FUSION: A Cloud-Based Framework for Integrated Software-Hardware Reliability Assessment Using Advanced Analytics and RBDs

Kazuhira Okumoto* and Hoang Pham**
* Sakura Software Solutions LLC, Naperville, IL, USA 60563.
** Department of Industrial and Systems Engineering, Rutgers University, Piscataway, NJ, USA 08854.

Keywords: Cloud-Based Digital Reliability Tools, Operational-Phase Analysis, Software-Hardware Integrated System Reliability, Reliability Block Diagrams

Abstract - As artificial intelligence, automation, and robotics become increasingly integral to modern systems, ensuring the reliability of both software and hardware components is paramount. Existing models primarily address software defects during development, leaving a critical gap in operational-phase reliability assessment. Similarly, while hardware reliability is well-studied, integrated models that encompass both software and hardware are limited. This paper introduces FUSION, a cloud-based tool designed to bridge this gap by aligning software failure rates with hardware reliability models, creating a unified framework for operational-phase reliability analysis. FUSION leverages advanced analytics and reliability block diagrams (RBDs) to provide comprehensive system evaluations. Key contributions include: (1) the development of an integrated software-hardware reliability model, (2) the implementation of automated reliability analysis algorithms, and (3) the creation of a digital tool for system-level reliability assessment. By integrating these elements, FUSION enhances reliability analysis, enabling practitioners to improve system resilience, user satisfaction, and operational efficiency

1. Introduction

In the era of rapid technological advancements, the integration of artificial intelligence (AI), automation, and robotics into modern systems has become increasingly prevalent. These technologies promise significant enhancements in efficiency, functionality, and user experience. However, they also introduce complex challenges, particularly in ensuring the reliability of both software and hardware components. Even minor defects can lead to substantial disruptions, underscoring the critical need for robust reliability assessment methodologies [11-12].

Current reliability models predominantly focus on software defects during the development phase, providing valuable insights into potential issues before deployment [4]. Despite their utility, a significant gap remains in assessing reliability during the operational phase, when software interacts with real-world conditions and user environments. This phase is crucial, as it is often when latent defects manifest, potentially leading to system failures [1]

On the hardware side, reliability assessment is well-established, with numerous models and methodologies developed to predict and mitigate failures [2]

However, the interaction between software and hardware components adds a layer of complexity that existing models do not fully address. Integrated reliability models that consider both software and hardware are limited, leaving practitioners without comprehensive tools to evaluate and enhance system resilience [3]

To bridge this gap, we introduce FUSION, a cloud-powered tool designed to integrate software and hardware reliability models for operational-phase analysis. FUSION aligns software failure rates with hardware reliability models, creating a unified framework that leverages advanced analytics and reliability block diagrams (RBDs) for comprehensive system evaluations. This tool not only facilitates accurate failure predictions but also provides actionable insights to improve system resilience, user satisfaction, and operational efficiency.

The key contributions of this paper are threefold: (1) the development of an integrated software-hardware reliability model, (2) the implementation of automated reliability analysis algorithms, and (3) the creation of a digital tool for system-level reliability assessment. By combining these elements, FUSION represents a significant advancement in reliability engineering, offering practitioners a powerful tool to anticipate and mitigate failures in integrated systems.

2. Literature Review

2.1 Software Reliability Models

Software reliability has been extensively studied, with numerous models developed to predict and mitigate software defects during the development phase [9-10]. Traditional models, such as the Jelinski-Moranda model and the Musa-Okumoto model, focus on defect prediction and reliability growth during software testing. These models have been instrumental in identifying potential issues before deployment, but they often fall short in addressing reliability during the operational phase, where software interacts with real-world conditions [5], [6].

Recent advancements have introduced more sophisticated approaches, such as Bayesian networks and machine learning-based models, which offer improved prediction accuracy by incorporating historical defect data and real-time monitoring. However, these models primarily concentrate on software reliability in isolation, without considering the interplay between software and hardware components [7], [8].

2.2 Hardware Reliability Models

Hardware reliability assessment is a well-established field, with models like the Weibull distribution and the Crow-AMSAA model being widely used to predict hardware failures. These models have been effective in various applications, from consumer electronics to aerospace systems, providing valuable insights into hardware performance and failure rates [15], [16].

In recent years, there has been a growing interest in reliability assessment methods for complex hardware systems, such as deep neural networks (DNNs) and their accelerators. These methods often employ fault injection, analytical, and hybrid approaches to evaluate the reliability of hardware components under different conditions. Despite their effectiveness, these models typically treat hardware reliability in isolation, without integrating software reliability considerations [17], [18].

2.3 Integrated Software-Hardware Reliability Models

The interaction between software and hardware components adds a layer of complexity that traditional models do not fully address. Some recent studies have attempted to bridge this gap by developing integrated reliability models. For instance, Zhu and Pham [13] proposed a novel system reliability model that incorporates hardware failures, software failures, and hardware-software interaction failures. This model provides a novel perspective by classifying interaction failures into software-induced hardware failures and hardware-induced software failures [13].

While these integrated models represent a significant advancement, they often rely on complex mathematical formulations and may not be easily applicable to real-world systems. Additionally, they may lack the scalability and flexibility required for modern, cloud-based environments [19], [20].

2.4 Novel Contributions of FUSION

FUSION addresses the limitations of existing models by providing a cloud-powered tool that integrates software and hardware reliability models for operational-phase analysis. Unlike traditional models that focus on either software or hardware reliability in isolation, FUSION offers a unified framework that aligns software failure rates with hardware reliability models. This approach leverages advanced analytics and reliability block diagrams (RBDs) to provide comprehensive system evaluations [21].

Key contributions of FUSION include:

1. **Integrated Software-Hardware Reliability Model**: FUSION develops a combined model that aligns software failure rates with hardware reliability behavior, creating a unified framework for operational-phase analysis.
2. **Automated Reliability Analysis Algorithms**: The tool implements algorithms for automated reliability analysis, enabling real-time predictions and assessments.
3. **Digital Tool for System-Level Assessments**: FUSION provides a user-friendly digital tool that facilitates the creation and evaluation of reliability block diagrams, offering actionable insights to improve system resilience and operational efficiency.

By integrating these elements, FUSION represents a significant advancement in reliability engineering, offering a practical and scalable solution for modern, integrated systems.

3. FUSION Overview

The methodology behind FUSION involves several key steps and algorithms to ensure accurate and comprehensive reliability assessments:

1. **Software Customer-Found Defect Prediction**:
   - **Operational Phase Segmentation**: The operational phase of software usage is divided into distinct periods: early customer testing, customer testing, post-deployment, and customer migration to the next release. Each period exhibits varying rates of defect identification.

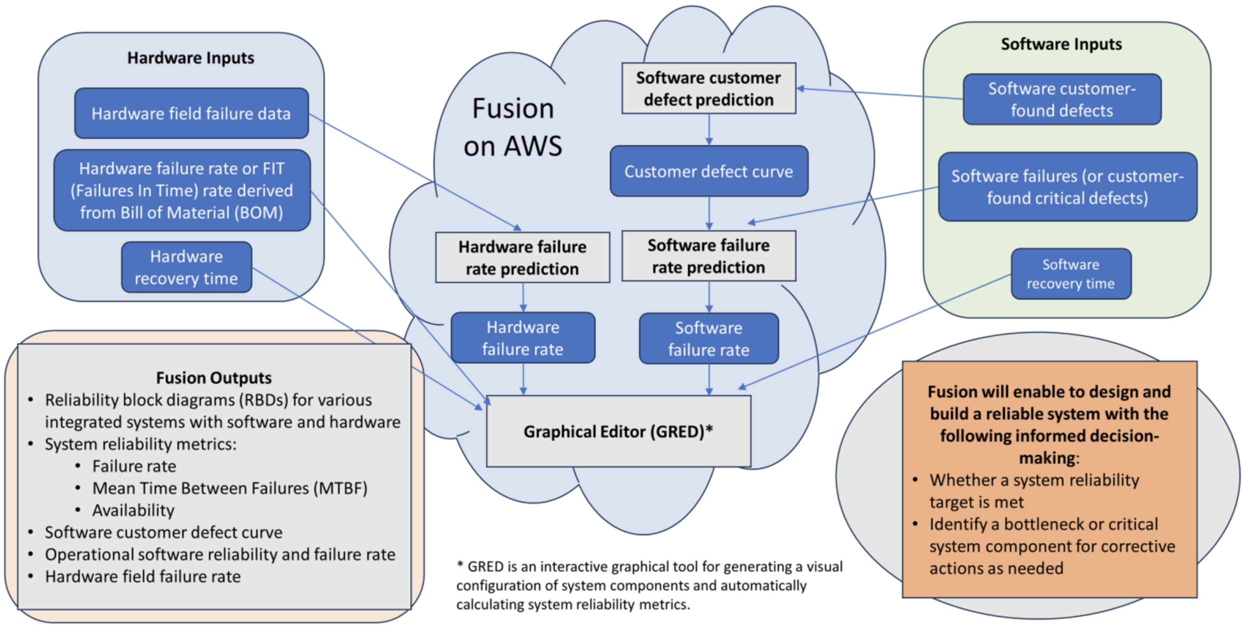

Hardware Inputs
Hardware field failure data
Hardware failure rate or FIT (Failures In Time) rate derived from Bill of Material (BOM)
Hardware recovery time
Fusion on AWS
Software customer defect prediction
Customer defect curve
Hardware failure rate prediction
Software failure rate prediction
Hardware failure rate
Software failure rate
Graphical Editor (GRED)*
Software Inputs
Software customer-found defects
Software failures (or customer-found critical defects)
Software recovery time
Fusion Outputs
Reliability block diagrams (RBDs) for various integrated systems with software and hardware
System reliability metrics:
Failure rate
Mean Time Between Failures (MTBF)
Availability
Software customer defect curve
Operational software reliability and failure rate
Hardware field failure rate
Fusion will enable to design and build a reliable system with the following informed decision-making:
Whether a system reliability target is met
Identify a bottleneck or critical system component for corrective actions as needed
* GRED is an interactive graphical tool for generating a visual configuration of system components and automatically calculating system reliability metrics.

Fig. 1. An Overview of FUSION

- **Defect Trend Model**: A defect trend model spans the entire operational phase, represented as a sequence of empirically derived straight lines. An algorithm autonomously identifies the transitions (inflection points) between these lines, dynamically adapting to distinct patterns within the data.

2. **Software Failure Rate Prediction**:

- **Transformation of Defect Data**: The cumulative predicted defect curve is transformed into the cumulative failure curve using parameters α, β, and γ. These parameters represent horizontal shift, scaling factor, and vertical scaling factor, respectively.

- **Optimization Algorithm**: The algorithm optimizes the parameters to best fit the predicted failure curve to the actual one. The objective function is based on the absolute difference between predicted defects and actual failures over the last 10 weeks. The algorithm iterates over three levels to identify the global minimum sum of squared differences (SSQ).

3. **Intelligent Interactive Graphical Editor (iGRED) for RBDs**

- **RBD Construction**: iGRED allows users to construct detailed RBDs for software and hardware systems. Built-in real-time reliability metric calculations simplify assessments and accelerate system design.

- **Preconfigured Templates**: iGRED provides templates for single-unit, active-standby, and active-active systems. Users input failure rates and recovery times, and iGRED calculates and displays reliability metrics instantly.

4. **Reliability Metrics Calculation**

- **Key Metrics**: FUSION calculates key reliability metrics, including failure rate (λ), Mean Time to Failure (MTTF), Mean Time Between Failures (MTBF), Mean Time to Repair (MTTR), and availability (A). These metrics are computed for software, hardware, and the integrated system.

- **Formulas and Calculations**: The reliability metrics are derived using established formulas and methods, such as the exponential distribution for constant failure rates and the Markov chain method for complex configurations.

In the following subsections, we detail each of the above components.

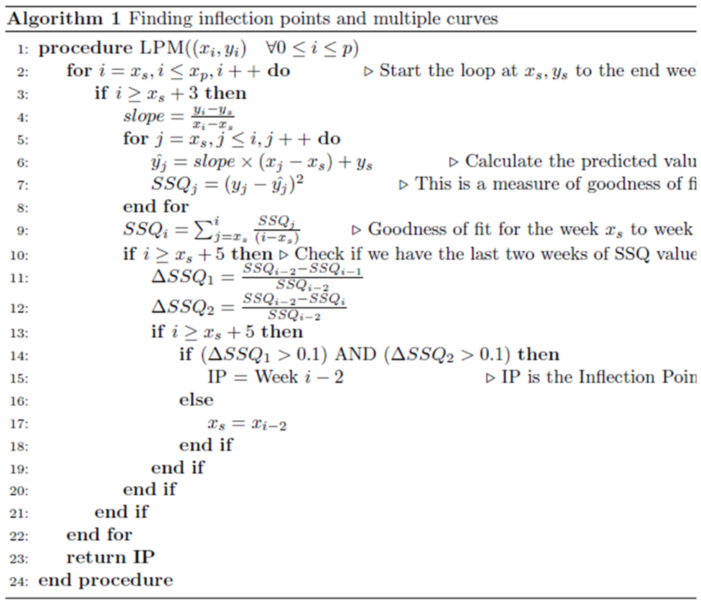
3.1 Software Customer-Found Defect Prediction

The operational phase of software usage can be divided into distinct periods: early customer testing, customer testing, post-deployment, and customer migration to the next release. Each period exhibits varying rates of defect identification. To account for this variability, we developed a defect trend model that spans the entire operational phase, represented as a sequence of empirically derived straight lines. The primary technical challenge is creating an algorithm that autonomously identifies the transitions (inflection points) between these lines. This algorithm must dynamically adapt to the distinct patterns within the data and operate seamlessly in real-time as new defect data becomes available.

The segmented linear equation is expressed mathematically to capture the distinct defect trends in each period. The cumulative defects, *m*(*x*), found by time *x* during inflection points (*j* – 1) and *j* is described as:

$$m(x) = m(x_{j-1}) + \theta_j (x - x_{j-1}) \qquad \forall x_{(j-1)} \leq x \leq x_j \qquad (1)$$

Note that θ_j represents the slope of the curve during the period. Algorithm 1 shows the algorithm that we developed.



Algorithm 1 Finding inflection points and multiple curves

```
procedure LPM(($x_i, y_i$)   $\forall 0 \leq i \leq p$)
    for $i = x_s, i \leq x_p, i++$ do                ▷ Start the loop at $x_s, y_s$ to the end wee
        if $i \geq x_s + 3$ then
            $slope = \frac{y_i - y_s}{x_i - x_s}$
            for $j = x_s, j \leq i, j++$ do
                $\hat{y}_j = slope \times (x_j - x_s) + y_s$          ▷ Calculate the predicted valu
                $SSQ_j = (y_j - \hat{y}_j)^2$            ▷ This is a measure of goodness of fi
            end for
            $SSQ_i = \sum_{j=x_s}^{i} \frac{SSQ_j}{(i-x_s)}$        ▷ Goodness of fit for the week $x_s$ to week
            if $i \geq x_s + 5$ then ▷ Check if we have the last two weeks of SSQ value
                $\Delta SSQ_1 = \frac{SSQ_{i-2} - SSQ_{i-1}}{SSQ_{i-2}}$
                $\Delta SSQ_2 = \frac{SSQ_{i-2} - SSQ_i}{SSQ_{i-2}}$
                if $i \geq x_s + 5$ then
                    if ($\Delta SSQ_1 > 0.1$) AND ($\Delta SSQ_2 > 0.1$) then
                        IP = Week $i - 2$                ▷ IP is the Inflection Poin
                    else
                        $x_s = x_{i-2}$
                    end if
                end if
            end if
        end if
    end for
    return IP
end procedure
```

Algorithm 1. Algorithm for operational software defect prediction

3.2 Software Failure Rate Prediction

FUSION introduces groundbreaking methods for predicting software defects and failures, demonstrating that software defect and failure rates remain constant during the operational phase. This capability is significant as it aligns software reliability modeling with established hardware reliability models, offering a unified approach to system reliability. The analytics for predicting software defects are robust enough to extend to hardware field failure data, providing a versatile solution for reliability analysis across various system components.

An innovative algorithm has been developed to predict software failure rates during the operational phase. The required input data includes the defect-predicted curve (i.e., the output of the defect-predicted curve) and the software failure data (i.e., critical defects). The cumulative predicted defect curve (x, y) is transformed into the cumulative failure curve (u, v) using the parameters, α, β, and γ:

$$u = \alpha + \beta x \qquad (2)$$

$$v = \gamma y \qquad (3)$$

Parameters are interpreted as follows.

– α: Horizontal shift (right if positive, left if negative, no shift if zero).

– β: Scaling factor (delay if greater than 1, acceleration if less than 1, no change if 1).

– γ: Vertical scaling factor (doubles height if 2, reduces to 50% if 0.5, no change if 1).

Algorithm for Software Failure Prediction

The algorithm optimizes the parameters α, β, and γ to best fit the predicted failure curve to the actual one. The objective function is based on the absolute difference between predicted defects and actual failures over the last 10 (default) weeks. To calculate the objective function, follow these steps:

- Calculate transformed values (u, v) using the equations (2) and (3) for given values of α, β, and γ.
- The actual failure curve (xact, yact) remains on the original (x, y) scale. Calculate vact, for u = xact using interpolation:

$$v_{act} = v_1 + \frac{v_2 - v_1}{u_2 - u_1} \times (x_{act} - u_1) \qquad (4)$$

- Calculate the sum of the absolute differences (SSQ) between vact, and yact:

$$SSQ = \sum |v_{act} - y_{act}| \qquad (5)$$

The parameter ranges are:α = 5 to 0 with a decrement of 0.5, β = 0.7 to 1.1 with an increment of 0.05, and γ = 0.86 to 1.2 with an increment of 0.02.

Solving the non-linear optimization problem:

• The algorithm iterates over three levels:

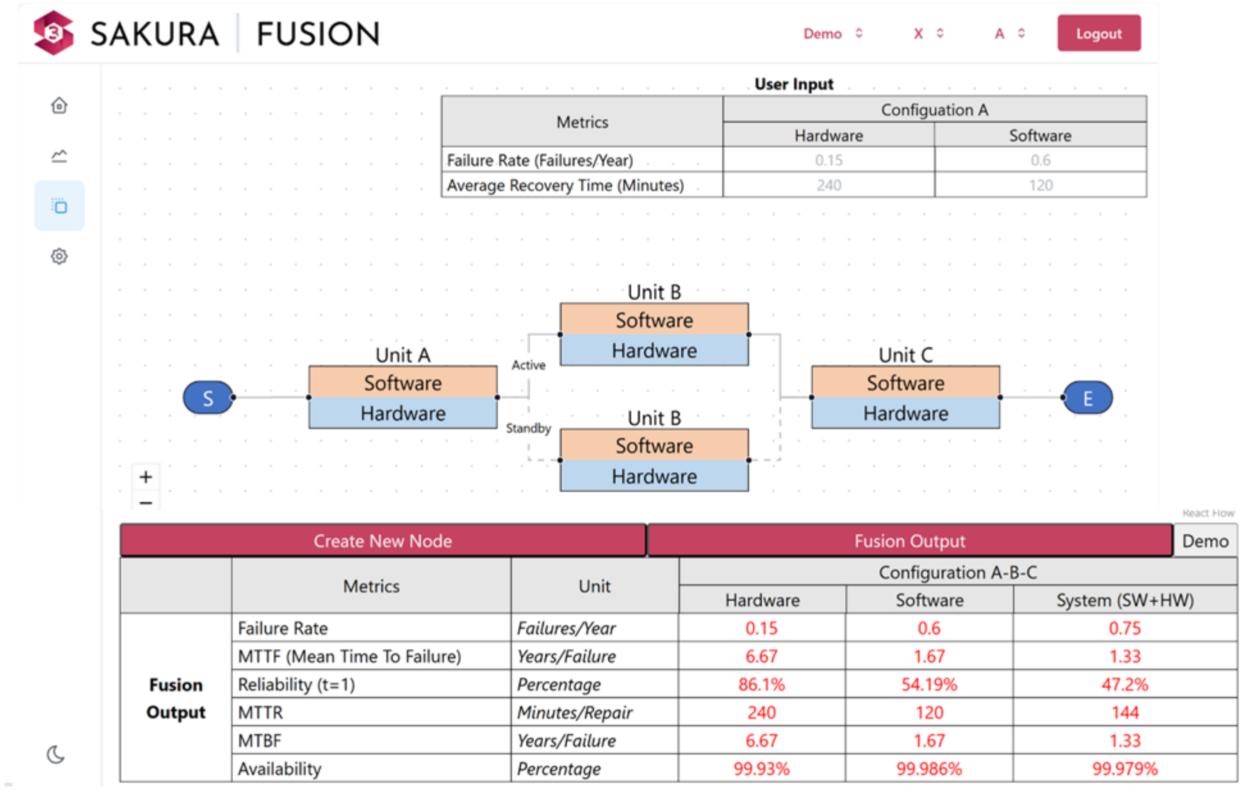
– Level 1: Iterate over γ from 0.86 to 1.2 with an increment of 0.01.

– Level 2: For each value of γ, iterate over β from 0.7 to 1.1 with an increment of 0.01.

– Level 3: For each combination of γ and β, iterate over α from 5 to 0 with a decrement of 0.1

• Calculate SSQ for each combination and track the minimum SSQ value:

– For a fixed value of γ, iterate through α and β. Terminate Level 3 if the SSQ exceeds the previous value.



User Input

Metrics	Configuation A	
	Hardware	Software
Failure Rate (Failures/Year)	0.15	0.6
Average Recovery Time (Minutes)	240	120

	Metrics	Unit	Configuration A-B-C		
			Hardware	Software	System (SW+HW)
Fusion Output	Failure Rate	*Failures/Year*	0.15	0.6	0.75
	MTTF (Mean Time To Failure)	*Years/Failure*	6.67	1.67	1.33
	Reliability (t=1)	*Percentage*	86.1%	54.19%	47.2%
	MTTR	*Minutes/Repair*	240	120	144
	MTBF	*Years/Failure*	6.67	1.67	1.33
	Availability	*Percentage*	99.93%	99.986%	99.979%

Fig. 2: A sample iGRED with the user input and the output

– Continue Level 2 until the minimum SSQ for the current γ is found.
– Repeat the process for γ until the global minimum SSQ is identified.

This algorithm provides a methodical approach to determining the best-fit parameters that minimize discrepancies between the predicted and actual software failure curves.

3.3 Intelligent Interactive Graphical Editor (iGRED) for Reliability Block Diagrams (RBDs)

iGRED, integrated into **FUSION**, allows users to construct detailed **reliability block diagrams (RBDs)** for software and hardware systems. Built-in real-time reliability metric calculations simplify assessments and accelerate system design. RBDs visually represent system configurations, enabling reliability analysis through **series and parallel equations**. iGRED also provides **preconfigured templates** for single-unit, active-standby, and active-active systems.
Users input **failure rates** and **recovery times**, and upon saving, iGRED instantly calculates and displays reliability metrics. Figure 2 shows a sample iGRED with the user input and the Fusion output.

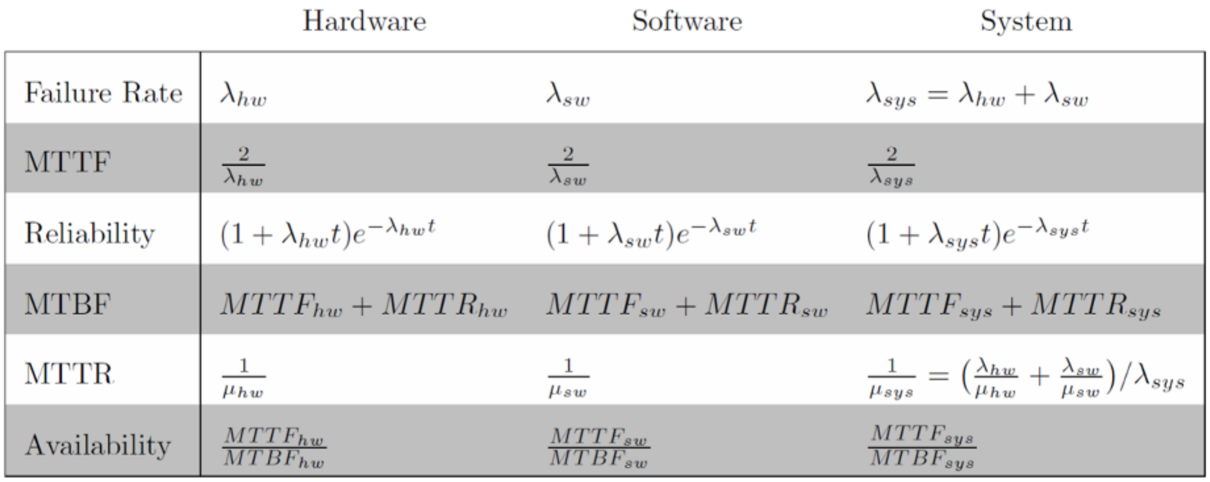
3.3.1 Reliability Metrics

Reliability metrics evaluate system performance and operational efficiency. Key metrics include failure rate, Mean Time to Failure (MTTF), Mean Time Between Failures (MTBF), Mean Time to Repair (MTTR), and availability.

Failure Rate (λ): The failure rate measures how often a system or component fails over a specified time, expressed as failures per unit of time (e.g., per hour or year). It is crucial in reliability engineering to predict system reliability. A low failure rate indicates high reliability, while a high rate suggests frequent failures. Failure rates can be constant (common in electronics during the useful life period) or time-varying (changes over time, seen during early failures or wear-out phases).

Reliability Function (R(t)): The probability that a system operates without failure over time t. For a constant failure rate, it follows an exponential distribution.

MTTF (Mean Time to Failure): MTTF measures the average operational lifespan of non-repairable systems before failure. Higher MTTF indicates greater reliability. It applies to items like light bulbs or batteries.



	Hardware	Software	System
Failure Rate	λ_{hw}	λ_{sw}	$\lambda_{sys} = \lambda_{hw} + \lambda_{sw}$
MTTF	$\frac{2}{\lambda_{hw}}$	$\frac{2}{\lambda_{sw}}$	$\frac{2}{\lambda_{sys}}$
Reliability	$(1+\lambda_{hw}t)e^{-\lambda_{hw}t}$	$(1+\lambda_{sw}t)e^{-\lambda_{sw}t}$	$(1+\lambda_{sys}t)e^{-\lambda_{sys}t}$
MTBF	$MTTF_{hw} + MTTR_{hw}$	$MTTF_{sw} + MTTR_{sw}$	$MTTF_{sys} + MTTR_{sys}$
MTTR	$\frac{1}{\mu_{hw}}$	$\frac{1}{\mu_{sw}}$	$\frac{1}{\mu_{sys}} = \left(\frac{\lambda_{hw}}{\mu_{hw}} + \frac{\lambda_{sw}}{\mu_{sw}}\right)/\lambda_{sys}$
Availability	$\frac{MTTF_{hw}}{MTBF_{hw}}$	$\frac{MTTF_{sw}}{MTBF_{sw}}$	$\frac{MTTF_{sys}}{MTBF_{sys}}$

Fig. 3: Sample reliability metrics formulas for a two-unit active-active configuration

MTBF (Mean Time Between Failures): MTBF measures the average time between failures for repairable systems. It includes MTTF (Time before failure) and **MTTR (Time to Repair)**, which is the Time needed to restore functionality after failure. MTBF = MTTF + MTTR. Higher MTBF values indicate greater reliability, aiding maintenance planning and uptime estimation.

Availability (A): Availability reflects the proportion of operational time versus expected time. Calculated as **A = Uptime / (Uptime + Downtime)** or **A = MTTF / MTBF.** High availability (close to 1) indicates minimal downtime and is essential for critical industries like telecommunications, healthcare, and military applications. It supports **System Design (**ensuring systems meet operational requirements), **Maintenance Planning (**optimizing schedules to maximize uptime), and **SLAs (**defining reliability standards such as “99.9% uptime”).

These metrics guide reliability engineering, enabling better design, maintenance, and operational strategies for minimizing failures and downtime.

3.3.2 Templates for Reliability Block Diagrams

Phase I iGRED development focuses on three primary configurations: A single-unit system with software and hardware, a two-unit active-standby system, and a two-unit active-active system. Phase II will address additional configurations.

FUSION calculates key reliability metrics, including **failure rate (**frequency of failures), **MTTF** (average operational lifespan before failure), **reliability** (percentage of systems expected to operate without failure over one year), **MTBF** (average time between successive failures, **MTTR** (average time required to restore functionality after a failure), and **availability** (proportion of operational time versus total expected time).

These metrics are computed for software, hardware, and the integrated system (HW + SW), providing a comprehensive overview of system reliability. Fig. 3 shows the reliability metrics formulas for a two-unit active-active template. Formulas for other templates have also been developed using the Markov chain method, although they are not shown here due to space limitations.

3.3.3 Combinations of the Basic Configurations / Templates

Each template requires specific starting and ending points to construct a network of templates. The network begins at the Start node and ends at the End node. Complex systems may combine series and parallel configurations. The reliability of such systems can be calculated by breaking them down into more straightforward series and parallel components and then combining their reliabilities using appropriate formulas. Sorting them topologically simplifies the metrics calculation. Fig. 2 shows an example with three configurations: two single-unit configurations and a two-unit active-active load-sharing configuration. These configurations are connected in series. Reliability metrics are already calculated for each template, allowing us to add them to determine the system failure rate. Figure 2 also shows the outputs for the sample configurations. FUSION will automatically calculate for any combinations of the templates.

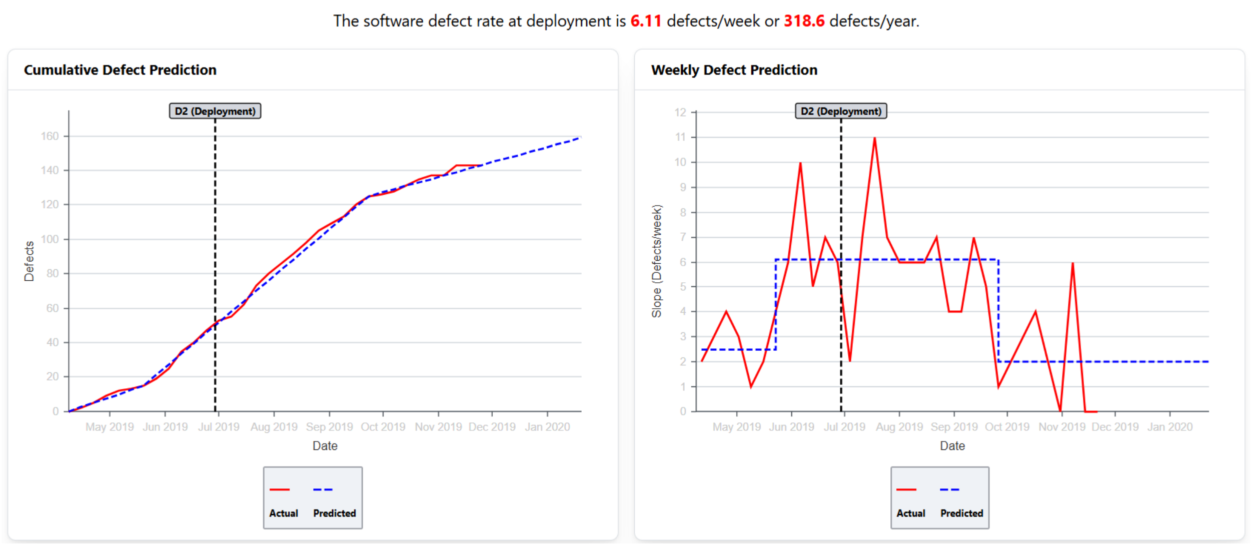

The software defect rate at deployment is **6.11** defects/week or **318.6** defects/year.

Fig. 4. A sample output of software defect prediction algorithm

3.4 System Design and Building

FUSION enables users to design and optimize reliable systems through advanced analytics and interactive modeling tools. Its software failure rate prediction component delivers accurate defect and failure forecasts, supporting proactive maintenance and system enhancements. With **iGRED's** intuitive interface and automated calculations, users can effortlessly create comprehensive **reliability block diagrams (RBDs)**, streamlining system evaluation and reliability improvement.

4. FUSION Implementation

FUSION is designed to streamline the entire workflow of reliability assessment, encompassing data extraction, pre-processing, core analytics, and post-processing. The system is built on the AWS platform, ensuring scalability and reliability. The architecture leverages several key components and technologies:

1. **Data Extraction and Pre-Processing**:
   - **APIs and Data Collection**: FUSION uses Flask APIs to collect data from various defect-tracking tools. This data is consolidated into two optimized databases: PostgreSQL for structured data and DynamoDB for unstructured data.
   - **Pre-Processing Steps**: Data undergoes rigorous pre-processing to ensure consistency and accuracy. This includes standardizing field and value mappings, categorizing field values, and detecting duplicates and cross-release defects. Data is aggregated into weekly or specified time frames to prepare inputs for the core analytics engine.
2. **Core Analytics Engine**:
   - **Predictive Modeling and Analytics**: Developed using the Python scientific stack, the core analytics engine performs advanced predictive modeling and reliability metrics calculation. It leverages libraries such as NumPy, SciPy, and scikit-learn for statistical analysis and machine learning.
   - **Reliability Block Diagrams (RBDs)**: The engine supports the creation and evaluation of RBDs, which visually represent system configurations and facilitate reliability analysis through series and parallel equations.
3. **User Interface**:
   - **Interactive and Intuitive Design**: The user interface is built with JavaScript ES6 and React, providing a responsive and intuitive experience. Users can design system reliability block diagrams, input failure rates and recovery times, and view real-time reliability metrics.
   - **Infrastructure Management**: Terraform AWS manages the system's infrastructure as code (IaC), ensuring efficient deployment and scalability.

5. Results and Discussion

Fig 4 illustrates these sequential lines, visually representing the algorithm's results. For enhanced clarity, the figure includes a weekly breakdown of the trend. The algorithm's

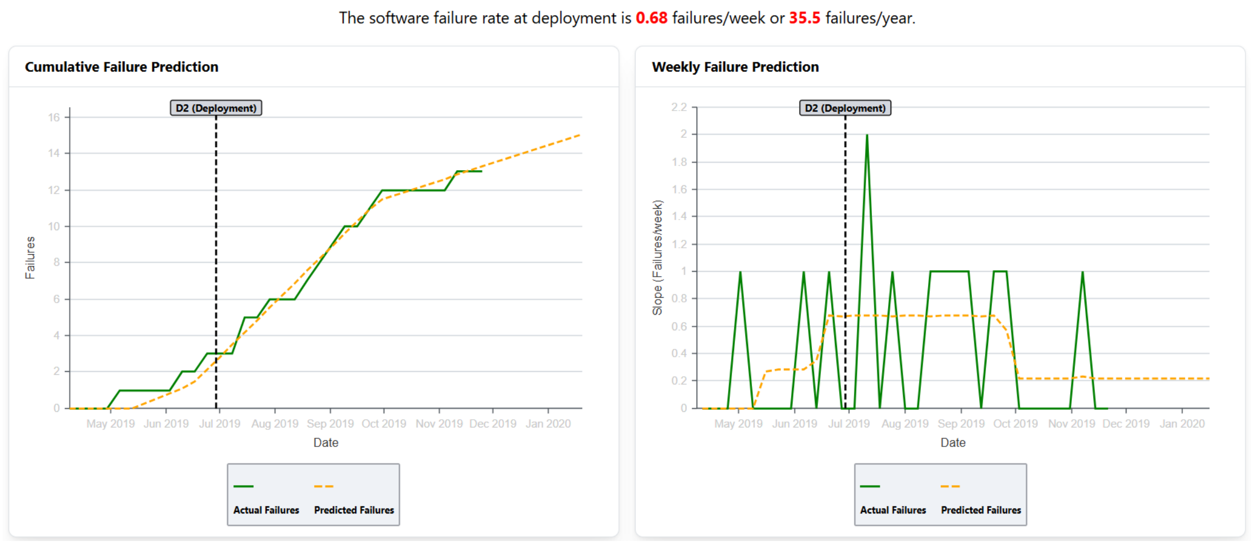

The software failure rate at deployment is 0.68 failures/week or 35.5 failures/year.

Fig. 5: A sample output of the software failure prediction algorithm.

utility extends beyond software defects, including hardware failure data. Additionally, we compare the observed hardware failure rates with those projected from the bill of materials (BOM) to deliver a comprehensive understanding of system reliability. This approach enables accurate modeling of defect trends and ensures timely and effective defect management throughout the software lifecycle. By integrating this capability, FUSION enhances reliability analysis for software and hardware components, providing actionable insights to improve system performance.

Fig 5 illustrates the predicted software failure curve alongside actual data, presented in cumulative and weekly views. These visualizations confirm that the software failure rate remains constant after deployment. This method has been applied to data from multiple projects, consistently validating these findings. A notable advantage of this approach is its effectiveness even when the number of failures is minimal, such as fewer than 10 occurrences. By integrating this method, FUSION enhances the accuracy and applicability of failure rate predictions, contributing to improved system reliability across diverse operational scenarios.

Figure 5: This figure showcases a sample configuration created using the Intelligent Interactive Graphical Editor (iGRED) integrated into FUSION. The figure includes a pop-up input table that appears when selecting a template, allowing users to input failure rates and recovery times. The RBD (Reliability Block Diagram) visually represents the system configuration, enabling reliability analysis through series and parallel equations. The figure demonstrates the use of preconfigured templates for single-unit, active-standby, and active-active systems. Once the RBD is completed, users can view real-time reliability metrics computed at the bottom of the page. These metrics include failure rate, Mean Time to Failure (MTTF), Mean Time Between Failures (MTBF), Mean Time to Repair (MTTR), and availability. iGRED simplifies the construction of detailed RBDs and accelerates system design by providing instant calculations and visual feedback, thereby enhancing the overall reliability assessment process.

6. Conclusion and Future Work

This paper introduced **FUSION**, a cloud-based tool that bridges a critical gap in operational-phase reliability modeling by aligning software and hardware reliability models. Our research shows that software failure rates during operation can be effectively modeled as segmented straight lines. With advanced analytics and a graphical editor, FUSION streamlines **reliability block diagram (RBD)** creation, accelerating reliability assessments and enabling targeted improvements. FUSION combines predictive analytics with an intuitive graphical interface, providing system reliability engineers with a powerful tool to anticipate and mitigate failures. This integrated approach enhances software and hardware reliability, ensuring overall system robustness.

By empowering practitioners to enhance system performance, FUSION significantly boosts reliability and customer satisfaction. This tool represents a significant advancement in reliability engineering, setting the stage for more resilient, integrated systems.

Acknowledgments

This research is supported by the National Science Foundation (NSF) Small Business Technology Transfer (STTR) program under Award Number 2348264. The author gratefully acknowledges Rashid Mijumbi, Joseph Good, Mike Okumoto, Andrew Raposo, Dylan Wong, and JiaXuan Mi for their valuable contributions to this paper.

References

[1] Ellison, Robert J. "Assuring software reliability." *Carnegie Mellon University, Tech. Rep.* (2014).

[2] Trivedi, Kishor. (2021). Reliability and Availability of Hardware-Software systems: Stochastic Reliability Models of Real Systems. 10.13140/RG.2.2.30286.48960.

[3] Sun, Bo, et al. "Physics-of-failure and computer-aided simulation fusion approach with a software system for electronics reliability analysis." *Eksploatacja i Niezawodność* 22.2 (2020). M. Dan, "The business model of 'Software-As-A-Service,'" in *Proc. 2007 IEEE Int. Conf. Services Comput.*, 2007, pp. 118-125.

[4] H. Pham, *Software Reliability*, Springer, Berlin, Heidelberg, 1999.

[5] P. D. T. O'Connor, "Software Reliability Engineering, John D. Musa, McGraw-Hill," *Quality and Reliability Engineering International*, vol. 16, no. 5, pp. 391-400, 2000.

[6] M. R. Lyu, *Handbook of Software Reliability Engineering*, McGraw-Hill, Inc., USA, 1996.

[7] D. D. Hanagal and N. N. Bhalerao, "Literature Survey in Software Reliability Growth Models," in *Software Reliability Growth Models*, Springer, Singapore, 2021, pp. 13-26.

[8] R. Mijumbi, K. Okumoto, and A. Asthana, "Software Reliability Assurance in Practice," in *Wiley Encyclopedia of Electrical and Electronics Engineering*, 2019.

[9] K. Okumoto, "Early Software Defect Prediction: Right-Shifting Software Effort Data into a Defect Curve," in *Proc. 2022 IEEE Int. Symp. Software Reliability Engineering Workshops (ISSREW)*, 2022, pp. 43-48.

[10] K. Okumoto, "Digital Engineering-driven Software Quality Assurance-as-a-Service," in *Proc. 28th ISSAT Int. Conf. Reliability and Quality in Design (RQD)*, 2023.

[11] X. Zhang and H. Pham, "Software field failure rate prediction before software deployment," *Journal of Systems and Software*, vol. 79, no. 3, pp. 43-56, 2006.

[12] K. Okumoto, "Experience Report: Practical Software Availability Prediction in Telecommunication Industry," in *Proc. Int. Symp. Software Reliability Engineering (ISSRE)*, 2016, pp. 19-25.

[13] M. Zhu and H. Pham, "A novel system reliability modeling of hardware, software, and interactions of hardware and software," *Mathematics*, vol. 7, no. 11, pp. 1049-1060, 2019.

[14] P. D. T. O'Connor and A. Kleyner, *Practical Reliability Engineering*, 5th ed., Wiley, 2011.

[15] U.S. Department of Defense, *Military Handbook MIL-HDBK-217E, Reliability Prediction of Electronic Equipment*, 1986.

[16] T. P. Ryan and W. Q. Meeker, "System Reliability Theory: Models, Statistical Methods, and Applications," *Journal of Quality Technology*, vol. 37, no. 1, pp. 5-10, 2005.

[17] W. Nelson, W. Q. Meeker, and L. A. Escobar, "Statistical Methods for Reliability Data," *Technometrics*, vol. 40, no. 3, pp. 293-297, 1998.

[18] A. Kleyner and V. Volovoi, "Application of Petri nets to reliability prediction of occupant safety systems with partial detection and repair," *Reliability Engineering and System Safety*, vol. 95, no. 6, pp. 123-130, 2010.

[19] A. Birolini, *Reliability Engineering: Theory and Practice*, 7th ed., Springer, 2014.

[20] Y. Jackson, P. Tabbagh, P. Gibson, and E. Seglie, "The new department of defense (DoD) guide for achieving and assessing RAM," in *Proc. Annual Reliability and Maintainability Symp.*, 2005, pp. 140-145.

[21] Relyence, "A Deep Dive into System Modeling Using Reliability Block Diagram (RBD) Analysis," Technical report, 2021.