# Digital Assurance for Embedded Space Controllers

Bharat Bhrgava
bbshail@purdue.edu
Purdue University
USA

## 1 500 words Summary

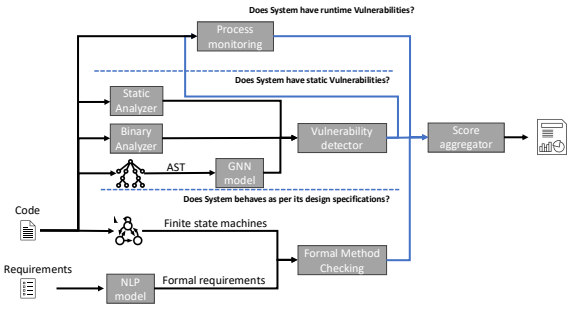


**Figure 1: Diagram of proposed framework.**

Our goal is to provide rigorous digital assurance for high-consequence systems by designing a unified framework that integrates verification, fault detection, monitoring, failure impact quantification, and defensive mechanisms. This approach will provide a more efficient, and interpretable assurance process, ensuring that systems remain secure and functional even in under faulty conditions. We also aim to introduce a quality metric that offers a holistic view of attacks/faults and their impact. This will enable the designers to make informed trade-off decisions.

Currently, digital assurance is addressed using separate tools and techniques that focus on individual aspects of system reliability. For instance, model checking is widely used to verify that a system behaves as intended, but it is time-consuming and requires specialized expertise. fault analysis tools like static analysis, binary analysis, and penetration testing provide detailed reports of potential weaknesses, but these reports are sometimes less interpretable and does not cover the holistic view of the system. Monitoring techniques, such as anomaly detection, can identify ongoing attacks or deviations from expected behavior, but they operate independently from other assurance methods. Defensive mechanisms, like task redundancy and address randomization, mitigate the impact of intelligent attacks but are not integrated into a unified system for digital assurance. One of the key limitations of current practices is the fragmentation of tools, making it difficult to connect insights across different assurance processes. Additionally, existing impact metrics for faults often focus only on the count of detected issues, neglecting their severity and potential impact.

Our approach addresses these limitations by designing a modular pipeline that integrates verification, fault detection, monitoring, and defensive mechanisms. This unified framework ensures fine-grained analysis through a feedback loop that enhances overall system assurance. For example, faults identified during analysis will directly influence monitoring strategies, while real-time monitoring data can update formal verification requirements to address emerging threats. We wil use Predictive state-based analytics to monitor system behavior, searching for discrepancies in data and identifying hidden patterns indicative of faults or attacks. We assume that attackers are intelligent, and we will use Markov decision process to model the behavior of the attack. Additionally, our framework incorporates advanced AI techniques to enhance fault detection and streamline the process. Graph Neural Networks (GNNs) will improve the accuracy of fault detection by analyzing system code and execution paths, while Natural Language Processing (NLP) will translate informal system requirements into mathematical representations for formal model checking. We will use interpretable techniques in machine learning to summarize technical outputs into accessible reports, making the results interpretable for non-technical stakeholders.

A key innovation in our framework is the introduction of a new metric for quantifying the impact of faults. Instead of counting detected issues, we will measure the value of metric to each vulnerability based on its potential impact. These scores will be weighted according to their impact, and faults across different security subdomains will be aggregated using a weighted geometric mean. This approach ensures that critical faults have a substantial effect on the overall metric, allowing stakeholders to prioritize issues more effectively.

This unified framework will be particularly valuable for critical operations where system failures can have catastrophic consequences, including space, defense, nuclear plant, etc. By providing comprehensive and rigorous digital assurance, improving decision-making, and making advanced assurance techniques accessible to non-technical stakeholders, our framework will help reduce the risk of system failures and cyberattacks. It will also improve reliability and security of high-consequence systems.

Despite its potential, challenges remain in ensuring generalizability and scalability across diverse system architectures and threat landscapes. To address this, we will employ distributed techniques for parallel execution across system components, reducing bottlenecks and enhancing resilience by preventing single points of failure. This approach ensures the framework remains adaptable across various high-consequence systems.

## 2 notes

### 2.1 What are you trying to do?

We aim to provide comprehensive digital assurance for high-consequence systems, ensuring that they function efficiently and as intended, even in the face of potential vulnerabilities or external threats. Digital assurance involves a holistic approach to verifying the reliability of the system, identifying vulnerabilities, monitoring live performance, and implementing safeguards to prevent and mitigate

attacks. Our objective is to develop a framework that not only integrates these processes but also makes their results interpretable for both technical and non-technical stakeholders. We also propose a novel software quality metric that presents a holistic view of the vulnerability and their severity for easier comparison of similar softwares.

## 2.2 How is it done today, and what are the limits of current practice?

Today, digital assurance is addressed through individual tools and techniques that target specific aspects of the process:

- Verification (Model Checking): Model checking is widely used to ensure that a system behaves as per its design specifications. While effective, this method requires converting the system code into finite-state models and verifying them against requirements. Tools exist, but they demand significant technical expertise to use and interpret, making them inaccessible for non-expert users.
- Vulnerability Analysis: Static analysis, binary analysis, and penetration testing are used to detect vulnerabilities. These methods rely on state-of-the-art tools that produce highly technical reports, detailing how vulnerabilities could be exploited. However, these reports are difficult to interpret for decision-makers without technical backgrounds.
- Live Monitoring: Techniques like anomaly detection, sensor data analysis, and model checking during runtime can detect ongoing attacks or deviations from expected behavior. These methods are effective but are not well-integrated into broader assurance frameworks, and they often focus narrowly on specific attack scenarios or system behaviors.
- Defensive Mechanisms: Techniques such as task redundancy, address randomization, and checksum validation help mitigate the impact of attacks. However, these defenses operate in isolation and are not part of a unified system for digital assurance.

Numerous studies have been conducted on software grading methodologies. Broadly, these approaches can be categorized into static and dynamic metrics. Static metrics analyze the code without execution, identifying vulnerabilities through static analysis tools. These methods typically assign a score based on the number of detected vulnerabilities, without considering execution behavior.

Conversely, dynamic metrics focus on runtime vulnerabilities, identifying security issues that manifest during execution. Despite advancements in vulnerability detection across various system components (e.g., authentication flaws, SQL injection attacks), scoring mechanisms often rely solely on the count of detected vulnerabilities. The severity of each vulnerability is typically not accounted for by default and is instead treated as a tunable parameter.

Key limitations of the current practice include:

- Fragmentation: The assurance process is not unified. Tools operate independently, and their results are difficult to combine and analyze comprehensively.
- Technical Expertise Requirements: Most tools produce outputs that are inaccessible to non-technical users, creating a barrier for wider adoption.
- Impact Quantification: There is no consistent metric for quantifying the impact of potential failures or vulnerabilities, making it difficult for users to prioritize or choose the best solutions.
- Requirement Completeness: Exhaustive requirement generation for formal verification is challenging, often leading to incomplete or insufficient models.
- Cross-Module Analysis: Current methods fail to account for how a vulnerability detected in one part of the system might affect other parts, limiting their ability to assess system-wide impacts.
- Work with older IDS systems: We need to use defence mechanisms which will not make current IDS systems provide more false positives. This may need a channel of communication with current IDS systems.
- Software quality metric accumulation: the effects of a vulnerability count alone, often neglecting the contextual severity and potential impact of each detected issue.

## 2.3 What is new in your approach, and why do you think it will be successful?

Our approach is innovative and comprehensive, aiming to unify digital assurance methods into a single, modular framework that:

- Integrates Modules: By combining verification, vulnerability detection, live monitoring, and defensive mechanisms into a cohesive pipeline, we ensure that each step informs the others, improving overall system assurance. For example, vulnerability analysis results can directly inform live monitoring strategies or model-checking requirements.
- Simplifies Outputs: We will leverage large language models (LLMs) to aggregate and summarize highly technical outputs into accessible formats for non-technical stakeholders. This will enable decision-makers to interpret results and act on them without relying on technical intermediaries.
- Quantifies Risk and Impact: We will introduce a combined metric for evaluating system quality and the impact of potential failures or vulnerabilities. This will help users make informed decisions about alternative software options.
- Uses Advanced Techniques: Our framework will incorporate cutting-edge methods such as Graph Neural Networks (GNNs): Applied to abstract syntax trees and execution paths, these will enhance vulnerability detection accuracy. Natural Language Processing (NLP): Used to translate informal requirements into mathematical representations for model checking, addressing the challenge of exhaustive requirement generation.
- Provides Flexibility and Generalization: The framework is modular and extensible, allowing additional tools or techniques to be integrated as needed. This ensures adaptability to new challenges and technologies.
- Cross-Module Impact Analysis: By linking the results from one module to others, our framework will provide a more holistic view of system health and the potential cascading effects of vulnerabilities.

For grading a software we will use this plan of action:

- Scoring Individual Vulnerabilities: Each detected vulnerability is assigned a severity score using a standardized metric like the Common Vulnerability Scoring System (CVSS) or a custom scale. The overall risk score is calculated by summing the severity scores of all vulnerabilities, each weighted by its importance. The weight accounts for factors such as exploitability and impact.
- Aggregating Across Subdomains: Since different security subdomains, such as authentication and access control, have varying importance, their scores are combined using a weighted geometric mean rather than a simple average. This approach ensures that a highly vulnerable subdomain has a disproportionately large effect on the overall security score. Each subdomain's contribution is adjusted based on its relative importance.
- Final Score Interpretation: The aggregated security score is then normalized to a scale from 0 to 100, where higher scores indicate better security. Specific thresholds categorize security levels as excellent, acceptable, or poor.

We believe this approach will succeed because it addresses the core limitations of current methods while leveraging advancements in AI, machine learning, and software analysis to provide actionable, interpretable, and comprehensive assurance.

## 2.4 Who cares? If you are successful, what difference will it make?

The proposed framework is crucial for organizations and industries where system failures can have catastrophic consequences, including:

- Critical Infrastructure: Operators of power grids, nuclear plants, and transportation networks can rely on our framework to ensure uninterrupted services and mitigate risks of cyberattacks.
- Space: High-assurance systems are critical for mission success and the safety of personnel and equipment.

If successful, our framework will:

- Reduce the likelihood of system failures and cyberattacks.
- Enable faster and more informed decision-making by providing interpretable assurance metrics.
- Lower the barrier to adopting advanced assurance techniques by eliminating the need for extensive technical expertise.
- Improve trust in software systems across high-stakes industries.

## 2.5 What are the risks?

- Complexity of Integration: Combining diverse modules (e.g., model checking, live monitoring, vulnerability analysis) into a unified framework may pose technical and logistical challenges.
- Scalability: Ensuring that the framework works efficiently across systems of varying sizes and complexities could be challenging.
- Accuracy and Reliability of Metrics: Developing meaningful and universally applicable metrics for quality and risk assessment may require significant experimentation and refinement.
- Interpretability of AI Models: While LLMs and GNNs hold promise, ensuring their outputs are both accurate and interpretable may be difficult.
- Adaptability: As software systems and attack methods evolve, the framework must remain flexible and up-to-date to remain effective.

## 2.6 What are the mid-term and final "exams" to check for success?

Mid-term goals:

- Development of a prototype framework that integrates model checking, vulnerability analysis, live monitoring, and defensive mechanisms.
- Validation of the framework on benchmark systems with documented vulnerabilities and assurance requirements.
- Demonstration of LLM-generated reports that non-technical users can interpret.

Final goals:

- Deployment and testing of the framework in real-world systems across multiple domains (e.g., healthcare, finance, critical infrastructure).
- Successful generation of risk and impact metrics that align with industry standards and user expectations.
- Evidence of reduced system vulnerabilities, improved decision-making, and enhanced trust in high-consequence systems.

## 2.7 How long with this project take?

The duration of this project will be three years.

## 2.8 How much will it cost?

The estimated budget for this project is $200k$ per year. This will be used for student and faculty salary.