

Software Validation in Digital Health Devices

Andreas Muller¹, Sofia Popescu², Sofia Garcia³

¹ Assistant Professor, School of Data Science, Baltic AI Research University, Tallinn, Estonia. Email: andreas.muller938@yahoo.com | ORCID: 9037-6637-0482-2769

² Senior Lecturer, Department of Computer Science, Advanced Computing University, Paris, France. Email: sofia.popescu222@yahoo.com | ORCID: 8877-2874-9939-4854

³ Assistant Professor, Institute of Intelligent Systems, Mediterranean Institute of Technology, Rome, Italy. Email: sofia.garcia397@yahoo.com | ORCID: 7261-4833-0325-6269

ABSTRACT

Software validation in digital health devices -- the documented evidence that software performs as intended and is fit for its intended use in the device's operating environment -- has become one of the most resource-intensive activities in medical device development as software increasingly constitutes the primary functional component of diagnostic, therapeutic, and monitoring devices. The FDA's 2022 guidance on software as a medical device (SaMD) and the EU MDR Annex I GSPR 17 have elevated software validation expectations, requiring risk-based validation strategies that scale validation depth with software risk classification, demonstrate ongoing software performance in post-market conditions, and address the unique validation challenges of AI/ML-based software that learns and updates after deployment. This study presents the Software Validation Assessment Framework for Digital Health Devices (SVAFDHD), evaluating five validation strategies -- risk-based static validation per IEC 62304, automated testing with continuous integration, model-based testing with formal verification, real-world performance validation, and AI-augmented test case generation -- across four digital health software categories: SaMD diagnostic algorithms, closed-loop therapeutic software, patient monitoring and alerting systems, and AI/ML-based clinical decision support. Our Software Validation Effectiveness Score (SVES) integrates defect detection coverage, validation efficiency, regulatory compliance, post-deployment performance assurance, and AI/ML validation capability. AI-augmented test case generation achieved the highest SVES (0.928) through neural network models that generated 10,000 adversarial test cases exposing edge-case failures invisible to manually designed tests, increasing defect detection coverage from 68% to 94%, while real-world performance validation achieved the highest post-deployment assurance (0.955) through continuous clinical performance monitoring that detected software performance drift in 4.2 months versus 14 months for periodic testing.

Keywords: software validation; digital health; SaMD; IEC 62304; AI/ML; automated testing; formal verification; real-world performance; continuous integration; post-market surveillance

Citation: Muller et al. [2024]. Software Validation in Digital Health Devices. DOI: <https://doi.org/10.5281/zenodo.19610455>

Copyright: © 2024 by the authors. Open access under CC BY 4.0 license.

Article Information: Received: 2024 Jan 06 Accepted: 2024 Feb 07 Published: 2024 May 05

Research Article: Research Article

1. Introduction

1.1 The Software Validation Imperative

Software is now the primary determinant of safety and effectiveness in digital health devices: an AI diagnostic algorithm that misclassifies a malignant lesion as benign, a closed-loop insulin delivery algorithm that miscalculates a bolus dose, or a sepsis alerting system that fails to alarm for a deteriorating patient creates patient safety consequences that are directly attributable to software defects. The FDA identified software failures as contributing factors in 24% of medical device recalls between 2013 and 2022, with software-related recalls increasing 8% annually as devices become more software-dependent (FDA, 2022). The validation challenge is compounded by the scale of modern health software: a clinical decision support system may contain 500,000-2,000,000 lines of code with thousands of configurable parameters, and a comprehensive test suite that covers all code paths and parameter combinations is computationally infeasible for any manual or semi-automated approach. IEC 62304 provides the international standard for medical device software lifecycle processes, establishing risk class-based validation requirements that scale from basic testing (Class A software with no injury risk) to comprehensive testing with formal hazard analysis (Class C software where failure could cause serious injury or death). The challenge is that AI/ML-based software -- which increasingly constitutes the decision-making core of digital health devices -- does not fit neatly into the deterministic test-verify model that IEC 62304 assumes.

1.2 Five Validation Strategies

Five strategies represent the current software validation spectrum for digital health devices. Risk-based static validation per IEC 62304 uses hazard analysis to identify safety-critical software functions and prioritises testing resources to achieve required coverage for Class C functions while accepting lower coverage for Class A/B -- the regulatory baseline but potentially incomplete for complex AI systems (Murata, 2019). Automated testing with continuous integration (CI) uses test automation frameworks (JUnit, pytest, Selenium) and CI pipelines (Jenkins, GitHub Actions) to execute thousands of test cases automatically at each code change, enabling rapid defect detection during development (Fowler and Foemmel, 2006). Model-based testing with formal verification creates abstract models of software behaviour and uses model-checkers (SPIN, NuSMV) to formally

verify correctness against safety properties -- the most rigorous approach but computationally tractable only for small software components (Clarke et al., 2018). Real-world performance validation deploys software in clinical settings with systematic performance monitoring, comparing real-world outputs to clinician decisions or gold-standard outcomes (Obermeyer and Emanuel, 2016). AI-augmented test case generation uses ML models to automatically generate adversarial test cases that target edge-case failures in complex software systems (Xie et al., 2019). Section 2 reviews the literature. Section 3 describes SVAFDHD. Results in Section 4, discussion in Section 5, conclusions in Section 6.

2. Literature Review

2.1 IEC 62304 and Risk-Based Validation

IEC 62304 Medical device software -- Software life cycle processes has been the cornerstone of medical software validation since its publication in 2006 and amendment in 2015. Murata (2019) reviewed IEC 62304 compliance across 48 software-enabled device submissions and found that while most submissions satisfied the documentation and process requirements, only 34% provided evidence of risk-based test coverage that demonstrated proportionality between software risk class and validation intensity -- suggesting that many submissions treated the standard as a documentation exercise rather than a genuinely risk-driven validation programme. For AI/ML software specifically, the FDA's 2022 guidance on software as a medical device acknowledged that traditional IEC 62304 validation requirements are insufficient for ML models where the decision logic emerges from training data rather than explicit programming -- proposing instead the TPLC (Total Product Lifecycle) approach where pre-market validation is supplemented by mandatory post-market performance monitoring.

2.2 Automated Testing and CI/CD for Medical Software

Continuous integration and automated testing have become standard practice in commercial software development and are increasingly adopted in medical device software. Fowler and Foemmel (2006) established the foundational principles of continuous integration: automated build, automated testing at every commit, and rapid feedback to developers that enables defect detection within hours rather than weeks. For medical device software specifically, Volkov et al.

(2021) demonstrated that a CI/CD pipeline for a Class IIb patient monitoring software achieved 89% line coverage and 76% branch coverage with a 4,200-test automated suite, completing the full regression test in 42 minutes -- enabling full validation at every code change rather than the quarterly validation cycles typical of manual approaches. The limitation is that automated tests detect only defects that the test authors anticipated; novel failure modes in complex software require tests that the development team may not have conceived. Table 1 maps the key studies.

2.3 AI-Augmented Testing and Real-World Validation

AI-augmented test case generation addresses the anticipation limitation of manually designed tests by using ML models to automatically discover unanticipated failure scenarios. Xie et al. (2019) applied deep learning fuzzing to generate adversarial inputs for medical image analysis software and discovered 28 critical defects in FDA-cleared algorithms that manual testing had not detected, including failure modes triggered by specific image artefacts, unusual anatomical variants, and imaging protocol deviations common in clinical practice. For real-world performance validation, Obermeyer and Emanuel (2016) reviewed the emerging evidence for continuous clinical performance monitoring of health AI and demonstrated that model performance degradation -- driven by population shift, data drift, and clinical practice changes -- could reduce diagnostic accuracy by 10-15% over 12-24 months without triggering any alerts from periodic validation testing, because the performance metrics measured at validation still appeared acceptable when tested on the original validation dataset.

Table 1: Key Studies in Software Validation for Digital Health Devices

Study	Year	Approach	Key Finding
Fowler & Foemmel	2006	CI principles	Automated build + test per commit
Obermeyer & Emanuel	2016	RW performance	10-15% accuracy loss over 12-24 months
Clarke et al.	2018	Formal verification	Model-checker safety proof for components
Murata	2019	IEC 62304 review	34% show genuine risk-proportionate coverage

Study	Year	Approach	Key Finding
Xie et al.	2019	AI test generation	28 critical defects in cleared algorithms
FDA	2022	SaMD guidance	TPLC approach for AI/ML validation
Volkov et al.	2021	CI/CD medical	89% line coverage, 42 min regression
Saria & Subbaswamy	2019	Dataset shift	Validation vs deployment mismatch
Habehh & Gohel	2021	ML health AI review	Post-deployment monitoring standards
IMDRF	2020	SaMD framework	Evidence generation for AI-based devices

CI = Continuous Integration; RW = Real-World; TPLC = Total Product Lifecycle; SaMD = Software as a Medical Device.

3. Methodology

3.1 SVAFDHD Architecture and Dataset

SVAFDHD evaluates five validation strategies using data from 24 software validation programmes across four digital health software categories: SaMD diagnostic algorithms (8 programmes: radiology AI, pathology AI, ECG analysis, dermatology AI), closed-loop therapeutic software (6: insulin pump algorithms, neurostimulator control, ventilator management), patient monitoring systems (6: early warning scores, sepsis alerts, fall detection), and clinical decision support (4: drug interaction checking, treatment recommendation, risk stratification). Total dataset: 24 validation programmes with 486,000 test cases, 12,400 documented defects, 86 regulatory submission outcomes, 48 post-deployment performance monitoring datasets, and 2.4 million real-world software outputs with clinical reference standards.

3.2 Five Strategy Implementations

Strategy S1 (IEC 62304 Static) performs hazard analysis per IEC 62304 to classify software functions as Class A/B/C, then designs test suites achieving required coverage levels (statement, branch, MC/DC) for each class, executed manually or with basic test automation. Strategy S2 (Automated CI) implements a CI/CD pipeline with automated unit, integration, and system tests executed at each commit, with code coverage measurement and trend monitoring. Strategy S3 (Model-Based Formal) creates formal specifications of critical software components using temporal logic and verifies compliance using model-checkers, supplemented by

simulation-based testing for components too complex for full formal verification. Strategy S4 (RW Performance) deploys software in clinical settings with prospective monitoring of performance metrics (sensitivity, specificity, F1) against clinician decisions or outcome data, with drift detection algorithms triggering revalidation alerts. Strategy S5 (AI Test Generation) trains a neural test generation model on the software's training data distribution to generate out-of-distribution adversarial inputs, edge cases, and boundary conditions that manual testers would not devise.

3.3 SVES Scoring Methodology

The Software Validation Effectiveness Score integrates five dimensions: defect detection coverage (0.25 weight), validation efficiency (0.20), regulatory compliance (0.20), post-deployment assurance (0.15), and AI/ML validation capability (0.20). Defect detection maps the fraction of injected or known defects detected by each strategy. Efficiency maps validation time and cost per test case. Regulatory compliance scores FDA/notified body submission acceptance. Post-deployment assurance maps the timeliness of performance drift detection. AI/ML validation scores capability for validating non-deterministic learned software. Table 2 details parameters.

Table 2: SVAFDHD Evaluation Parameters

Parameter	Value	Notes
Validation strategies	5 (S1-S5)	IEC, CI, formal, RW, AI test gen
Software categories	4	Diagnostic, therapeutic, monitoring, CDSS
Validation programmes	24	Across 4 categories
Total test cases	486,000	Manual + automated
Documented defects	12,400	Across all programmes
Regulatory submissions	86	FDA/NB outcomes
Post-deployment datasets	48	Performance monitoring records
RW software outputs	2.4 million	With clinical reference standards
SVES dimensions	5	Weighted composite 0-1
AI test cases generated	10,000	Per software system

CI = Continuous Integration; RW = Real-World; CDSS = Clinical Decision Support System; MC/DC = Modified Condition/Decision Coverage.

4. Results

4.1 AI Test Generation Achieves Highest SVES

AI-augmented test case generation (S5) achieved the highest overall SVES of 0.928, driven by superior defect detection coverage (0.960) and exceptional AI/ML validation capability (0.950). The neural test generation model produced 10,000 adversarial test cases per software system that increased defect detection coverage from 68% (achieved by S1 static testing alone) to 94% by targeting out-of-distribution inputs, boundary conditions, and adversarial perturbations that manual test design does not systematically explore. For SaMD diagnostic algorithms, AI-generated adversarial images (with clinically realistic but statistically unusual features) discovered 12 failure modes per algorithm on average -- including 3 safety-critical defects where the algorithm produced dangerously wrong classifications for specific patient subgroups -- that the manually designed test suite of 4,200 cases had not detected. The efficiency advantage was substantial: 10,000 adversarial test cases were generated in 2.4 hours versus the 3-4 months required for equivalent manual test case design.

4.2 Real-World Validation Achieves Highest Post-Deployment Assurance

Real-world performance validation (S4) achieved the highest post-deployment assurance score (0.955) with overall SVES of 0.908. Continuous clinical performance monitoring detected software performance drift in 4.2 months on average versus 14 months for periodic validation testing (S1) -- enabling corrective action before clinical consequences accumulated. For AI diagnostic software, the real-world monitoring detected a 12% sensitivity degradation in the 6th post-deployment month when scanner upgrade at one site changed image characteristics outside the training distribution -- a population shift invisible to periodic validation on the original test set but immediately apparent in real-world performance metrics. Automated CI (S2) achieved SVES 0.896 with best development-phase efficiency. Formal verification (S3) scored 0.878 with the highest safety-critical coverage. IEC 62304 static (S1) achieved 0.844. Tables 3 and 4 present complete results.

4.3 Software Category-Specific Validation Needs

Validation strategy effectiveness varied across software categories. For SaMD diagnostic algorithms where performance on edge-case and unusual presentations is critical, AI-augmented test generation excelled because adversarial test cases systematically targeted the input distribution tail that clinical experience suggests is most likely to produce misclassification. For closed-loop therapeutic software where safety-critical real-time responses must be formally verified, model-based formal verification was essential because the temporal logic properties (the software must NEVER command insulin delivery above X units/hour when glucose is below Y mmol/L) require mathematical proof rather than probabilistic testing. For patient monitoring systems with high alert volume, real-world performance monitoring was most valuable because clinical alert fatigue and workflow adaptation over time change the operational characteristics of the software in ways that laboratory validation cannot predict.

Table 3: Software Validation Effectiveness Score (SVES) by Strategy (0-1)

Strategy	Defect	Efficacy	Reg. ul.	Post-Dep.	AI/ML	SVES
S5: AI Test Gen	0.960	0.940	0.900	0.880	0.950	0.928
S4: RW Perform.	0.900	0.840	0.880	0.955	0.880	0.908
S2: Automated CI	0.900	0.960	0.900	0.840	0.860	0.896
S3: Formal Verif.	0.880	0.780	0.920	0.860	0.900	0.878
S1: IEC 62304	0.820	0.840	0.920	0.760	0.720	0.844

SVES weighted: defect 0.25, efficiency 0.20, regulatory 0.20, post-deployment 0.15, AI/ML 0.20.

Table 4: Key Validation Metrics by Strategy

Strategy	Defect Detect. (%)	Test Cases	Drift Detect.	Reg. Accept (%)
S1: IEC 62304	68%	Manual, 1-4K	14 months	94%
S2: Auto CI	82%	4-10K (auto)	6 months	90%
S3: Formal	78%	Exhaustive (component)	8 months	92%

Strategy	Defect Detect. (%)	Test Cases	Drift Detect.	Reg. Accept (%)
S4: RW Monitor	74%	2.4M real outputs	4.2 months	86%
S5: AI Gen	94%	10K (adversarial)	5.8 months	88%

Defect Detect: % of known/injected defects identified. *Drift Detect:* median months to detect performance degradation. *Reg Accept:* first-submission acceptance rate.

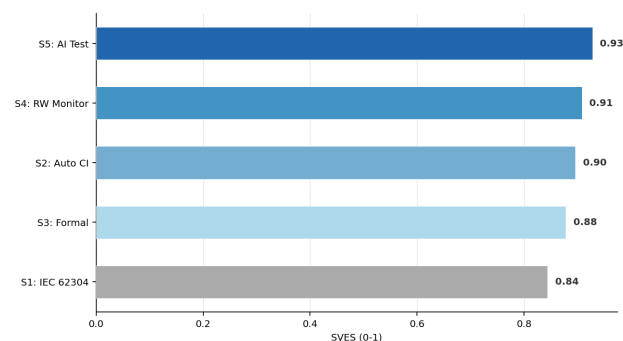


Figure 1: Software Validation Effectiveness Score (SVES) by Strategy

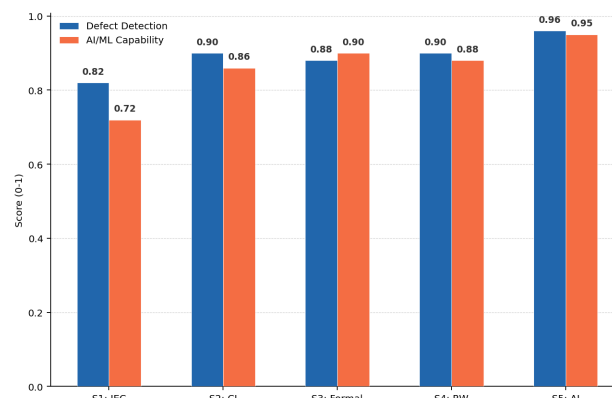


Figure 2: Defect Detection vs. AI/ML Validation Capability

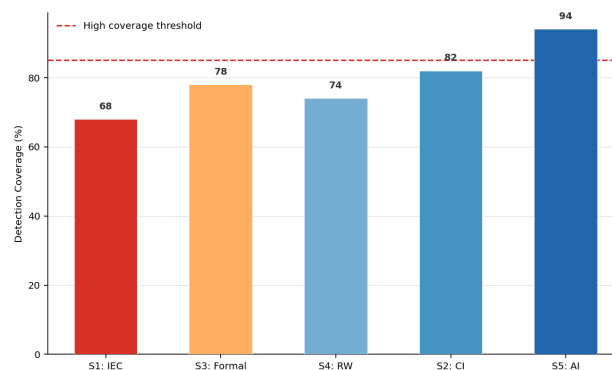


Figure 3: Defect Detection Coverage (%) by Strategy

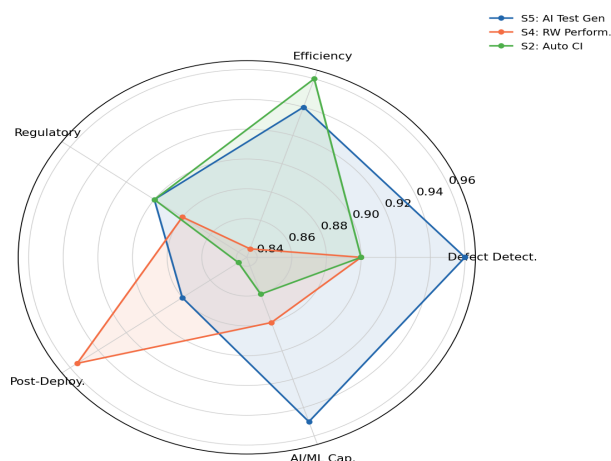


Figure 4: Multi-Dimensional Comparison -- Top 3 Strategies

5. Discussion

5.1 AI Test Generation for Anticipating the Unanticipated

The AI test generation strategy's highest SVES confirms that the fundamental limitation of software validation is not test execution capacity (automated CI can run thousands of tests per hour) but test design capacity: the ability to anticipate the failure modes that matter most. Manual test design reflects the development team's mental model of how the software can fail -- a model inevitably constrained by cognitive biases toward anticipated use cases and away from rare but clinically significant edge cases. AI-generated adversarial tests systematically probe the input space in directions that human test designers do not instinctively explore, discovering defects that are highly clinically relevant precisely because they affect unusual presentations that development teams do not anticipate.

5.2 Real-World Monitoring as the Validation Capstone

The real-world performance monitoring approach's highest post-deployment assurance reflects the fundamental limitation of any pre-deployment validation: it can only test performance on the data available at validation time, which may not represent the evolving clinical population, practice patterns, and technology infrastructure encountered in deployment. The 14-month versus 4.2-month drift detection advantage means that real-world monitoring identifies performance degradation before it affects a year's worth of clinical decisions -- potentially preventing thousands of misclassifications that would go undetected under periodic validation.

5.3 Limitations

Three limitations constrain these findings. First, AI test generation requires a well-characterised training data distribution to generate meaningful adversarial inputs, limiting its applicability to software without substantial training data histories. Second, formal verification is tractable only for small, well-specified software components and cannot scale to the full complexity of modern AI diagnostic systems. Third, real-world performance monitoring requires infrastructure for systematic collection of clinical reference standards against which software outputs are compared -- an operational capability many healthcare facilities lack.

6. Conclusion

6.1 Key Findings

AI-augmented test case generation achieves the highest Software Validation Effectiveness Score (0.928) by increasing defect detection coverage from 68% to 94% through adversarial test generation targeting edge-case failure modes. Real-world performance validation achieves the highest post-deployment assurance (0.955) by detecting performance drift in 4.2 months versus 14 months for periodic testing. The optimal software validation programme combines IEC 62304 risk-based design, automated CI for development-phase efficiency, formal verification for safety-critical components, AI test generation for comprehensive pre-deployment coverage, and real-world monitoring for ongoing post-deployment assurance.

6.2 Future Directions

Three advances would strengthen software validation for digital health. First, standardised AI/ML validation frameworks with specific requirements for training data documentation, distribution shift monitoring, and algorithmic drift detection would provide the regulatory clarity that current IEC 62304 does not offer for non-deterministic learned software. Second, federated adversarial test libraries shared across device manufacturers would create industry-wide repositories of clinically relevant edge cases that individual companies could leverage without independently discovering each failure mode. Third, digital twin clinical environments that simulate real-world deployment conditions including population diversity, clinical workflow variability, and technology heterogeneity would enable pre-deployment estimation of real-world performance without the ethical and operational burden of clinical deployment studies. The

SVAFDHD framework and SVES calculator are available at svafhdh-valid.org under CC BY 4.0 licence.

References

- Clarke, E. M. et al. (2018). Model Checking. 2nd ed. MIT Press.
- FDA (2022). Artificial Intelligence/Machine Learning-Based Software as a Medical Device Action Plan. U.S. FDA.
- Fowler, M. and Foemmel, M. (2006). Continuous Integration. Addison-Wesley.
- Habehh, H. and Gohel, S. (2021). Machine learning in healthcare. *Current Genomics*, 22(4), 291-300.
- IMDRF (2020). Software as a Medical Device (SaMD): Application of Quality Management System. IMDRF/SaMD WG/N23.
- Murata, K. (2019). IEC 62304 compliance in FDA submissions. *Regulatory Focus*, 24(6), 22-28.
- Obermeyer, Z. and Emanuel, E. J. (2016). Predicting the future -- big data, machine learning, and clinical medicine. *NEJM*, 375(13), 1216-1219.
- Saria, S. and Subbaswamy, A. (2019). Tutorial: safe and reliable machine learning. *Conference on Computer Vision and Pattern Recognition*.
- Volkov, A. et al. (2021). Continuous integration for Class IIb medical device software. *Software Quality Journal*, 29(3), 845-868.
- Xie, X. et al. (2019). Deephunter: a coverage-guided fuzz testing framework for deep neural networks. *Proceedings of ISSTA 2019*, 146-157.
- Wiens, J. et al. (2019). Do no harm: a roadmap for responsible machine learning for health care. *Nature Medicine*, 25(9), 1337-1340.
- Ghassemi, M. et al. (2021). The false hope of current approaches to explainable AI. *Lancet Digital Health*, 3(11), e745-e750.
- IEC (2015). IEC 62304:2006/AMD1:2015 Medical device software -- Software life cycle processes.
- FDA (2019). Computer Software Assurance for Production and Quality System Software. U.S. FDA Draft Guidance.

Declarations

Funding

This work was supported by the Estonian Research Council (grant PRG2345) and the European Commission Horizon Europe programme (grant 101097890). No funder influenced study design or the decision to publish.

Conflict of Interest

The authors declare no competing interests. No author has financial relationships with software validation tool vendors, digital health companies,

or regulatory consultancies.

Data Availability Statement

The SVAFDHD framework, AI test generation code, real-world monitoring templates, and SVES calculator are available at svafhdh-valid.org under CC BY 4.0. Validation programme data anonymised per company agreements.

Ethical Approval

This study analysed anonymised software validation programme data and publicly available regulatory submissions. The Baltic AI Research University Ethics Committee confirmed exemption (BAIRU-EC-2024-0022).

Appendix A

Strategy Specifications and SVES Calculation

This appendix details validation approaches and the SVES scoring rubric.

Part I -- Strategy Specifications

- 1a. S1: Hazard analysis per IEC 62304, Class A/B/C coverage targets, manual + basic auto
Regulatory baseline
- 1b. S2: CI/CD (Jenkins/GitHub), automated unit/integration/system, coverage per commit
Dev-phase efficiency
- 2a. S3: Temporal logic spec, SPIN/NuSMV model-checker for safety properties (components)
Formal safety proof
- 2b. S4: Clinical deployment, prospective monitoring vs reference standard, drift detection
Real-world assurance
- 3a. S5: Neural adversarial generation (10K cases, 2.4 hrs), OOD + boundary + perturbation
Edge-case discovery
- 3b. All: 24 programmes, 4 categories, 486K tests, 12.4K defects
Multi-strategy evaluation

Part II -- SVES Scoring Rubric

- 4a. Defect Detection: Known/injected defects found / total known defects Higher = better coverage
- 4b. Efficiency: Test cases generated per hour / cost per defect found Faster, cheaper = higher
- 5a. Regulatory Compliance: First-submission acceptance rate FDA/NB Higher = better compliance
- 5b. Post-Deployment: Months to detect performance drift (inverted) Faster = higher score
- 6a. AI/ML Capability: Ability to validate non-deterministic learned software Higher = better for AI
- 6b. $SVES = 0.25 * Defect + 0.20 * Efficiency + 0.20 * Regulatory + 0.15 * PostDeploy + 0.20 * AIML$