ADM: An Agile Development Manager for Early Defect Prediction in Small Software Teams

Kazuhira Okumoto
Sakura Software Solutions, LLC
Naperville, IL 60563, USA

***Abstract*—Small, agile, and CI/CD software teams operate in short cycles, with sparse data and limited quality assurance resources, making early quality assessment and defect prediction particularly difficult. Traditional reliability and defect models, developed for larger, more stable projects, often fail to provide timely or reliable insights in these environments. This paper introduces the Agile Development Manager (ADM), a lightweight analytics module that extends the STAR quality-intelligence platform to small-scale projects. ADM applies segmented linear modeling over short rolling windows to capture evolving defect trends and provide early, stability-aware predictions with minimal data requirements. Experimental results using real project data show that ADM enables a 46% reduction in the time required to achieve stable prediction. This result demonstrate that ADM not only accelerates stabilization in absolute days but also significantly reduces the proportion of the lifecycle required to obtain reliable predictive confidence. ADM supports early risk visibility, improves prediction behavior, and enables proactive corrective actions in fast-cycle development environments.**

***Index Terms*—Agile software analytics, defect prediction, lightweight reliability modeling, CI/CD quality management, software quality intelligence**

I. Introduction

Agile and continuous integration/continuous delivery (CI/CD) practices [1]–[3] have reshaped modern software development by enabling rapid iteration, frequent releases, and close alignment with customer needs. Small development teams, in particular, rely on short development cycles and automated pipelines to deliver functionality quickly and respond to change. However, these same characteristics complicate quality management. Sparse defect data, compressed schedules, and limited access to dedicated quality assurance resources often force teams into reactive modes, where defects are addressed only after they accumulate and begin to threaten delivery commitments.

Traditional software reliability and defect prediction models [4]–[15] were primarily developed for environments with longer development phases, stable testing periods, and higher defect volumes. These assumptions rarely hold in small agile projects, where defect arrivals are irregular, testing is continuous, and system behavior evolves rapidly as features are integrated. As a result, conventional models often struggle to converge, produce unstable predictions, or deliver signals too late to support effective corrective action. Unlike NHPP-based SRGMs or deep neural cross-project models [15], ADM is explicitly optimized for low-volume, short-window, intra-project incremental adaptation without parameter-heavy estimation.

Recent research and industrial practice increasingly emphasize lightweight, incremental analytics capable of operating under sparse data conditions while still providing early, actionable insight. In particular, there is growing demand for short-window modeling, continuous trend assessment, and early indicators of instability that can support day-to-day agile decision-making. However, most existing tools emphasize descriptive dashboards rather than predictive, adaptive modeling.

To address this gap, this paper introduces the Agile Development Manager (ADM), a lightweight analytics module that extends the STAR quality-intelligence platform [16]–[18] to small-scale agile and CI/CD projects. ADM is designed to provide early defect-trend visibility, short-horizon prediction, and stability-aware modeling without imposing heavy data or process overhead. By applying segmented linear modeling over small, rolling time windows, ADM captures evolving defect dynamics and supports continuous assessment of project health. This paper presents ADM's methodology, demonstrates its behavior through experimental results, and discusses how lightweight prediction can support proactive quality management in fast-cycle software development environments.

The main contributions of this paper are:

- A formal segmented rolling-window defect modeling framework
- A lightweight adaptive inflection detection mechanism
- Working-day normalization analysis
- Empirical validation under sparse data conditions

II. State of the Art

Software defect prediction and reliability estimation have been studied extensively in software engineering. Early research focused on statistical *software reliability growth models* (SRGMs), which characterize the defect discovery process during testing. Classical models such as the Jelinski–Moranda model [4] and the Goel–Okumoto non-homogeneous Poisson process (NHPP) model [6] describe reliability growth as faults are progressively detected and removed. These models have been widely applied to estimate remaining defects and assess release readiness. Subsequent extensions incorporated imperfect debugging, delayed fault detection, and multiple change points to better represent evolving development behavior [10]. Execution-time-based formulations, such as the Musa–Okumoto logarithmic Poisson model [8], further improved accuracy by aligning reliability estimation with actual execution effort rather than calendar time.

Despite their theoretical strengths, traditional SRGMs rely on assumptions that are often difficult to satisfy in modern development environments. In particular, they typically require relatively stable testing phases, sufficient defect data for parameter estimation, and clearly observable reliability growth patterns. These assumptions rarely hold in small agile or CI/CD teams, where development cycles are short, defect observations are sparse, and testing occurs continuously rather than in dedicated phases.

More recently, machine learning techniques have been applied to defect prediction. Approaches based on decision trees, support vector machines, and deep neural networks attempt to identify defect-prone modules using software metrics, process indicators, and historical defect data. Cross-project and deep learning models have demonstrated promising results when large datasets are available [15]. However, such approaches often require extensive historical data and may behave as black-box predictors, limiting interpretability and reducing their practicality for small teams with limited data histories.

Another line of research focuses on adaptive or change-point models that allow defect arrival rates to vary across development intervals. Segmented modeling techniques can capture structural shifts in defect behavior resulting from feature expansion, architectural refactoring, or improvements in testing automation. While these approaches improve flexibility compared to classical SRGMs, most existing models still assume relatively large datasets or longer observation periods than those typically available in small agile projects.

Recent advances in software analytics platforms emphasize continuous monitoring of development activity and quality indicators within DevOps pipelines [16], [17]. These systems integrate defect tracking, build pipelines, and development metrics to provide real-time dashboards of project health. However, many of these platforms focus primarily on descriptive analytics, while lightweight predictive models suitable for sparse data environments remain limited.

The Agile Development Manager (ADM) proposed in this paper addresses these limitations by introducing a lightweight predictive framework specifically designed for small agile teams. ADM applies segmented linear modeling over short rolling windows, enabling early defect trend prediction even under sparse data conditions. Unlike classical SRGMs, ADM does not require long observation periods for model convergence. Unlike machine learning approaches, it preserves a transparent mathematical structure that supports engineering interpretation and operational decision-making. Table I summarizes the differences between major classes of defect prediction approaches and the proposed ADM framework.

TABLE I
COMPARISON OF DEFECT PREDICTION APPROACHES

Approach	Data	Interpretability	Agile Fit
SRGMs [4], [6]	Med–High	High	Limited
Machine Learning [15]	High	Low	Moderate
Change-Point Models [10]	Medium	Medium	Moderate
ADM (Proposed)	**Low**	**High**	**High**

III. ADM OVERVIEW AND PREDICTION METHODOLOGY

ADM is a lightweight predictive quality analytics module designed for small development teams operating under agile and CI/CD practices. Unlike traditional reliability and defect growth models, which typically assume long lifecycles, stable testing phases, and large historical datasets, ADM is engineered to operate under sparse data conditions, short iterations, and continuously evolving project environments. ADM extends the STAR quality-intelligence platform with analytics optimized for incremental development, providing an entry point for predictive quality management with minimal configuration and operational overhead. As projects scale or data maturity increases, teams can transition seamlessly from ADM to full STAR analytics without disrupting workflows.

ADM continuously ingests lightweight defect and development activity data from agile toolchains and applies incremental modeling techniques to generate short-horizon defect forecasts, closure projections, and stability indicators. These outputs are designed to support operational decisions such as sprint planning, backlog prioritization, and release-readiness assessment. By emphasizing near-term trends rather than retrospective metrics, ADM enables small teams to detect emerging instability and intervene before defect accumulation impacts delivery.

At the core of ADM is segmented linear modeling [19] applied over short, rolling time windows. Instead of fitting a single global growth curve, ADM represents cumulative defect behavior as a sequence of locally linear segments, each capturing the defect trend over a specific development interval. This structure allows the model to adapt rapidly to regime changes, including feature expansion, architectural refactoring, improvements to test automation, and shifts into stabilization phases. As new data becomes available, inflection points are identified, segments are updated, and predictions are revised, producing a continuously adapting model of defect arrival and closure dynamics.

ADM analyzes defect data using small, fixed-duration increments (for example, daily or two-day windows). Within each increment, cumulative defect arrival and closure data are evaluated, and candidate linear segments are fit. When a statistically meaningful change in trend is detected, an inflection point is introduced, and a new segment is created. The segmented linear equation is formulated to capture distinct defect trends in each period. The cumulative defects, $m(x)$, observed by time x between inflection points $(j-1)$ and j, are described as

$$m(x) = m(\tau_{j-1}) + \theta_j(x - \tau_{j-1}), \qquad \tau_{j-1} \le x < \tau_j \quad (1)$$

where τ_j is the j-th inflection point, $m(\tau_{j-1})$ is the cumulative number of defects at the previous inflection, and θ_j represents the slope of the curve during the period, corresponding to the local defect arrival or closure rate.

This formulation allows ADM to explicitly represent changing defect dynamics, rather than forcing all project behavior into a single averaged model.

Figure 1 illustrates how ADM constructs a cumulative defect-arrival trend using incremental data windows, enabling early prediction even with limited observations. The resulting arrival curve is then used to forecast the total number of defects expected by the end of development (D_0). Figure 2 shows how ADM detects inflection points in the defect data and adaptively updates both defect-arrival and defect-closure prediction curves as new observations become available. Defect closure is modeled independently, and the difference between the predicted arrival and closure curves defines the open-defect curve, representing detected but unresolved defects. The quality objective is to minimize the number of open defects at D_0 to improve release readiness and delivery stability.

Each time new data arrives, ADM reevaluates the most recent segment and determines whether the current slope remains valid or whether a new inflection point should be introduced. This continuous recalibration produces short-term forecasts for defect arrival and closure while preserving previously learned behavior. Figure 3 illustrates how predictions evolve, with early projections stabilizing as more data becomes available. This stability behavior is critical in agile environments. Early in a sprint or release cycle, predictions may shift as limited data is incorporated. As development progresses, slopes converge, and the prediction bands narrow, providing greater confidence for planning, resourcing, and release-readiness decisions.

By combining segmented modeling with short rolling windows, ADM achieves three objectives essential for small teams: rapid responsiveness to changing project conditions, mathematically interpretable trend representation, and early visibility into defect trajectories. Teams gain timely forecasts of defect growth and closure capability, allowing them to adjust testing intensity, backlog priorities, and stabilization activities before quality risks accumulate.

IV. Case Study and Experimental Results – Impact of Working-Day Normalization

This section presents experimental results from applying ADM to a representative small agile software project. The objective is to evaluate ADM's ability to generate early predictions, adapt to changing defect dynamics, and improve prediction stability under sparse-data conditions. The study also examines the impact of time-scale normalization, specifically, the removal of non-working days, on prediction accuracy.

A. Dataset Description

The dataset used in this study originates from a small-scale Agile development project associated with an interface subsystem of a large telecommunications software platform. The interface module serves as an integration layer between core network services and higher-level application components, thereby representing a realistic subsystem-level development scenario in complex, large-scale telecommunication environments. The project followed an 8-week delivery cycle organized under an Agile development framework. Development activities included incremental feature implementation, interface adaptation, defect correction, and regression testing. The short development horizon and iterative structure make this dataset particularly suitable for analyzing the dynamics of defect arrival and resolution under compressed schedules.

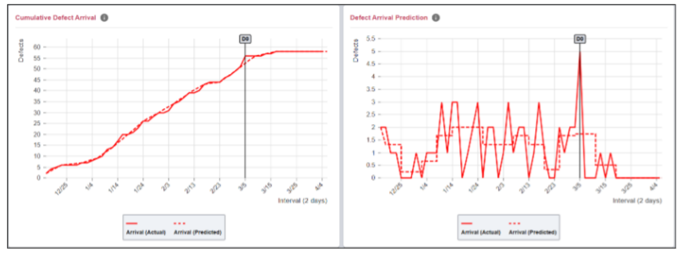

Fig. 1. ADM predictions of defect detection curves for a small agile project.

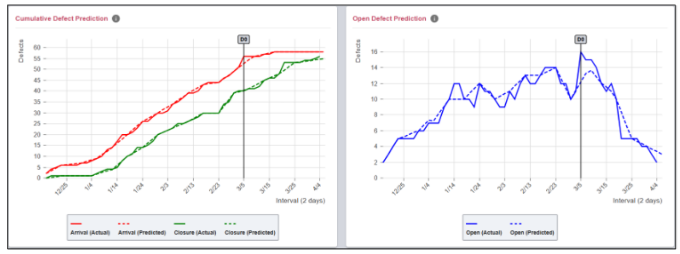

Fig. 2. ADM predictions of defect detection and fix curves for a small agile project.

The project exhibited a characteristic temporal evolution. During the initial weeks, development progressed at a relatively modest pace. The defect discovery rate was correspondingly low, reflecting limited functional exposure and incomplete integration. As feature implementation expanded and integration activities intensified, defect detection increased. Following this ramp-up phase, the defect arrival and closure processes transitioned into a more stable regime, characterized by relatively consistent inflow and resolution rates.

The dataset therefore captures three distinct development phases commonly observed in short-cycle Agile projects: (1) an initial ramp-up phase with limited defect visibility, (2) an acceleration phase associated with increased integration and defect discovery, and (3) a stabilization phase in which defect arrival and correction rates approach dynamic equilibrium. Due to its controlled scope, well-defined duration, and observable transition dynamics, this dataset provides an appropriate empirical basis for evaluating predictive defect modeling approaches, stability assessment techniques, and methods for quantifying early-release risk.

B. Results

Figures 1 and 2 show ADM's defect arrival and closure predictions generated using incremental data windows early in the development cycle. Despite limited initial data, ADM constructs segmented linear models that capture emerging

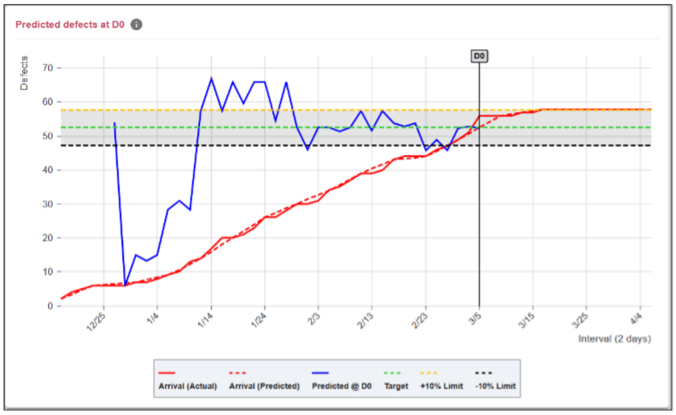

Fig. 3. Prediction stability of the defect detection curve for a small agile project.

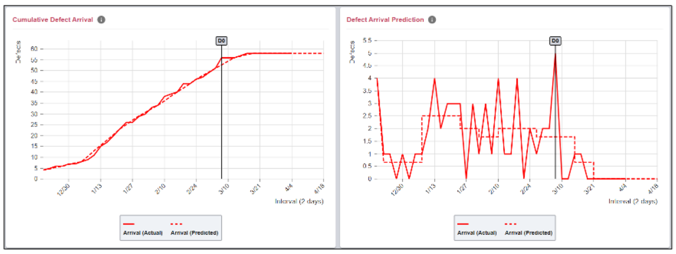

Fig. 4. ADM predictions of the defect detection curve for a small agile project based on working days.

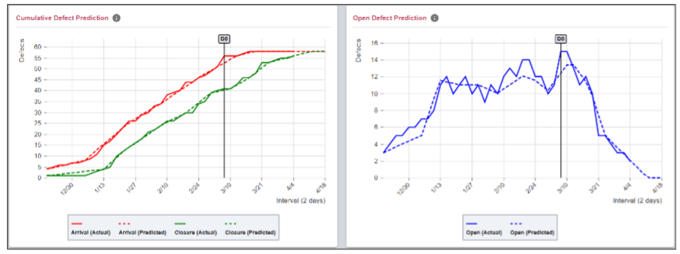

Fig. 5. ADM predictions of defect detection and fix curves for a small agile project based on working days.

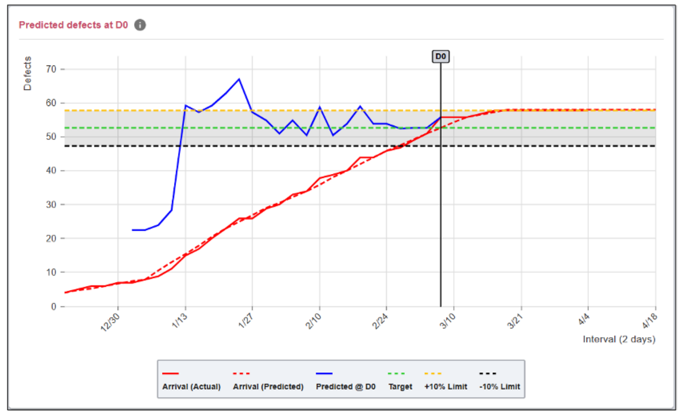

Fig. 6. Prediction stability of the defect detection curve for a small agile project based on working days.

trends and automatically introduce inflection points when statistically meaningful changes are detected. This enables usable forecasts long before traditional models would converge. Figure 3 illustrates the evolution of predictions over time. Early in the cycle, prediction slopes vary as new data arrives. As development progresses, the segmented structure stabilizes, slopes converge, and forecast variance narrows. This behavior demonstrates ADM's ability to balance responsiveness with stability—adapting quickly when conditions change, while avoiding excessive oscillation once trends are established.

Figures 4, 5, and 6 examine the same project data after eliminating non-working days (weekends and holidays) and recalculating cumulative defect curves using only working days. This transformation has a measurable effect on early-phase predictions. In Figure 4, the apparent low defect detection rate observed in the early calendar-day model disappears when defect activity is expressed in working-day time. The initial slope becomes consistent with later development phases, eliminating the artificial flattening caused by weekends and holidays. This indicates that early underestimation of defect rates was not a process effect, but a time-scale artifact. Figures 5 and 6 further show that working-day normalization produces smoother segmented fits, more consistent slopes, and earlier stabilization of predictions.

Forecast accuracy improves because ADM's segmented models are driven by actual engineering activity rather than idle calendar intervals. This effect is significant for small agile projects, where total defect volumes are low and a small number of non-working days can significantly distort perceived defect rates. By aligning time measurement with actual development effort, ADM improves sensitivity, reduces noise, and strengthens the reliability of early predictions. The normalization by working-day time is analogous to the execution time model [8].

The experimental results demonstrate that ADM's segmented modeling approach provides stable and responsive defect trend prediction even under sparse, short-window data conditions typical of small agile projects. The figures show that ADM captures both early-stage volatility and later stabilization, enabling meaningful predictions before traditional methods become effective. A key observation is the strong sensitivity of prediction accuracy to data normalization. Figures 4–6 illustrate that removing non-working days significantly improves trend clarity and early-phase prediction stability. This confirms that defect rates based on effective working days provide a more accurate representation of engineering activity, particularly for small teams generating low daily defect volumes. From an engineering perspective, these results indicate that lightweight, window-based predictive analytics can support actionable quality management without requiring large defect datasets or long test phases. ADM enables early detection of abnormal trends, supports continuous calibration of development effort, and provides a practical foundation for proactive corrective actions within agile and CI/CD workflows.

TABLE II
PREDICTION STABILIZATION COMPARISON BETWEEN STAR AND ADM

Metric	STAR (Baseline)	ADM
Model role	Enterprise-quality defect prediction platform with high trend fidelity	Lightweight segmented modeling designed for small Agile teams
Prediction stabilization definition	Prediction remains within $\pm 10\%$ of final estimate over subsequent updates	Same criterion applied
Operational impact	Later stabilization reduces early intervention window	Earlier stabilization enables proactive resource and testing adjustments

Start	End
17/12/2018	06/03/2019

	Stable date	Duration (stable - start)	%improvement over STAR	% of start to end for stabilization
STAR	05/02/2019	50		63%
ADM - calendar days	30/01/2019	44	12%	56%
ADM - working days	13/01/2019	27	46%	34%

Fig. 7. ADM Comparison with STAR

V. ADM COMPARISON WITH STAR

To evaluate ADM's performance improvement in short-cycle Agile projects, we compare its stabilization behavior with STAR. STAR is used as the baseline reference because, among existing defect prediction and reliability growth models, it uniquely demonstrates near-perfect trend tracking across the majority of project data. No other established models provide comparable fidelity in capturing the dynamics of defect arrival and closure; therefore, STAR serves as a rigorous, technically defensible benchmark rather than a weak baseline.

In this study, prediction stabilization is formally defined as the time at which the predicted number of defects at the development end date remains within ±10% of its eventual steady-state estimate over subsequent updates. Using the 2-day interval dataset, ADM stabilized its predictions on January 30, whereas STAR stabilized on February 5. Given the project start date of December 17, ADM delivered stable, actionable predictive insights 6 days earlier, resulting in a 12% reduction in time-to-stabilization relative to STAR. Operationally, this earlier stabilization expands the window for management intervention, including resource reallocation, test-intensity adjustments, and release-risk reassessment.

When working-day normalization was applied, ADM stabilized on January 13. This represents a 46% reduction in time-to-stabilization compared to STAR. Under this configuration, reliable predictive insight becomes available during the early-to-mid phase of development rather than closer to late integration, substantially enhancing proactive control capability. Considering the full project duration from December 17 to the development end date of March 6, STAR required approximately 63% of the project timeline to reach stabilization. In contrast, ADM required 56% under calendar-day normalization (an 11% relative reduction) and only 34% under working-day normalization, representing a 46% reduction relative to STAR in the time required to achieve stable prediction. These results demonstrate that ADM not only accelerates stabilization in absolute days but also significantly reduces the proportion of the lifecycle required to obtain reliable predictive confidence.

It should be noted that the current ADM implementation excludes weekends but does not yet remove holidays and other non-working days. Incorporating these additional inactive periods is expected to advance stabilization time further while maintaining high predictive accuracy.

VI. CONCLUSION AND FUTURE WORK

This paper introduces ADM, a lightweight quality-intelligence module designed to support small, agile, and CI/CD software teams through segmented, window-based defect modeling. The presented methodology demonstrates that meaningful prediction and early-risk visibility can be achieved even under sparse data conditions, where traditional reliability and defect models typically underperform. The experimental results confirm that ADM can stabilize early predictions, capture evolving defect trends, and provide actionable insight without imposing heavy process or data-collection overhead.

Future work will focus on expanding ADM's modeling depth and operational integration. Planned directions include incorporating severity-aware and component-level modeling, adaptive window selection, and automated calibration of inflection points. Additional validation across multiple industrial projects is also scheduled to quantify prediction accuracy, robustness, and decision impact. Longer-term research will explore integrating ADM outputs with corrective-action optimization and AI-assisted development analytics further to enhance proactive quality management in fast-cycle software environments.

ACKNOWLEDGMENT

The author gratefully acknowledges Rashid Mijumbi, Joseph Good, Mike Okumoto, and Ezra Redeatu for their valuable contributions to this paper.

REFERENCES

[1] J. Humble and D. Farley, *Continuous Delivery: Reliable Software Releases through Build, Test, and Deployment Automation*. Boston, MA, USA: Addison-Wesley Professional, 2010.

[2] N. Forsgren, J. Humble, and G. Kim, *Accelerate: The Science of Lean Software and DevOps: Building and Scaling High-Performing Technology Organizations*. Portland, OR, USA: IT Revolution Press, 2018.

[3] J. Highsmith, *Agile Project Management: Creating Innovative Products*, 2nd ed. Boston, MA, USA: Addison-Wesley Professional, 2009.

[4] Z. Jelinski and P. Moranda, “Software reliability research,” in *Statistical Computer Performance Evaluation*, W. H. Freiberger, Ed. Englewood Cliffs, NJ, USA: Prentice-Hall, 1972, pp. 465–484.

[5] L. H. Schick and C. Wolverton, “An investigation of software reliability and associated metrics,” *IBM Journal of Research and Development*, vol. 20, no. 3, pp. 308–326, May 1976.

[6] A. R. 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.

[7] S. Yamada and S. Osaki, “Software reliability growth modeling: Models and applications,” *IEEE Transactions on Software Engineering*, vol. SE-11, no. 12, pp. 1431–1437, Dec. 1985.

[8] J. D. Musa and K. Okumoto, “A logarithmic Poisson execution time model for software reliability measurement,” in *Proc. 7th Int. Conf. Software Reliability Engineering (ISSRE)*, Washington, DC, USA, 1986, pp. 126–132.

[9] H. Pham, *Software Reliability*. Springer Science & Business Media, 2000.

[10] M. Jain, T. Manjula, and T. R. Gulati, “Software Reliability Growth Model with Imperfect Debugging, Fault Reduction Factor and Multiple Change Point,” in *Advances in Intelligent and Soft Computing*. Springer, 2011, pp. 1027–1037.

[11] M. Anjum, A. N. Ahmad, and A. Asraf, “Analysis and ranking of software reliability growth models using multi-criteria decision making,” *International Journal of Information Technology and Computer Science*, vol. 5, no. 2, pp. 1–13, 2013.

[12] H. Okamura and T. Dohi, “SRATS: Software Reliability Assessment Tool on Spreadsheet,” Research Report, 2016.

[13] V. Nagaraju, V. Shekar, J. Steakelum, M. Luperon, Y. Shi, and L. Fiondella, “Practical software reliability engineering with the Software Failure and Reliability Assessment Tool (SFRAT),” *SoftwareX*, vol. 10, Art. no. 100357, 2019.

[14] N. A. AL-Saati and M. Abd-AlKareem, “Selecting Best Software Reliability Growth Models: A Social Spider Algorithm-based Approach,” *arXiv preprint arXiv:2001.09924*, 2020.

[15] T. Kim, D. Ryu, and J. Baik, “Deep Synthetic Cross-Project Approaches for Software Reliability Growth Modeling (DSC-SRGM),” *arXiv preprint arXiv:2509.16939*, Sep. 2025.

[16] K. Okumoto, “Digital Transformation in Software Quality Assurance,” in *Analytics Modeling in Reliability and Machine Learning and Its Applications*, H. Pham, Ed. Cham, Switzerland: Springer, 2025, pp. 237–265.

[17] K. Okumoto, “Digital Engineering-Driven Software Quality Assurance-as-a-Service,” in *Proc. 28th Int. Symp. Reliability and Quality in Design (ISSAT RQD)*, San Francisco, CA, USA, Aug. 2023.

[18] 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)*, Charlotte, NC, USA, Oct. 2022, pp. 43–48.

[19] K. Okumoto and H. Pham, “FUSION: A Cloud-Based Framework for Integrated Software-Hardware Reliability Assessment Using Advanced Analytics and RBDs,” in *Proc. 30th Int. Symp. Reliability and Quality in Design (ISSAT RQD)*, Honolulu, HI, USA, Aug. 2025.