From Residual Defects to Operational Failures: A Two-Stage Mapping Framework for Predictive Operational Risk

Kazuhira Okumoto
Sakura Software Solutions, LLC
Naperville, IL 60563

Abstract

Software reliability growth models have long been used to estimate residual defects during software testing and to support release readiness decisions. However, these models generally terminate at software release and do not provide a quantitative mechanism for predicting operational reliability after deployment. Consequently, organizations lack an effective method for transforming development-phase software quality measurements into operational reliability metrics.

This paper presents a quantitative methodology that establishes a direct relationship between development-phase residual defects and operational failure rates through two empirically calibrated transformation stages. The proposed **Two-Stage Mapping Framework** decomposes the prediction process into two measurable transformations: (1) mapping residual defects to annual customer-found defect rates and (2) mapping customer-found defect rates to annual operational failure rates. The resulting framework provides a direct quantitative mapping from development-phase residual defects to operational failure rates through empirically calibrated transformation coefficients.

The proposed methodology is supported by a statistical formulation and validated using industrial software development and operational datasets. The empirical results demonstrate stable relationships between residual defects, customer-found defect rates, and operational failure rates, enabling operational reliability to be estimated before software deployment. The framework is further integrated with the STAR (Software Testing Analysis and Reliability) platform and the FUSION operational reliability framework, providing a continuous software lifecycle reliability management approach that connects software testing, deployment, and operational monitoring.

By establishing a quantitative bridge between software quality assurance and operational reliability engineering, the proposed framework enables earlier operational risk assessment, more informed release decisions, reduced lifecycle software maintenance costs, and predictive software lifecycle management across modern software-intensive systems.

3S White Paper Series, WP-2026-002 | Version 1.0 | August 2026.

Keywords

Software Reliability, Residual Defects, Operational Failure Prediction, Operational Risk, Software Reliability Growth Models, Software Quality Assurance, Predictive Software Lifecycle Management

1. Introduction

Software reliability engineering has traditionally focused on estimating software quality during development through defect detection, reliability growth modeling, and residual defect estimation [3], [4], [5]. These techniques support release readiness decisions by helping development organizations determine whether software has achieved an acceptable level of quality before deployment. However, once software is released, engineering attention typically shifts from software quality metrics to operational reliability measures such as customer-found defects, operational failures, service availability, and system reliability.

Although operational failures ultimately originate from residual defects remaining in the delivered software, a quantitative relationship between development-phase software quality and operational reliability has not been widely established. Existing software reliability growth models generally terminate at software release and do not provide a mechanism for estimating operational failure rates using development-phase quality measurements. As a result, organizations lack a practical framework for predicting operational reliability before deployment.

This paper presents a **two-stage mapping framework** that quantitatively transforms residual defects into operational failure rates through an intermediate customer-found defect rate. The framework decomposes the prediction process into two independently measurable mappings, allowing development-phase quality measurements to be translated into operational reliability metrics. The resulting model provides a quantitative bridge between software quality assurance and operational reliability management.

The proposed framework is supported by a statistical formulation and validated using industrial software development and operational datasets. The empirical results demonstrate stable transformation relationships between residual defects and customer-found defect rates, and between customer-found defect rates and operational failure rates. Together, these mappings enable estimation of operational failure rates before deployment using measurable software quality indicators.

The proposed mapping methodology is further integrated with the STAR (Software Testing Analysis and Reliability) platform to extend development-phase software quality assessment into

operational reliability prediction, and with the FUSION operational reliability framework to establish a continuous software lifecycle reliability management process. This integration enables organizations to connect software testing, deployment, and operational monitoring within a unified predictive reliability framework.

The remainder of this paper is organized as follows. Section 2 discusses the importance of residual defects and the need for quantitative operational risk prediction. Section 3 introduces the proposed two-stage mapping framework, followed by the statistical foundation in Section 4 and empirical validation in Section 5. Sections 6 through 8 describe the operational cost implications and integration with the STAR and FUSION platforms. Section 9 discusses representative application domains, and Section 10 outlines future research directions.

2. Why Residual Defects Matter

No software system is completely free of defects at the time of deployment. Regardless of the testing methodology employed, practical constraints on schedule, budget, and test coverage inevitably leave a number of residual defects in the delivered software. These residual defects represent latent software faults that may become visible only after the system is placed into operational service.

During software testing, residual defects are commonly used as an indicator of release readiness and product quality. Software reliability growth models estimate the number of defects remaining at the completion of testing and assist project managers in determining whether additional testing is warranted before release. Consequently, residual defect estimation has become an established practice in software quality assurance.

After deployment, however, engineering attention shifts from software quality metrics to operational reliability measures, including customer-found defects, operational failures, service interruptions, and system availability. These operational measures are typically analyzed independently from the development process, resulting in a discontinuity between software quality engineering and operational reliability engineering.

In practice, not every residual defect becomes visible to customers, and not every customer-found defect results in an operational failure. Nevertheless, operational failures ultimately originate from residual defects that remain in the software at deployment. Residual defects therefore constitute the earliest measurable indicator of future operational risk.

The economic consequences of operational failures further emphasize the importance of residual defect estimation. According to an IBM Systems Sciences Institute study, defects discovered in production can cost **30 to 100 times more** to correct than defects identified during the early

phases of software development [2]—consequently, the ability to estimate operational reliability before deployment has significant technical and economic value.

Despite their importance, there is currently no widely accepted quantitative framework for transforming development-phase residual defect estimates into operational reliability metrics. Existing software reliability growth models [3], [5] primarily focus on defect discovery during testing and generally terminate their predictions at software release. As a result, organizations lack a practical mechanism for estimating operational reliability using information available before deployment.

This paper addresses this gap by establishing a quantitative relationship between residual defects and operational failure rate through a two-stage mapping framework. The proposed approach enables development-phase quality measurements to be translated into operational reliability metrics, providing a quantitative bridge between software quality assurance and operational reliability management.

3. The Two-Stage Mapping Framework

Software quality metrics collected during development are traditionally used to assess release readiness and estimate the number of residual defects remaining at deployment. Operational reliability, however, is evaluated using a different set of measures, including customer-reported defects, operational failures, availability, and service reliability. As a result, development quality and operational reliability have generally been treated as separate engineering disciplines with limited quantitative linkage.

The proposed framework establishes a bridge between these two phases by introducing a two-stage mapping process that transforms development-phase quality measurements into operational reliability metrics. Rather than attempting to estimate operational failures directly from residual defects, the framework decomposes the relationship into two sequential transformations.

The first mapping relates the estimated residual defects remaining at deployment to the annual customer-found defect rate. This stage represents the activation of residual defects that become visible during operational use. Only a portion of the residual defects remaining after testing are expected to be encountered by customers during normal operation.

The second mapping relates the customer-found defect rate to the operational failure rate. This stage captures the observation that not every customer-reported defect results in an operational failure. Some defects have only minor functional impact, while others are corrected before affecting system operation. Consequently, operational failures represent a measurable subset of customer-found defects.

Figure 3.1 illustrates the overall mapping framework. The two transformation stages provide an intermediate observable variable—customer-found defect rate—that connects software quality measured during development with operational reliability measured after deployment. This decomposition enables each transformation to be independently calibrated and validated using empirical data.

Unlike conventional software reliability growth models, which typically terminate at software release, the proposed framework extends prediction beyond deployment by linking software quality to operational reliability through measurable transformation stages. This provides a quantitative mechanism for estimating operational failure rates before deployment using information available during software testing.

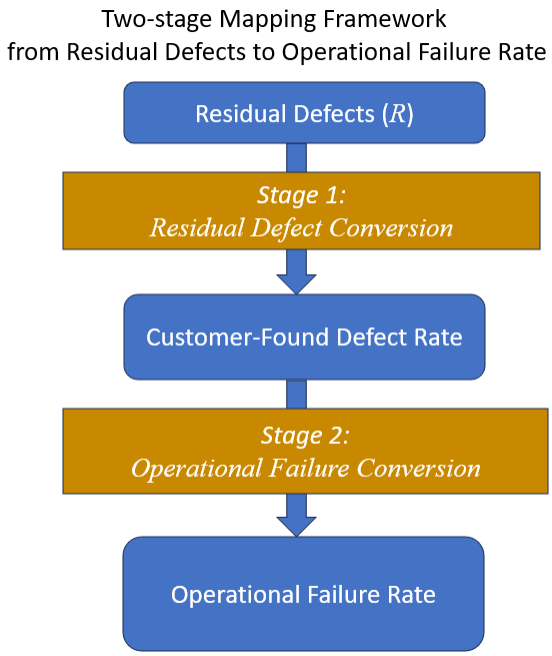

Figure 3.1. Two-stage mapping framework from residual defects to operational failure rate.

4. Statistical Foundation

The proposed mapping framework is based on the observation that the progression from development-phase residual defects to operational failures can be represented as a sequence of measurable stochastic transformations. Rather than attempting to model the relationship directly, the framework decomposes the process into two statistically independent mappings that can be estimated using industrial software development and operational data.

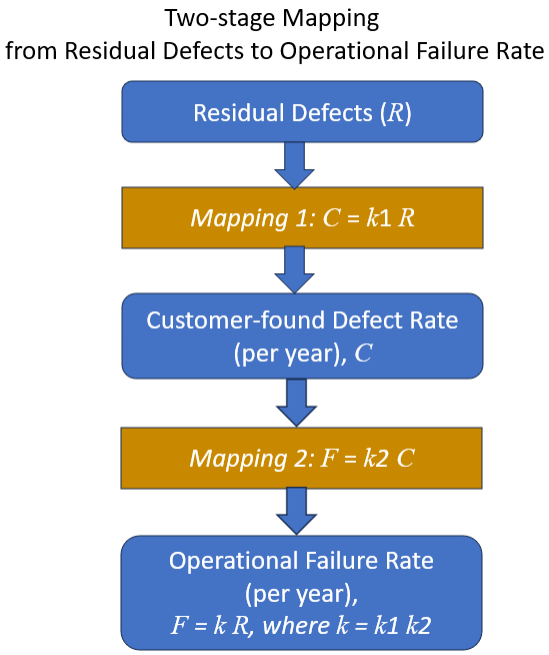

Figure 4.1. Two-stage mapping framework from residual defects to operational failure rate.

This mapping assumes that only a fraction of residual defects becomes visible to customers after deployment. The coefficient k_1 therefore represents the average activation probability of residual defects under similar development and operational environments.

The second transformation is modeled as

$$F=k_2C, \tag{4.2}$$

where k_2 is the **Operational Failure Conversion Ratio**.

This mapping recognizes that not every customer-found defect results in an operational failure. Many defects are corrected before they propagate further, while others have negligible operational consequences. The coefficient $k2$ therefore represents the average escalation probability from customer-found defects to operational failures.

Combining the two transformations yields

$$F=kR, \tag{4.3}$$

where

$$k=k_1k_2. \tag{4.4}$$

The coefficient k is referred to as the **Overall Operational Risk Mapping Coefficient**. It represents the expected operational failure rate associated with a unit residual defect remaining at deployment.

Unlike conventional software reliability growth models [3], [4], [6], which typically estimate defect discovery during testing, the proposed formulation explicitly bridges software quality metrics collected before deployment with operational reliability metrics observed after deployment. The decomposition into two measurable transformation coefficients enables each stage to be independently calibrated, validated, and updated as additional operational data become available.

The framework does not require every individual defect to follow an identical progression. Instead, it assumes that, when aggregated across multiple releases developed under comparable engineering processes, the average transformation behavior remains sufficiently stable that the coefficients k_1 and k_2 can be estimated empirically. This statistical assumption provides the foundation for predicting operational failure rates from development-phase software quality measurements.

5. Empirical Validation

The proposed framework was evaluated using industrial software development and operational datasets. The objective was to determine whether residual defects remaining at deployment can be quantitatively transformed into an operational failure rate through a sequence of empirically validated mappings.

5.1 Mapping Residual Defects to Customer-Found Defect Rate

Let

- R denote the **estimated residual defects remaining at deployment**, and
- C denote the **customer-found defect rate**, expressed as the annual number of customer-found defects.

The first mapping is defined as

$$C=k_1R \quad (5.1)$$

where

- k_1 is the **Residual Defect Conversion Rate**, representing the annual customer-found defect rate generated by each residual defect remaining at deployment.

Figure 5-1 illustrates the relationship between the estimated residual defects at deployment and the corresponding annual customer-found defect rates observed across multiple software releases. To provide a consistent basis for comparison, the original customer-found defects collected over a 20-month observation period were normalized to an equivalent one-year observation interval.

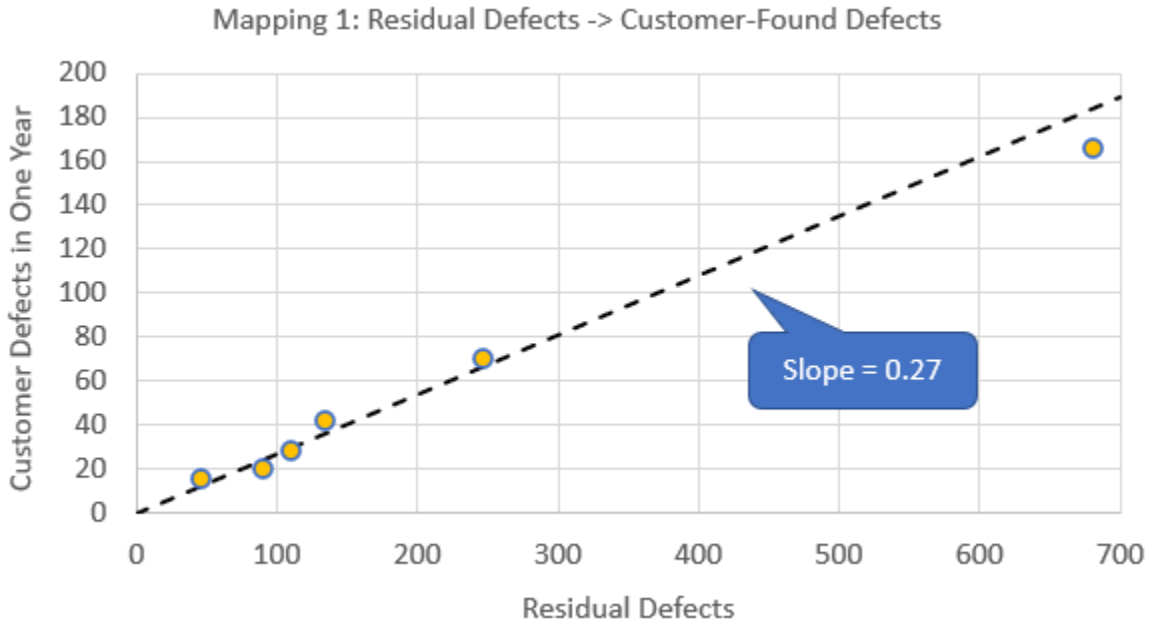

Figure 5.1 Mapping 1: Residual Defects -> Customer-found Defects.

The scatter plot exhibits a strong, approximately linear relationship. The fitted regression line has a slope of approximately

$$k_1 \approx 0.27. \quad (5.2)$$

This result indicates that the annual customer-found defect rate is approximately **27% of the residual defects remaining at deployment**. Despite variations in project size and residual

defect counts, the proportional relationship remains stable, demonstrating that residual defects provide a reliable predictor of post-release customer-visible quality.

5.2 Mapping Customer-Found Defect Rate to Operational Failure Rate

Let

- F denote the **operational failure rate**, expressed as the annual number of operational failures.

The second mapping is defined as

$$F=k_2C \tag{5.3}$$

where

- k_2 is the **Operational Failure Conversion Ratio**, representing the fraction of customer-found defects that escalate into operational failures.

Figure 5-2 compares the customer-found defect rate with the corresponding operational failure rate observed after deployment. Applying the proposed transformation produces an operational failure trajectory that closely follows the observed failure data. The calibrated model estimates an average operational failure rate equivalent to approximately **0.56 failures per week** while reducing prediction error by approximately **9.2%**.

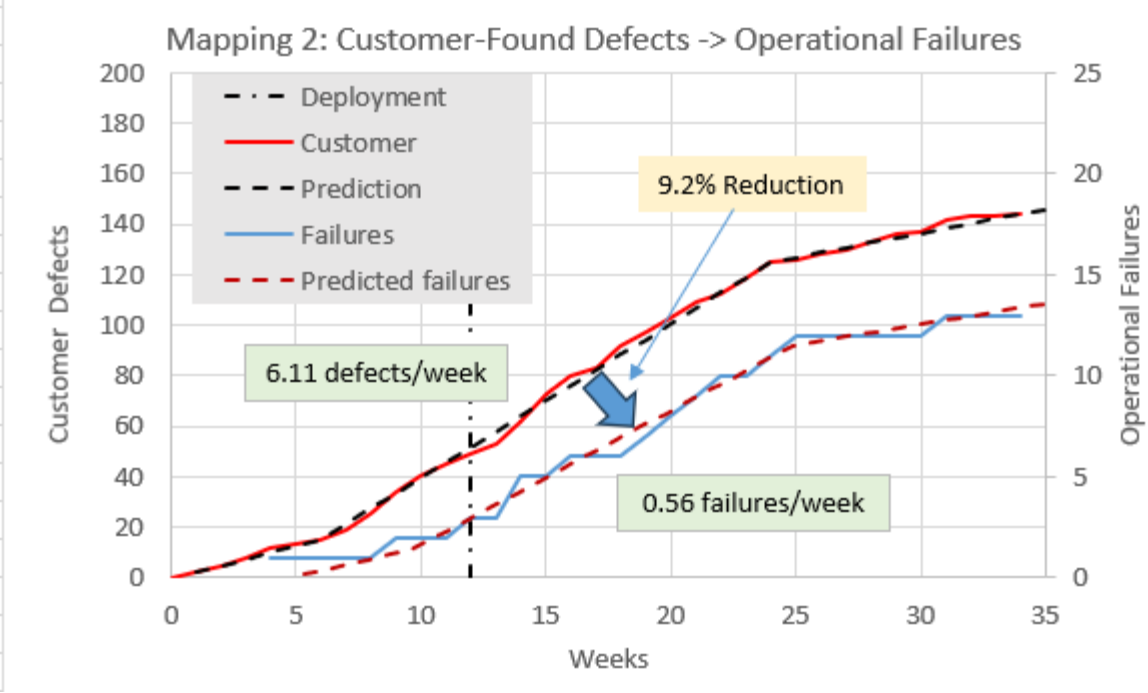

Figure 5.2 Mapping 2: Customer-found Defects -> Operational Failures.

The corresponding transformation coefficient is estimated as

$$k_2 \approx 0.092. \tag{5.4}$$

This result indicates that approximately **9.2% of customer-found defects ultimately manifest as operational failures**. The close agreement between the predicted and observed failure trajectories demonstrates that operational failures are the downstream manifestation of a measurable subset of customer-found defects.

5.3 Overall Mapping

Combining Equations (5.1) and (5.3) yields

$$F = kR \tag{5.5}$$

where

$$k = k_1 k_2. \tag{5.6}$$

Table 5.1 summarizes the coefficients.

Table 5.1

Coefficient	Definition	Value
(k_1)	Residual Defect Conversion Rate	0.27
(k_2)	Operational Failure Conversion Ratio	0.092
($k=k_1k_2$)	Overall Operational Risk Mapping Coefficient	0.0248

The coefficient k is referred to as the **Overall Operational Risk Mapping Coefficient**, directly relating the residual defects remaining at deployment to the operational failure rate observed after deployment.

Because the two transformation coefficients are independently measurable, the overall mapping coefficient can be estimated directly from empirical data as the product of the two mappings. This decomposition provides a quantitative bridge between development-phase software quality and operational reliability.

Unlike conventional software reliability models that terminate at software release, the proposed framework extends prediction into the operational phase through two empirically validated transformation stages. Consequently, residual defects become more than a software quality metric—they become a measurable leading indicator of operational failure risk, enabling earlier reliability assessment, proactive corrective actions, and more informed release decisions.

6. Operational Cost Implications

Residual defects are often regarded as an internal software quality metric whose primary purpose is to support release readiness decisions. The proposed two-stage mapping framework demonstrates that residual defects also serve as a quantitative predictor of operational failure rate. This capability has important economic implications because the cost of correcting software defects increases dramatically after deployment.

According to the IBM Systems Sciences Institute, defects discovered during production operation can cost **30 to 100 times more** to correct than defects identified during the early phases of software development [2]. These additional costs arise from customer support activities, operational troubleshooting, emergency software patches, service interruptions, regression testing, and deployment of corrective updates.

Figure 6.1 illustrates how the proposed framework transforms residual defect estimation into a predictive operational risk management process. By enabling corrective actions before

deployment, organizations can reduce residual defects while avoiding the substantially higher costs associated with correcting defects during operation.

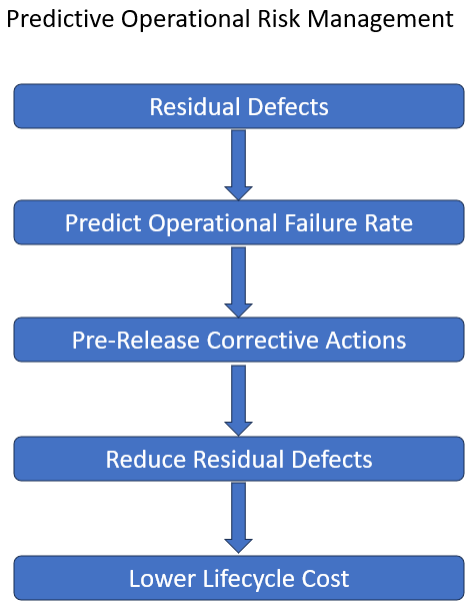

Figure 6.1. Predictive Operational Risk Management Framework

The proposed framework enables organizations to estimate the operational failure rate before software release by transforming estimated residual defects into operational reliability metrics. Rather than waiting for customer-reported defects or operational failures to reveal software quality problems, development organizations can quantitatively assess operational risk while corrective actions remain significantly less expensive.

This capability supports more informed release decisions. When the predicted operational failure rate exceeds an acceptable threshold, project managers may elect to extend testing, improve defect removal, postpone deployment, or implement targeted corrective actions before release. Conversely, projects with acceptably low predicted operational failure rates may proceed with greater confidence, reducing unnecessary testing and shortening release cycles.

The framework therefore shifts operational reliability management from a reactive process to a predictive process. By linking development-phase software quality measurements with operational reliability outcomes, organizations can reduce operational risk while minimizing lifecycle software maintenance costs.

Beyond direct cost savings, earlier prediction of operational failures can improve customer satisfaction, increase system availability, reduce emergency maintenance activities, and support

more effective allocation of software quality assurance resources. The economic value of the framework therefore extends beyond defect reduction to encompass the overall operational resilience of software-intensive systems.

7. Integration with STAR

The proposed two-stage mapping framework is designed to operate in conjunction with existing software quality assessment methods rather than replace them. In particular, it complements the STAR (Software Testing Analysis and Reliability) platform by extending its development-phase quality predictions into operational reliability assessment [1], [5], [17].

STAR estimates the number of residual defects remaining at deployment using software reliability growth modeling and release readiness analysis. These residual defect estimates have traditionally been used to evaluate software quality, determine release readiness, and support management decisions regarding additional testing before deployment. However, the estimated residual defects have generally represented the endpoint of the software quality assessment process.

The proposed framework extends the value of STAR by transforming the estimated residual defects into predicted customer-found defect rates and operational failure rates. Using the two-stage mapping model developed in this paper, the residual defect estimate generated by STAR becomes the input to a quantitative operational reliability prediction.

The integration is illustrated conceptually in Figure 7.1. STAR provides the estimated residual defects at deployment, which are subsequently transformed into customer-found defect rates and operational failure rates using the mapping coefficients k_1 and k_2. The resulting operational failure rate provides project managers with an estimate of operational risk before software release.

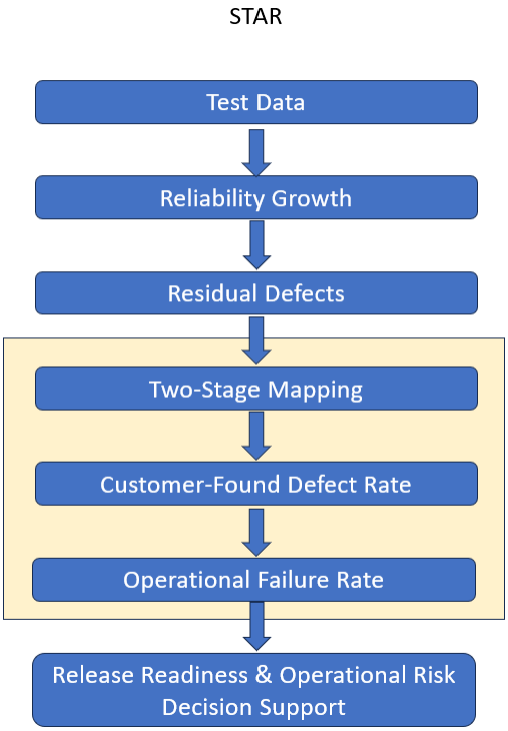

Figure 7.1. Integration of the proposed two-stage mapping framework with the STAR software quality assessment platform.

This integration expands the role of residual defect estimation from a development quality metric to a predictive operational reliability indicator. Rather than evaluating release readiness solely in terms of remaining defects, organizations can also evaluate the expected operational consequences of those defects. Consequently, release decisions can incorporate both software quality and predicted operational reliability.

The proposed integration also establishes a continuous connection between software testing and operational performance. As additional operational data become available after deployment, the mapping coefficients can be recalibrated, allowing future STAR analyses to produce progressively more accurate operational reliability predictions. This creates a feedback mechanism through which operational experience continuously improves future release assessments.

By combining development-phase software quality assessment with operational reliability prediction, STAR evolves from a release readiness decision-support tool into a predictive software lifecycle management platform capable of supporting engineering decisions throughout both development and operational phases

8. Integration with FUSION

While STAR provides quantitative assessment of software quality during development, FUSION extends reliability management into the operational phase [15]. Together, the two platforms establish a continuous software lifecycle framework that connects development-phase quality assessment with operational reliability management.

The proposed two-stage mapping framework provides the quantitative interface between these two environments. STAR estimates the residual defects remaining at deployment, and the mapping model transforms those estimates into predicted customer-found defect rates and operational failure rates. These predicted operational reliability metrics become the initial reliability baseline for FUSION immediately after software deployment.

During system operation, FUSION continuously analyzes customer-found defects, operational failures, hardware reliability, and integrated system reliability. As operational data accumulate, the observed customer defect and failure trends can be compared with the predicted values produced by the mapping framework. Differences between predicted and observed behavior provide feedback for recalibrating the mapping coefficients and improving future prediction accuracy.

This closed-loop process enables development and operational reliability engineering to function as a unified discipline rather than as independent activities. Development organizations can continuously refine residual defect estimation using operational experience, while operational organizations benefit from increasingly accurate reliability predictions generated before deployment.

Figure 8.1 illustrates the integration of STAR, the proposed two-stage mapping framework, and FUSION. The framework establishes a continuous flow of reliability information from software testing through operational deployment, creating a feedback mechanism that supports continuous improvement across multiple software releases.

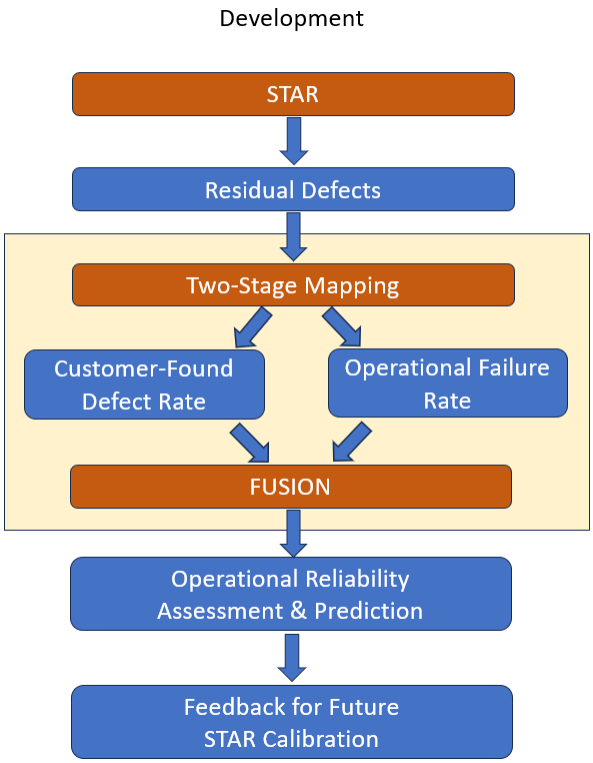

Figure 8.1. Integration of STAR, the proposed two-stage mapping framework, and FUSION for continuous software lifecycle reliability management.

The integrated framework also provides a foundation for predictive software lifecycle management. Instead of treating software testing and operational reliability as separate engineering activities, organizations can manage reliability throughout the complete software lifecycle using a common set of measurable quality and reliability indicators. This unified approach supports more accurate release decisions, improved operational planning, and continuous enhancement of software reliability.

9. Applications to DevOps, Telecom, Cloud, AI Systems

The proposed two-stage mapping framework applies to a broad range of software-intensive systems in which development-phase software quality influences operational reliability. Because the framework establishes a quantitative relationship between residual defects and operational failure rate, it provides a common reliability prediction methodology that is independent of a particular application domain.

9.1 DevOps and Continuous Delivery

Modern DevOps environments emphasize rapid software delivery through continuous integration and continuous deployment (CI/CD) [10], [11], [12]. Although these practices significantly shorten release cycles, they also reduce the amount of time available for comprehensive system testing. Consequently, development organizations require objective measures of release readiness that extend beyond traditional software quality metrics.

The proposed framework enables development teams to estimate operational failure rates before deployment by transforming residual defect estimates into predicted operational reliability metrics. This capability supports risk-based release decisions in high-frequency deployment environments, allowing organizations to balance deployment velocity with operational reliability.

9.2 Telecommunications and Cloud-Native Systems

Large-scale telecommunications and cloud-native software platforms require extremely high service availability and operational resilience. Even relatively small software defects may propagate into service disruptions affecting thousands or millions of users.

By estimating operational failure rates before deployment, the proposed framework provides operators with an early assessment of operational risk. Combined with continuous operational monitoring, the framework supports proactive reliability management for cloud-native applications, distributed microservices, Kubernetes-based platforms, Open RAN systems, and other highly distributed software infrastructures [13], [14].

9.3 AI-Assisted Software Engineering

The increasing use of artificial intelligence in software development introduces new opportunities for predictive software quality management. AI techniques can assist in estimating mapping coefficients, detecting changes in software behavior, identifying anomalous operational trends, and recommending corrective actions based on historical operational data.

Rather than replacing statistical reliability models, AI complements the proposed framework by improving parameter estimation and supporting adaptive model calibration as additional operational information becomes available. The combination of statistical reliability modeling and AI-assisted learning enables progressively more accurate operational reliability prediction over successive software releases.

9.4 Digital Engineering and Predictive Lifecycle Management

The proposed framework provides a quantitative foundation for integrating software quality engineering with operational reliability engineering throughout the software lifecycle. By linking development measurements with operational performance, organizations can establish closed-loop reliability management processes that continuously improve prediction accuracy using operational feedback.

This capability supports digital engineering initiatives in which software quality, operational reliability, maintenance planning, and lifecycle risk management are treated as components of an integrated decision-support framework. Consequently, the proposed mapping approach serves as a foundation for predictive software lifecycle management across diverse software-intensive systems.

10. Future Research

The two-stage mapping framework presented in this paper establishes an initial quantitative bridge between development-phase software quality and operational reliability. Although the empirical results demonstrate the feasibility of the proposed approach, several research directions remain for extending its applicability and predictive capability.

10.1 Validation Across Diverse Software Systems

The empirical validation presented in this paper is based on representative industrial software projects. Future research should evaluate the proposed mapping framework using additional datasets collected from multiple organizations, application domains, and software development methodologies. Broader validation will improve confidence in the stability and generality of the mapping coefficients across diverse operational environments.

10.2 Adaptive Estimation of Mapping Coefficients

The current framework estimates the transformation coefficients from historical project data. Future work will investigate adaptive estimation techniques that continuously recalibrate the mapping coefficients using operational feedback. Such adaptive models will enable the framework to respond to evolving development practices, changing software architectures, and varying operational environments while maintaining prediction accuracy over successive software releases.

10.3 AI-Assisted Reliability Prediction

Artificial intelligence offers significant opportunities to enhance the proposed framework. Future research will investigate AI-assisted estimation of mapping coefficients, automated detection of

changes in reliability behavior, anomaly detection, and intelligent recommendation of pre-release corrective actions. Combining statistical reliability modeling with AI-assisted learning has the potential to improve prediction accuracy while preserving the interpretability of the underlying engineering model.

10.4 Extension to Modern Software Architectures

Future research will extend the mapping framework to emerging software development environments, including cloud-native systems, microservice architectures, Kubernetes-based platforms, Open RAN networks, and AI-enabled software systems. These environments introduce additional sources of operational complexity that may influence the relationship between development-phase software quality and operational reliability.

10.5 Toward Predictive Software Lifecycle Management

The long-term objective of this research is to establish a unified predictive software lifecycle management framework that continuously links software development, deployment, operation, and maintenance through measurable reliability indicators. By integrating residual defect estimation, operational failure prediction, continuous operational monitoring, and adaptive feedback, organizations can transition from reactive software maintenance to predictive reliability engineering across the entire software lifecycle.

The two-stage mapping framework presented in this paper represents an initial step toward this vision by providing a quantitative foundation for connecting software quality assurance with operational reliability management. Continued research will further strengthen this foundation and expand its applicability to next-generation software-intensive systems.

11. Conclusion

This paper introduced a quantitative two-stage mapping framework that transforms development-phase residual defects into operational failure rates through empirically calibrated transformation coefficients. By establishing a measurable bridge between software quality assurance and operational reliability, the framework enables organizations to estimate operational risk before deployment, support more informed release decisions, and reduce lifecycle software maintenance costs. The proposed approach provides a foundation for predictive software lifecycle management and offers a practical direction for integrating software reliability engineering across development and operations.

References

[1] K. Okumoto and A. L. Goel, "Optimum release time for software systems based on reliability and cost criteria," *Journal of Systems and Software*, vol. 1, no. 4, pp. 315–318, 1980.

[2] IBM Systems Sciences Institute, *The Cost of Defects in Software Development*, IBM Corp., 2010.

[3] J. D. Musa, *Software Reliability Engineering*, McGraw-Hill, New York, 1998.

[4] M. R. Lyu (Ed.), *Handbook of Software Reliability Engineering*, McGraw-Hill and IEEE Computer Society Press, 1996.

[5] A. L. Goel and K. Okumoto, "Time-dependent error-detection rate model for software reliability and other performance measures," *IEEE Transactions on Reliability*, vol. R-28, no. 3, pp. 206–211, Aug. 1979.

[6] H. Pham, *System Software Reliability*, Springer, London, 2006.

[7] H. Pham, *Software Reliability*, Springer, Singapore, 2000.

[8] N. E. Fenton and J. Bieman, *Software Metrics: A Rigorous and Practical Approach*, 3rd ed., CRC Press, Boca Raton, FL, 2014.

[9] G. A. Lewis, *Microservices Architecture Patterns for DevOps*, Addison-Wesley, 2021.

[10] J. Humble and D. Farley, *Continuous Delivery: Reliable Software Releases through Build, Test, and Deployment Automation*, Addison-Wesley, 2011.

[11] G. Kim, J. Humble, P. Debois, and J. Willis, *The DevOps Handbook*, 2nd ed., IT Revolution Press, Portland, OR, 2021.

[12] N. Forsgren, J. Humble, and G. Kim, *Accelerate: The Science of Lean Software and DevOps*, IT Revolution Press, Portland, OR, 2018.

[13] The Linux Foundation, *Kubernetes Documentation*, 2025.

[14] O-RAN Alliance, *O-RAN Architecture Description*, latest release.

[15] K. Okumoto, "FUSION: A Cloud-Based Framework for Integrated Software and Hardware Reliability Assessment and Prediction," Proceedings of ISSAT Reliability and Quality in Design Symposium (RQD), 2025.

[16] K. Okumoto, "Adaptive Cross-Release Reliability Modeling," to appear in Proceedings of the Annual Reliability and Maintainability Symposium (RAMS), 2027.

[17] K. Okumoto, “Early Software Defect Prediction: Right-Shifting Software Effort Data into a Defect Curve,” Proc. 2022 IEEE Int. Symp. on Software Reliability Engineering Workshops (ISSREW), Charlotte, NC, USA, Oct. 2022, pp. 43–48.