



HIGHLY AUTOMATED TRANSPORTATION SYSTEMS

Automation Software Safety

Huafeng Yu, Wassim Najm (HASS)
Trung Pham, Derek Morgan (FAA)



U.S. Department of Transportation



Highly Automated Systems Safety
Center of Excellence

CONTENT

Introduction	1
Safety Challenges of Automation Software	2
Automation Software Safety Assurance Framework	3
Software Development Process	4
Software Safety Assurance	5
Software Risk Management	6
Structured Safety Argument	7
Operational Safety Evaluation	8
Standards and Compliance	9
Safety Certification	10
Conclusion	11

ACRONYMS

AI	<i>artificial intelligence</i>
ASSA	<i>Automation Software Safety Assurance (framework)</i>
FAA	<i>Federal Aviation Administration</i>
GSN	<i>Goal Structuring Notation</i>
HASS	<i>Highly Automated Systems Safety (Center of Excellence)</i>
HATS	<i>highly automated transportation systems</i>
ML	<i>machine learning</i>
NASA	<i>National Aeronautics and Space Administration</i>
SDLC	<i>software development lifecycle</i>
SRM	<i>software risk management</i>
USDOT	<i>U.S. Department of Transportation</i>
V&V	<i>verification and validation</i>

DISCLAIMER

The views and opinions expressed in this document are the author’s and do not necessarily reflect those of the U.S. Department of Transportation (USDOT). The contents do not necessarily reflect the official policy of the USDOT. The U.S. Government does not endorse products, manufacturers, or outside entities. Trademarks, names, or logos appear only because they are considered essential to the objective of the presentation. They are included for informational purposes only and are not intended to reflect a preference, approval, or endorsement of any one product or entity.

COMMUNICATIONS TEAM
Denise Baker (HASS)



INTRODUCTION

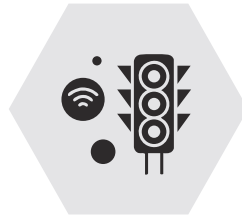
01

New technologies have been adopted at an unprecedented speed in transportation, expanding the application and use of highly automated transportation systems (HATS), such as unmanned aircraft systems, self-driving cars and trucks, and autonomous trains and ships.

Many emerging applications are based on new technologies like **AI/ML, big data, cloud computing**, and **digital twins** and are equipped with new capabilities to integrate large amounts of resources—from data, communication, and hardware to human knowledge—to achieve design and operations objectives.

Due to the high criticality and complexity of software in HATS and transportation systems in general, software safety is becoming one of the most important issues in the design and deployment of these systems.

EXAMPLES OF AUTOMATION SOFTWARE



Intelligent Transportation Systems

Traffic signal control
and management

Real time monitoring
and data analysis



Autonomous Vehicles

Automated
driving software
for perception,
localization,
planning and
control.



Unmanned Aircraft Systems

UAS automation
software for
perception,
localization, planning
and control



Railway Automation Systems

Train control and
signaling systems.

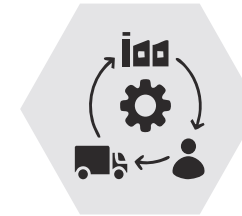
Railway traffic
management



Marine Automation Systems

Maritime fleet
management

Ship performance
and voyage
optimization



Logistics and Supply Chain Automation

Supply chain
and logistics
management.

Logistics and route
planning.

DOMAIN SPECIFIC CHALLENGES



SAFETY CHALLENGES OF AUTOMATION SOFTWARE

02

Different modes of transportation share common challenges, but each also faces unique issues stemming from vehicle types, operating environments, safety objectives, regulation requirements, and coordination methods.

COMMON CHALLENGES

1. **Complexity** of software and **scalability** of assurance tools.
2. **Estimation of probability** and safety impact caused by software failures.
3. Interaction with **unpredictable human behaviors**.
4. Infinite possibilities of **environmental conditions**.
5. Unexplainable and **unpredictable automation** behavior (e.g., AI/ML).
6. **Formalization of requirements** with uncertainties and unknowns.
7. **Expensiveness of resources**, including cost and time, as well as market pressure to follow the full software safety process.



AUTOMATION SOFTWARE

SAFETY ASSURANCE

FRAMEWORK

03

The **Automation Software Safety Assurance (ASSA) framework**, developed by USDOT HASS in collaboration with FAA and NASA, is a reference framework and tool for helping automation software safety experts and those adjacent identify, validate, assess, and deploy automation software within HATS.

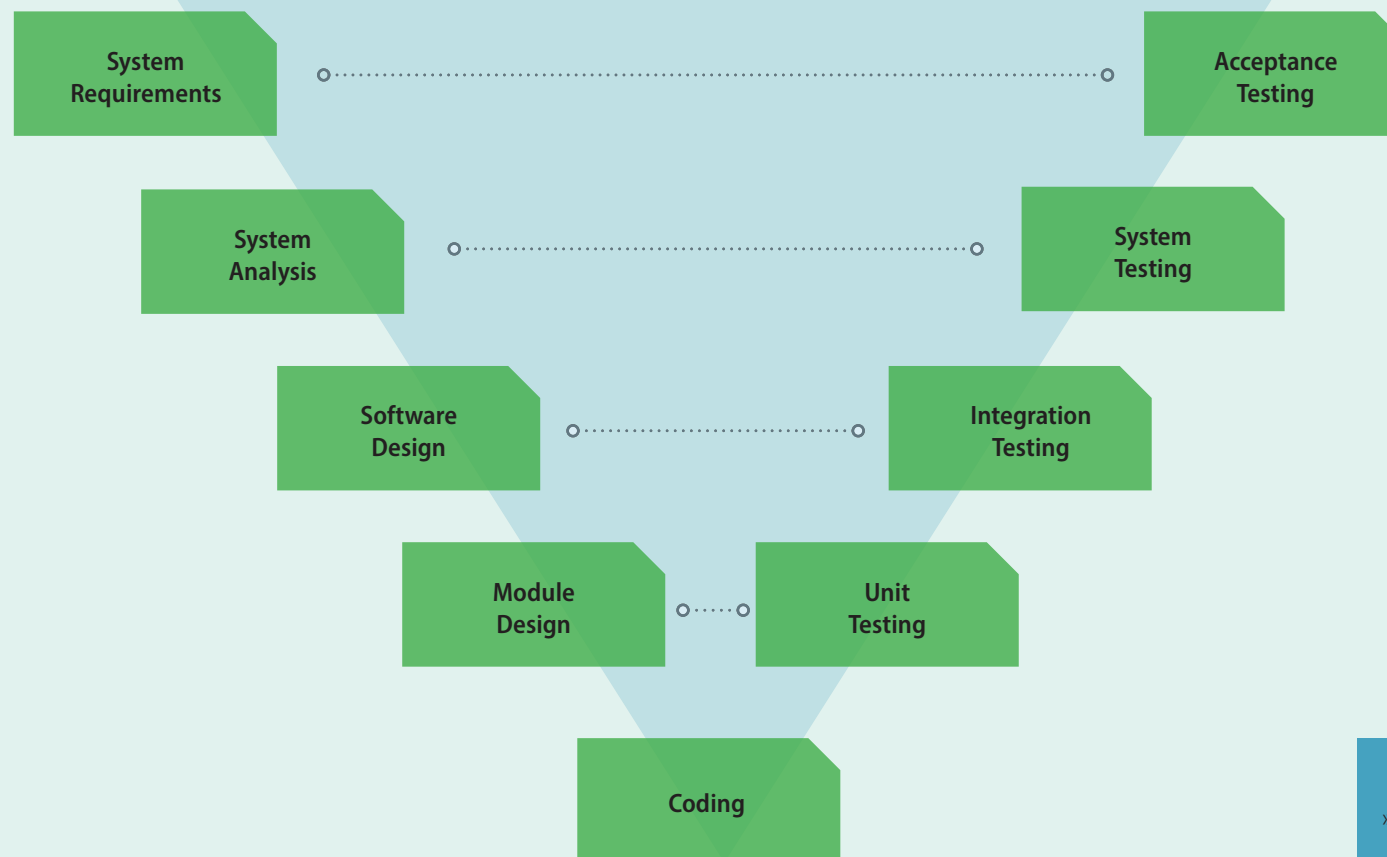
Compared to system safety, automation software safety is a relatively new field. Many existing system safety methods, tools, processes, and standards can be adapted and reused for software safety considering software is an integral part of a system.

The ASSA framework is not designed as nor intended to be a regulatory or mandatory framework but rather a reference to help better understand, plan for, and enact safety assurance efforts to the specific needs and contexts of a given automation software.

This framework acknowledges the differences between system safety and automation software safety, aiming to provide a comprehensive foundation for safety processes, techniques, standards, and assessments to improve the development, implementation, integration, deployment, and operation of automation software safety in HATS.



V-MODEL



The conventional software development lifecycle (SDLC) is well standardized in the V model. While the V model is commonly employed in SDLC, it poses challenges when employed to develop highly automated software and applications, such as AI/ML.

When developing automation software, there are several new features or factors to consider—human factors and interactions, more complex environments, intricate coordination and teaming, AI/ML integration, big data processing, increasing certification requirements, etc.

AI/ML requires a matured development process in which lifecycle activities—from development to deployment to operations—are well defined.



AI DEVELOPMENT

- » Gather system, operational, and data requirements
- » Model selection and configuration
- » Feature embodiment and model training
- » Model implementation



AI DEPLOYMENT

- » Environment and hardware setup
- » Software integration and optimization
- » Performance and reliability evaluation



AI OPERATIONS

- » Model operations
- » Data update
- » Model monitoring and maintenance
- » Contingency management

SOFTWARE SAFETY ASSURANCE

05

Safety assurance implies systematic methods, tools, technologies, and processes to ensure systems, products, or processes operate safely by avoiding unacceptable safety risks. Software safety assurance is part of system safety assurance.

Software verification and validation (V&V) is a process used to assess whether a software system meets its requirements and intended use. Verification assesses if the software implementation meets its requirements, while validation assesses if the software meets users' needs. There are four main approaches for V&V: **Peer Review & Inspection, Automatic Testing, Program Analysis, and Formal Methods**. However, ambiguous requirements, limited resources, frequent changes, and large complex systems can make V&V difficult.

When considering automation software, in general, many software testing methods can be used for automation software with or without adaptations:

- **Automation Software**—Additional factors (automation software interactions with its environment, environmental uncertainties, human behavior) may require test scenarios to cover not only the software but also any components with which it interacts.
- **AI/ML Software**—Testing relies on substantial data-driven statistical methods. Defining test objectives can be difficult or even impossible. As new training data are available, the trained model may change entirely, requiring reperforming all previous testing.

SOFTWARE SAFETY ASSURANCE TECHNIQUES

While there are many factors to consider when identifying and implementing the right software safety assurance methods, many software assurance techniques already exist.

Code Review



Reviews check software code for quality, readability, and maintainability, as well as its conformance to industry standards or best practices.

Static Analysis



Software source code is analyzed against predefined metrics or patterns to identify potential defects without executing the software.

Verification



Software implementation is checked by reviewing, analyzing, or testing to determine if software requirements are met.

Simulation



Simulation uses digital models to represent a process or system in the real world and compute its evolution based on these models.

Functional Testing



Software is tested against its functional requirements and specifications, such as intended functionality.

Nonfunctional Testing



Nonfunctional testing focuses on nonfunctional requirements like performance, security, and usability.

Regression Testing



Software testing is used to determine if the behavior of the software is kept unchanged and if new defects are introduced after software updates.

Integration Testing



Testing is performed after the integration of software units and checks interfaces and exchanged data as well as emerging properties.

Performance Testing



Performance testing aims to measure the speed, scalability, endurance, stress, and more of the software running in a real or realist environment.

Security Testing



Software is tested to identify security vulnerabilities like unintended weaknesses or flaws to reduce risks posed by attackers.

Formal Verification



Mathematical methods are used to prove or disprove the correctness of the software regarding expected properties like invariance and liveness.

Machine Learning



Machine learning is used to analyze large volumes of data to identify patterns and anomalies based on training from large amounts of data.



06

SOFTWARE RISK MANAGEMENT

Software risk management (SRM) aims to identify, assess, and mitigate potential risks to system safety by studying the impact of hazards and risks. SRM activities consider the overarching system rather than only the software itself.

RM borrows concepts from system safety management, but the efforts introduced by SRM extend beyond the software and can mitigate potential risks to the entire system's safety. Compared to V&V, SRM does not focus on finding bugs or defects; rather, it studies the impacts of bugs or defects on system safety. As such, SRM activities are required to take the overarching system into account, not just the software itself.

Both qualitative and quantitative approaches can be used to identify and assess software risks, such as developmental and operational experience and lessons learned as well as real-world, simulation, or external data. Using information learned from the assessment phase, risk mitigation measures are implemented to reduce the opportunities of identified risks arising and lessening their impact on the larger system.

RISK MANAGEMENT

The SRM process consists of three major components: **risk identification**, **risk assessment**, and **risk mitigation**. As software is part of a larger system, the technologies used in system risk management can be applied to SRM with adaptations.

RISK IDENTIFICATION



The first step in SRM is **risk identification**, which identifies and defines potential hazards and risks that may harm the system, including unsuccessful completion, degraded performance, or even safety consequences. The identification and definition are usually based on an existing knowledge base, past experience, expert judgement, systematic analysis, and so forth. This step is usually performed by software, system, or safety engineers.

RISK ASSESSMENT



Risk assessment has two arms. The first is to determine the properties of hazards like their causes and effects. The second is completing an overall assessment of each hazard's probability of occurring, severity, and overall risk. In practice, once a potential risk is identified, the next step is to assess its likelihood of occurrence and safety impact.

RISK MITIGATION



Risk mitigation is the minimization of risks posed by hazards through dedicated measures like elimination, prevention, operational avoidance, active monitoring, and contingency management and/or recovery. Risk mitigation is often performed in conjunction with risk identification and assessment to determine the effectiveness of mitigation measures in reducing risk against the potential new risks they may introduce by failing.

A safety argument is adopted in safety-critical systems to show that a system is acceptably safe or free from harm regarding its intended use. Safety arguments systematically use well-defined safety claims, evidence, and reasoning techniques to demonstrate a sound case for system safety.

EVALUATION OF THE ARGUMENT



Clarity—A clear and structured representation of safety claims and evidence.



Transparency—The structure, reasoning, and justification are transparent to stakeholders.



Complexity— Safety arguments can become complex, especially for large and complex systems.



Usefulness— Ensuring the safety argument is useful and acceptable by stakeholders.



Traceability—All safety requirements are traced to evidence and vice versa.



Compliance— Comply with safety standards, guidance, and regulations.



Completeness— All safety requirements are traced to evidence and vice versa.



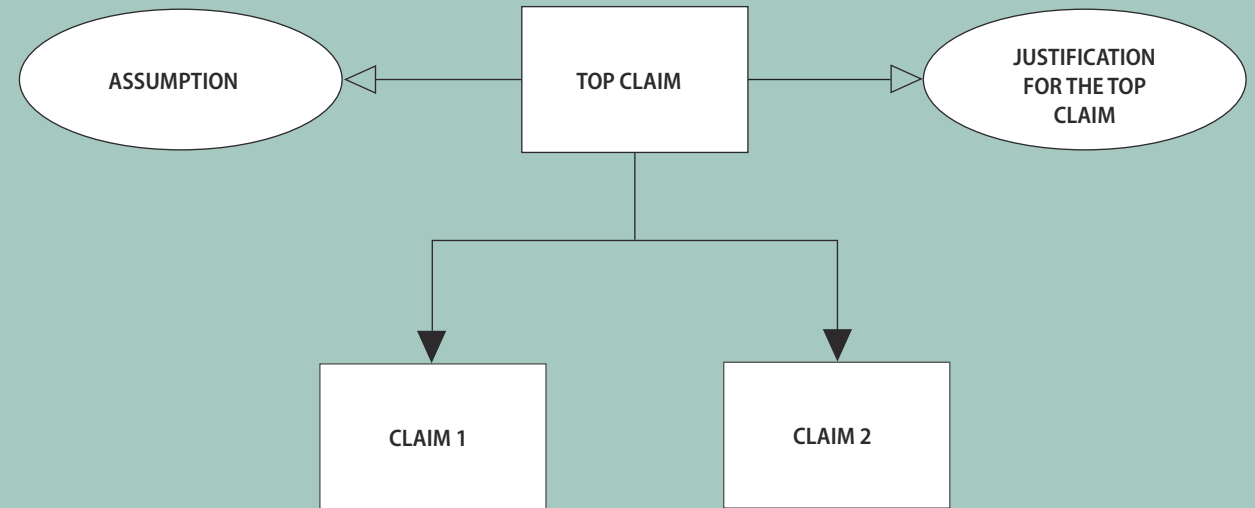
Update-to-date— Easy to keeping the argument up-to-date as the system evolves.

STRUCTURED ARGUMENT

Safety arguments can be expressed through text but are more clearly illustrated through structured diagrams, such as the Goal Structuring Notation (GSN).

Similar to a workflow diagram, GSN uses shapes to represent the elements of the argument and their relationships:

- Goals as rectangles.
- Evidence as circles.
- ▱ Assumptions and context as parallelograms.
- ▶ Reasoning and justifications as arrows or annotations.



In the context of software safety, the safety argument is a promising technique for bridging the concepts of safety risk management and software assurance activities. Assurance results, such as V&V, can be used as evidence in an argument to support safety requirements, verify if safety goals are met, or determine if safety risks are mitigated.

08

SAFETY EVALUATION

Safety evaluation involves evaluating whether a system consistently operates safely and reliably in the lab or within its intended environment (i.e., evaluating the operational safety of the system).

The primary objective is to identify and mitigate potential operational risks, ensure adherence to operational safety guidance and standards, and foster confidence in the system's ability to operate without posing harm to humans or the environment.

It is important to note that, in safety evaluation, evaluators may not have access to internal system information, so they only rely on data, system behavior, and safety impacts collected or identified during system operations.

Qualitative and quantitative approaches can identify and assess software risks based on development and operational experience and lessons learned as well as real-world, simulation, or external data.



STANDARDS AND COMPLIANCE

Software and safety standards are established and adopted to improve safety and efficiency, minimize potential risks, and reduce resources (i.e., costs and time). Most standards are developed by Standard Development Organizations (e.g., ISO, ANSI, SAE), which organize groups of domain experts and use consensus-based processes to develop and maintain standards to meet industry needs.

Depending on the criticality of the system’s safety, performance, and cost, standards requirements can have varying levels of conformance and they may be voluntary or strictly regulated.

While standards vary greatly across industries as well as across software and systems employed, the following is a shortlist of standards applicable in the transportation sector.

SCOPE	SOFTWARE	AI	SYSTEM
Development Process	ISO/IEC/IEEE 12207, ASPICE	ISO/IEC 5338, ISO/PAS 8800	ISO26262, ISO/IEC/IEEE 15288
Safety Assurance	ISO/CD TS 5083, ISO26262, RTCA DO-178C, ISO/IEC/IEEE 15026, ISO/IEC/IEEE 29119	ISO/IEC TS 25058, ISO/IEC 25059	ISO/CD TS 5083, ISO26262, SAE J3279, ISO/IEC/IEEE 15026, ISO 21448
Risk Management	ISO26262, RTCA DO-178C, SAE J3307	ISO/IEC 23894, ISO/PAS 8800	ISO26262, SAE J3307, IEC 61508, ISO 21448, SAE ARP 4761A
Safety Evaluation	ISO26262, RTCA DO-178C	ISO/IEC TS 25058, ISO/PAS 8800	ISO26262, ISO 21448, ISO 3450X
Safety Argument	ISO 29119, ANSI/UL 4600, MISRA GASA/SC	ISO/PAS 8800	ANSI/UL 4600, MISRA GASA/SC



SAFETY CERTIFICATION

10



SOFTWARE SAFETY
CERTIFICATION
STEPS

Software safety certification is the key for ensuring software is indeed safe and meets industry standards, regulatory requirements, and user expectations.

Consider the following steps:

1 ENGAGE
CERTIFICATION
AUTHORITY

2 IDENTIFY
CERTIFICATION
REQUIREMENTS

3 PREPARE
CERTIFICATION
DOCUMENTS

4 ADDRESS
AUDIT
FINDINGS

5 RECEIVE
CERTIFICATE

6 MAINTAIN
CERTIFICATION

Software safety certification is a key process that ensures software is safe and meets industry standards, regulatory requirements, and customer expectations.

By following structured processes, performing required safety assurance techniques, leveraging industry best practices, using appropriate tools, and documenting safety evidence along the way, safety certification can be carried out and achieved by software or system owners or operators, third parties, or regulators.



Automation software, particularly software implemented with artificial intelligence, plays an increasingly critical role in highly automated transportation systems. Due to their high criticality and complexity, software safety becomes one of the key considerations in the design and deployment of these systems.

This white paper highlights the growing application scenarios, significance, and challenges of software safety across key transportation sectors including aviation, automotive, transit, rail and maritime, especially while novel technologies like artificial intelligence and machine learning are being integrated. It also outlines foundational concepts in automation software safety, and presents a reference framework for automation software safety assurance, designed to enhance safety throughout the system lifecycle.

Core technical topics, such as software development process, software safety assurance, operational safety evaluation, and software risk management are first systematically reviewed. Furthermore, the authors emphasize the necessity of structured safety arguments and compliance with industry consensus standards to ensure system and operational safety. The paper concludes with a review of the safety certification process required prior to public deployment of autonomous systems.

This paper does not establish any mandatory or regulatory requirements for automation software safety; rather it provides a foundation for safety processes, techniques, standards, and assessments to support the development, implementation, integration, deployment, and operation of automation software for safe operations of highly automated transportation systems.