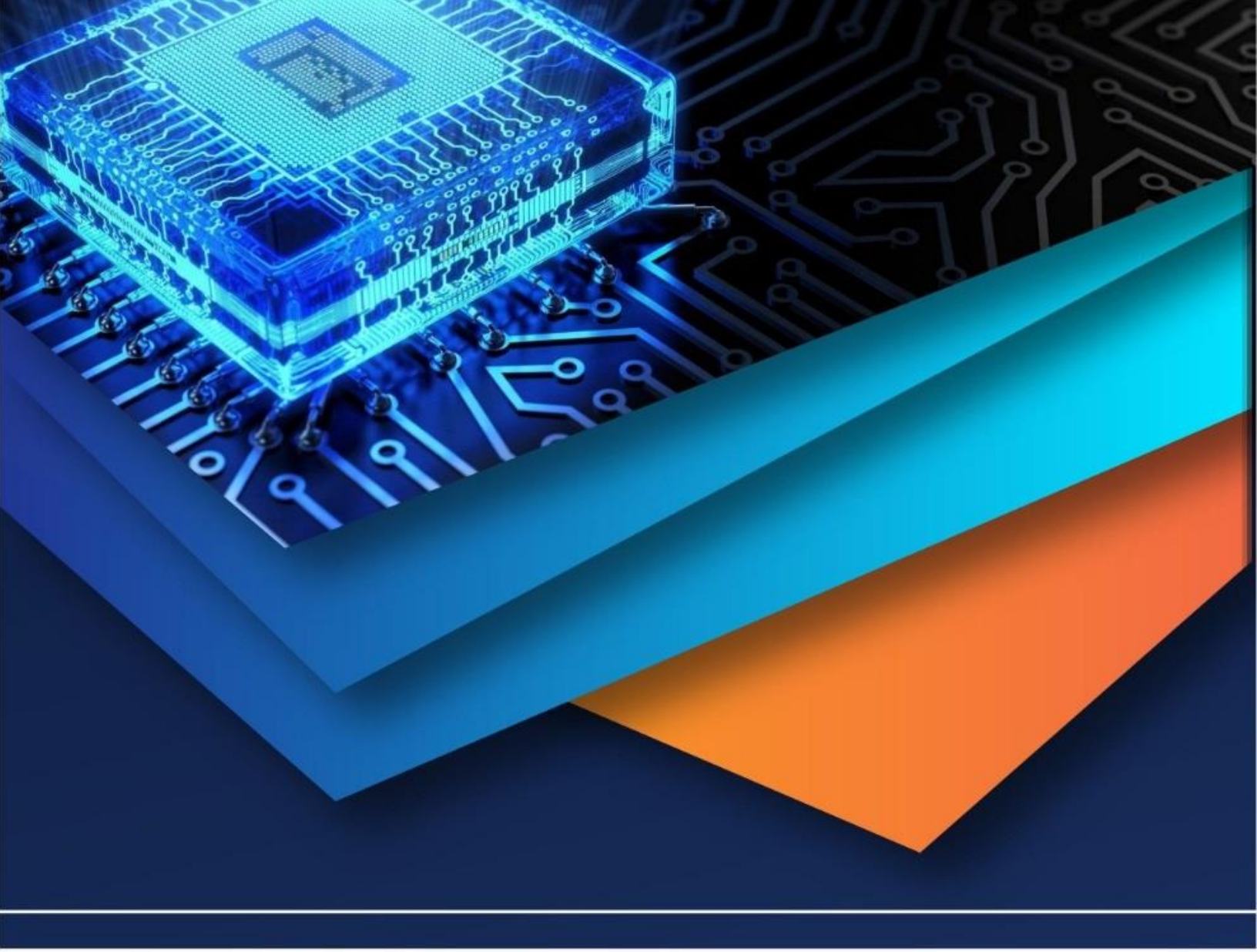
VI. .CONCLUSION

A robust environment for implementing tool interface requirements is vital in modern automotive diagnostic ecosystems. Through well-defined components such as control system design, structured integration strategies, standard specifications, centralized validation and rigorous implementation/testing, automotive OEMs can deliver scalable, maintainable, and high-quality diagnostic tools. While the process demands strong cross-functional collaboration and adherence to standards, the benefits of consistency, traceability, and reduced rework make it a worthwhile investment. These components defined in this process not only limited to Controls Integration but also integrates Data, Platform, Presentation and Process integration **[5]** Future enhancements may include AI-driven validation and automated requirement transformation tools to further streamline the development lifecycle.

REFERENCES

[1] H M. Kienle and H. A. Muller, "Requirements of Software Visualization Tools: A Literature Survey," 2007 4th IEEE International Workshop on Visualizing Software for Understanding and Analysis, Banff, AB, Canada, 2007, pp. 2-9, doi: 10.1109/VISSOF.2007.4290693. keywords: {Software tools; Visualization;Scalability;Softwarequality;Softwaremaintenance;Reverseengineering;Conferenceproceedings;Usability;Informationfiltering;Information filters}
[2] Title: Assessment of Safety Standards for Automotive Electronic Control Systems Creator(s): Van Eikema Hommes, QiCorporate Creator(s) : John A. Volpe National Transportation Systems Center (U.S.)Corporate Contributor(s) : United States. Department of Transportation. National Highway Traffic Safety Administration. Published Date: 2016-06-01.Report Number DOT-VNTSC-NHTSA-13-03;DOT HS 812 285; URL :https://rosap.ntl.bts.gov/view/dot/12302
[3] Lipol, Lefayet Sultan, and Jahirul Haq. "Risk analysis method: FMEA/FMECA in the organizations."International Journal of Basic & Applied Sciences 11.5 (2011): 74-82
[4] Thomas and B. A. Nejmeh, "Definitions of tool integration for environments," in IEEE Software, vol. 9, no. 2, pp. 29-35, March 1992, doi: 10.1109/52.120599.

[5] Fredrik Asplund, Martin Törngren,The discourse on tool integration beyond technology, a literature survey, Journal of Systems and Software, Volume 106,2015,Pages 117-131,ISSN 0164-1212, https://doi.org/10.1016/j.jss.2015.04.082.
[6] Sharma, Kapil Dev, and Shobhit Srivastava. "Failure mode and effect analysis (FMEA) implementation: a literature review."Journal of Advance Research in Aeronautics and Space Science 5.1 (2018): 1-17.
[7] Assawinjaipetch, Panuwat, et al. "Unified Diagnostic Services Protocol Implementation in an Engine Control Unit." (2013)
[8] M. Rodriguez, M. Piattini and C. Ebert, "Software Verification and Validation Technologies and Tools," in IEEE Software, vol. 36, no. 2, pp. 13-24, March-April 2019, doi: 10.1109/MS.2018.2883354.
[9] Allmann, Christian, Lydia Winkler, and Thorsten Kölzow. "The requirements engineering gap in the oem-supplier relationship." Journal of Universal Knowledge Management 1.2 (2006): 103-111.
[10] A. K. Thurimella and D. Janzen, "Metadoc Feature Modeler: A Plug-in for IBM Rational DOORS," 2011 15th International Software Product Line Conference, Munich, Germany, 2011, pp. 313-322, doi: 10.1109/SPLC.2011.17.



10.22214/IJRASET

45.98

IMPACT FACTOR: 7.129

IMPACT FACTOR: 7.429

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY