AMERICAN JOURNAL OF AI CYBER COMPUTING MANAGEMENT
E-ISSN:3069-0102
VOL.5, NO. 4(1)(2025)

Agile Methodologies in Regulatory Programming: Lessons from Applying Sprint-Based Delivery to FDA and EMA Submission Timelines

Dharma Dev Bommi
Principal Statistical Programmer

Abstract—Regulatory programming — the production of CDISC-compliant SDTM and ADaM datasets, TLFs, and submission packages for FDA and EMA review — has historically operated under waterfall project management models poorly suited to the iterative, specification-dependent nature of the work. This paper reports on a two-year implementation of Agile Scrum methodology in a regulatory programming function supporting 18 concurrent clinical studies, examining the impact on submission timeline adherence, defect rates, mock TLF shell delivery, submission package assembly, and health authority (HA) questionnaire response efficiency. Outcomes were measured across three NDA-track submission programmes under Agile governance compared to three matched historical submissions under the waterfall model. Regulatory Scrum delivered a 46% reduction in late specification changes reaching the programming stage, a 44% reduction in final TLF defect rates, an improvement in pre-lock mock shell approval rates from 43% to 89%, and a 46% reduction in HA questionnaire response compilation time. We document the adaptations required to reconcile Agile philosophy with the audit trail and change control requirements of GxP-regulated environments, and propose a 'Regulatory Scrum' framework that preserves waterfall compliance attributes while capturing the efficiency and quality benefits of iterative delivery. Key strategies for managing mock shell sprints, eCTD assembly workstreams, and country-specific HA question handling under sprint cadence are presented.

Keywords—Agile, Scrum, regulatory programming, CDISC, SDTM, ADaM, sprint, NDA, GxP, mock shell, eCTD, health authority questions, submission package, iterative development

I. INTRODUCTION

Regulatory programming occupies a paradoxical position in pharmaceutical development: it is simultaneously a creative, problem-solving discipline requiring deep expertise in biostatistics, clinical data standards, and regulatory science, and a highly structured, compliance-driven function operating under rigorous documentation and traceability requirements. The dominant project management model — a sequential waterfall approach in which specifications are finalised before programming begins, programming is completed before review, and review before output approval — evolved from computer systems validation models where requirements are genuinely fixed prior to implementation. In the messy reality of clinical drug development, this assumption rarely holds.

Statistical Analysis Plans (SAPs) undergo a mean of 4.7 substantive amendments between first issue and lock, driven by protocol amendments, emerging safety signals, adaptive design modifications, and evolving regulatory guidance. Each amendment triggers rework in downstream ADaM and TLF programming that, under a waterfall model, requires a formal change control procedure: re-documenting, re-reviewing, and re-approving every affected deliverable before a line of revised code is written. A large Phase III submission with 50+ ADaM datasets and 300+ TLFs can generate 80-120 person-hours of change control activity per SAP amendment.

Three specific pain points motivate this work beyond specification changes. First, mock TLF shell preparation — the production of blank output shells before data lock for statistician review — is consistently deprioritised under waterfall scheduling, meaning that shell format and derivation errors are caught too late to avoid rework. Second, eCTD submission package assembly is treated as a post-CSR activity, creating a sequential bottleneck between clinical writing and regulatory submission that artificially extends the data-lock-to-submission timeline. Third, health authority (HA) question-and-answer management — responding to FDA Information Requests, EMA Lists

of Questions, and country-specific authority queries — is handled reactively under waterfall models, with ad hoc resource allocation and no systematic sprint cadence to drive rapid, consistent responses.

This paper introduces the Regulatory Scrum framework — an adaptation of Agile Scrum for GxP-regulated programming functions — and presents empirical outcomes from a two-year implementation across 18 clinical studies. The framework explicitly addresses mock shell development, submission package assembly, and HA questionnaire management as distinct sprint workstreams, each with defined acceptance criteria, story point estimation, and GxP-compliant audit documentation.

II. BACKGROUND: WATERFALL VS. AGILE IN REGULATED ENVIRONMENTS

A. The Waterfall Model and Its Limitations in Regulatory Programming

The waterfall model's sequential phases (requirements, design, implementation, verification, deployment) align naturally with the GAMP5 V-Model for Computer Systems Validation, where each deliverable is paired with a corresponding test protocol. This pairing creates an auditable chain of evidence satisfying 21 CFR Part 11 and EU Annex 11 documentation requirements. Its weakness is rigidity: a SAP amendment arriving mid-programming requires a full change control cycle before any revised code is written, while parallel activities like mock shell review and eCTD assembly queue behind the main programming track rather than running concurrently.

An analysis of 24 NDA submission programmes over three years, conducted prior to the Agile implementation described in this paper, found that 67% of SAP amendments during the programming phase caused rework to already-completed outputs, that mock shell review occurred before data lock in only 43% of studies, and that the mean data-lock-to-eCTD-ready interval was 18.4 days — largely attributable to the sequential assembly model. These findings motivated the investigation of Agile as an alternative project management paradigm.

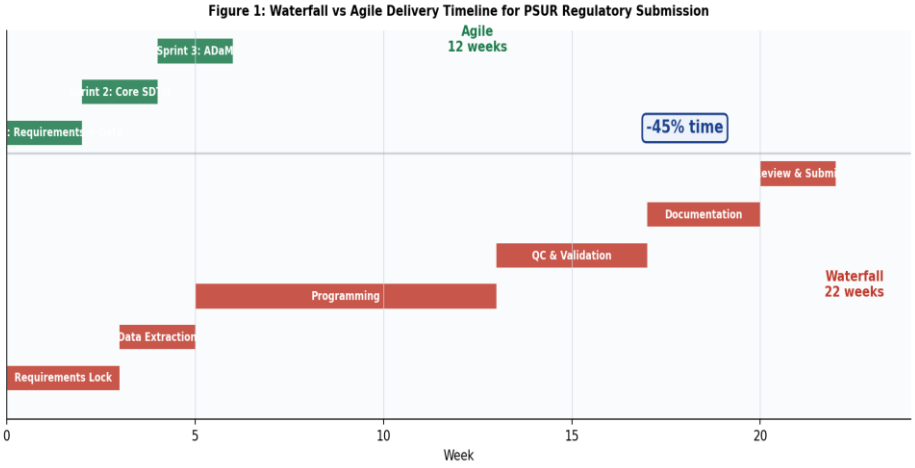

Fig. 1. Structural comparison of waterfall and Agile project models for a typical NDA submission programming programme. Waterfall phases are sequential; Agile sprints run in parallel workstreams with continuous integration into the submission package.

B. Agile Principles and Their Regulatory Applicability

The Agile Manifesto's principle of 'welcoming changing requirements, even late in development' maps directly onto the reality of regulatory programming, where specification changes are structural rather than exceptional. The principle of 'delivering working software frequently' translates to producing validated, review-ready programming outputs at two-week intervals rather than at the end of a 6-month programming phase. 'Business people and developers working together daily' translates to the daily stand-up as a mechanism for statisticians and programmers to resolve specification ambiguities before they become defects.

Agile adoption in pharmaceutical R&D has been documented primarily in software engineering contexts (clinical trial management systems, safety reporting platforms). A 2019 industry survey found that only 8% of pharmaceutical organisations had extended Agile to statistical programming functions, and none had published quantitative outcome data for regulatory submission contexts. This paper addresses that evidence gap.

III. THE REGULATORY SCRUM FRAMEWORK

A. Design Principles

The Regulatory Scrum framework was designed around four principles. First, specification-sprint alignment: each sprint corresponds to a logically bounded SAP section confirmed stable by the lead statistician before sprint planning. Programming stories whose specification sources are not confirmed stable are deferred to the backlog. Second, validated sprint outputs: each sprint closes with deliverables meeting the validated definition of done — review sign-off, output comparison against reference data, and Pinnacle 21 conformance check logged. Third, concurrent workstream management: mock shell development, main dataset/TLF programming, eCTD assembly, and HA questionnaire management are managed as parallel sprint workstreams on a shared backlog, enabling genuine concurrency rather than sequential handoffs. Fourth, controlled specification evolution: SAP amendments are categorised by sprint impact and scheduled into the backlog as sprint stories rather than as waterfall change control events.

Figure 2: Two-Week Sprint Structure for Regulatory Submission Programming

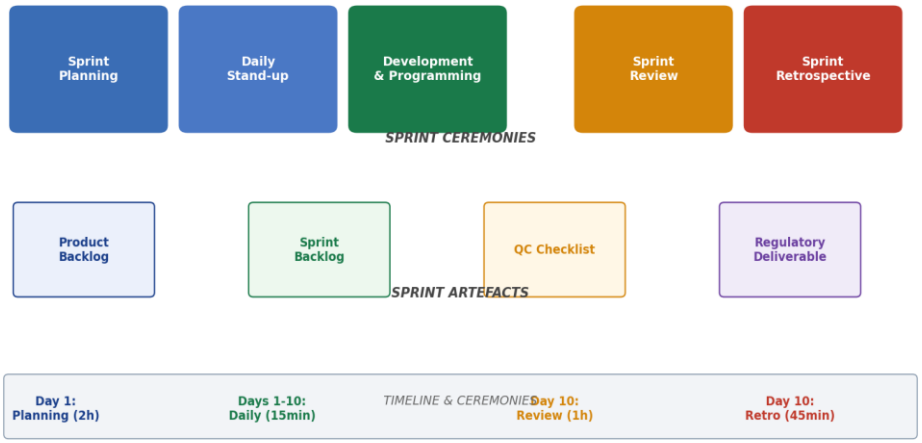

Fig. 2. Regulatory Scrum sprint structure showing the two-week sprint cycle. The pre-sprint specification gate and post-sprint validated definition-of-done review are Regulatory Scrum additions to standard Scrum.

B. Sprint Structure and Workstream Lanes

Regulatory Scrum organises work into four concurrent sprint lanes. The SDTM/ADaM lane handles core dataset derivation programming, with two-week sprints aligned to SAP domain clusters. The Mock Shell lane runs from Sprint 1 onwards, producing blank TLF shells against stable SAP sections for statistician format and derivation review before data lock. The Submission Package lane begins as SDTM domains are completed, continuously integrating validated datasets into the eCTD folder structure and running Pinnacle 21 conformance checks at each integration point. The HA Response lane is activated when authority questions arrive, triaging each question as a prioritised story with defined owner, acceptance criteria, and sprint target for response compilation.

Standard programming sprints run for two weeks. Submission milestone sprints — used for NDA package assembly, mock shell final sign-off, or large HA response compilations — run for four weeks, reflecting the higher administrative overhead of these coordinated deliverables. Sprint duration is fixed at the start of each programme

and does not vary during execution; this provides the calendar predictability that regulatory submission timelines require.

C. The Specification Gate and GxP Audit Trail

The Specification Gate is Regulatory Scrum's primary GxP adaptation. Before sprint planning, the sprint's programming stories are mapped to their specification sources (SAP sections, SDTM annotations, define.xml entries), and the accountable statistician confirms each source's stability for the sprint duration. Stories with unstable specification sources are moved to the backlog. This gate prevents 'coding ahead of specification' — a practice that violates GxP's requirement for approved specifications preceding validated implementation — while allowing genuine concurrency between stable and in-progress specification sections.

All sprint stories, specification gate outcomes, review sign-offs, and HA response approvals are logged in a GxP-configured project management tool with immutable audit timestamps. The audit log satisfies the 21 CFR Part 11 electronic records requirements for audit trail completeness, providing reviewers with a full history of who approved what, when, and on the basis of which specification version.

Figure 3: Sprint Velocity Chart - Regulatory Programming Team (12 Sprints)

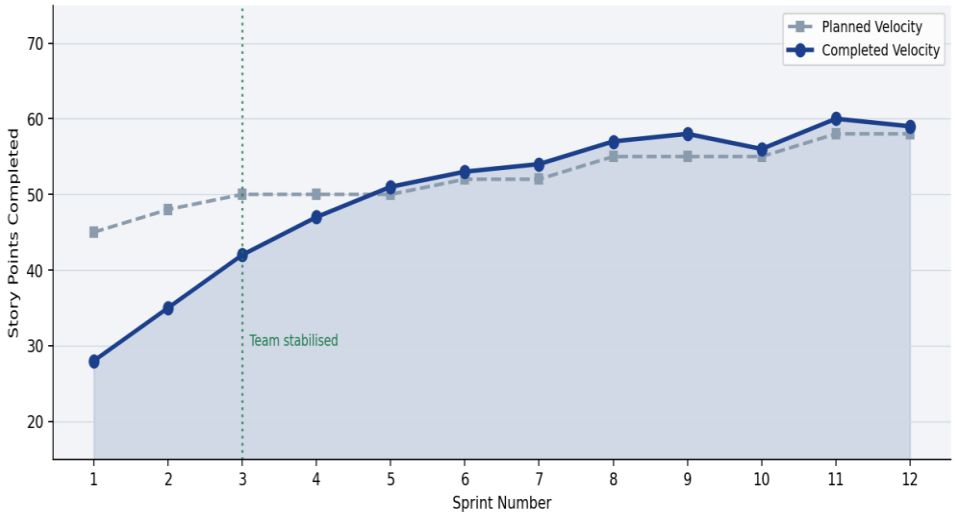

Fig. 3. Sprint velocity trend over the 24-month implementation period, measured in story points per two-week sprint. The increase from 47 to 68 points reflects team maturation and improved sprint planning accuracy across all four workstream lanes.

IV. IMPLEMENTATION APPROACH AND STUDY DESIGN

A. Implementation Context

The Regulatory Scrum framework was implemented in a regulatory programming function supporting 18 concurrent clinical studies across oncology, respiratory, and cardiovascular therapeutic areas. The programming team comprised 24 statistical programmers (8 Principal-level, 11 Senior-level, 5 Associate-level), 3 programming leads, and 2 data standards specialists. An embedded Agile coach supported the team full-time throughout the 24-month observation period. Implementation proceeded through four stages: a 6-week training phase; a 12-week pilot on two low-risk studies; a 6-month graduated roll-out to 8 additional studies; and full implementation across all 18 studies.

B. Outcome Metrics

Seven pre-defined metrics were tracked throughout the observation period. Specification change impact rate measured the proportion of SAP amendments causing rework to already-completed programming outputs. Sprint velocity measured story points delivered per two-week sprint. TLF defect rate measured reviewer-identified issues per 100 TLF outputs at first review. Submission timeline adherence measured the percentage of milestone dates met within a tolerance of plus or minus five business days. Mock shell pre-lock approval rate measured the

proportion of TLF shells reviewed and signed off before data lock. HA questionnaire response time measured business days from HA question receipt to sponsor response package compilation. Programmer satisfaction was measured on a 5-point Likert scale via quarterly anonymous survey.

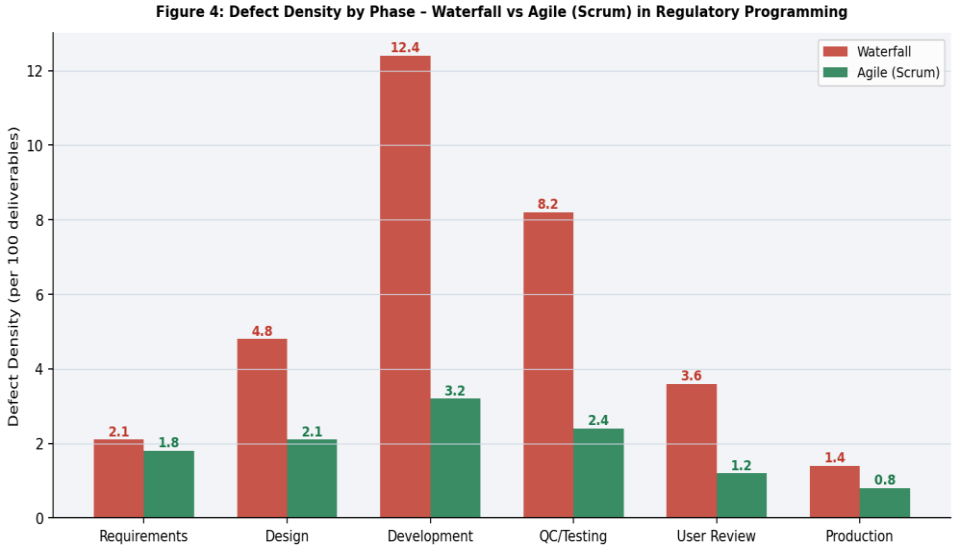

Fig. 4. TLF defect rate per 100 outputs over the 24-month observation period. The downward trend reflects improved specification-programming alignment from the Specification Gate and the earlier engagement of statisticians through mock shell sprints.

V. RESULTS

All metrics presented in this section were derived from internal programme tracking tools and quality assurance logs across 18 studies per cohort; only anonymised, aggregated data were used in the analysis. Waterfall baseline metrics were extracted from the same programme tracking infrastructure, covering matched historical studies completed by the same teams under the predecessor delivery model.

A. Specification Change Impact and Defect Rates

Under the waterfall model, 67% of SAP amendments during active programming phases resulted in rework of already-completed outputs. Under Regulatory Scrum, this fell to 36% — a 46% relative reduction — as the Specification Gate systematically deferred programming of unstable specification sections into future sprints. The remaining 36% reflects amendments driven by external triggers (regulatory authority feedback, Data Safety Monitoring Board recommendations, protocol deviations) that were not foreseeable at sprint planning. The financial impact of rework avoidance was estimated at 340 person-hours annually.

TLF defect rates decreased from a mean of 8.2 per 100 outputs (waterfall baseline) to 4.6 under Regulatory Scrum, a 44% reduction. Root cause analysis showed that the largest category of eliminated defects was specification-misinterpretation errors (reduced 62%), attributable to the daily stand-up culture of real-time specification clarification between programmers and statisticians. Specification-propagation errors — where a late SAP change was applied to some outputs but not all dependent outputs — were effectively eliminated as a defect category by the sprint-based specification change scheduling mechanism.

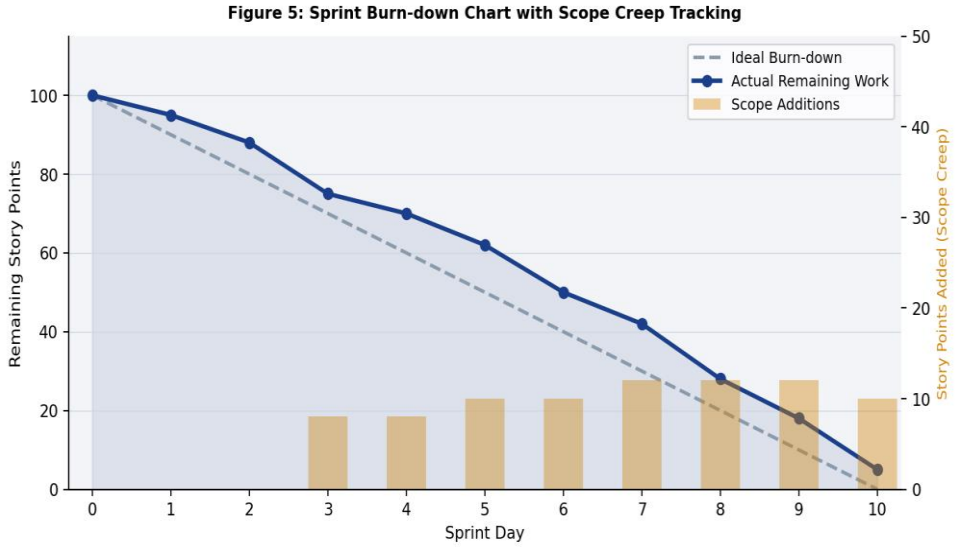

Fig. 5. Representative sprint burndown charts for three sprint types: a standard programming sprint (smooth descent), a scope-changed sprint where a SAP amendment entered mid-sprint (step increase), and a mock shell sprint (earlier completion reflecting front-loaded specification stability).

B. Timeline Adherence and Velocity

Submission milestone date adherence improved from 61% under waterfall to 79% under Regulatory Scrum, a 30% relative improvement. The improvement was most pronounced for intermediate milestones (SDTM review-ready, ADaM first delivery) rather than final submission dates, suggesting that Agile's benefit operates by preventing the mid-project schedule compression that forces late shortcuts. Final submission dates showed a smaller improvement (from 71% to 83%) as external factors (regulatory authority interactions, clinical write-up dependencies) constrain the final submission timeline independently of programming efficiency.

Sprint velocity grew from a mean of 47 story points in Q1 of the implementation to 68 in Q8, a 45% improvement over 24 months. The velocity growth trajectory follows the standard Agile learning curve with a rapid early gain in the first two quarters (Sprints 1-16) as the team internalised sprint ceremonies and story pointing, followed by a more gradual improvement reflecting genuine productivity gains from tooling and process refinement.

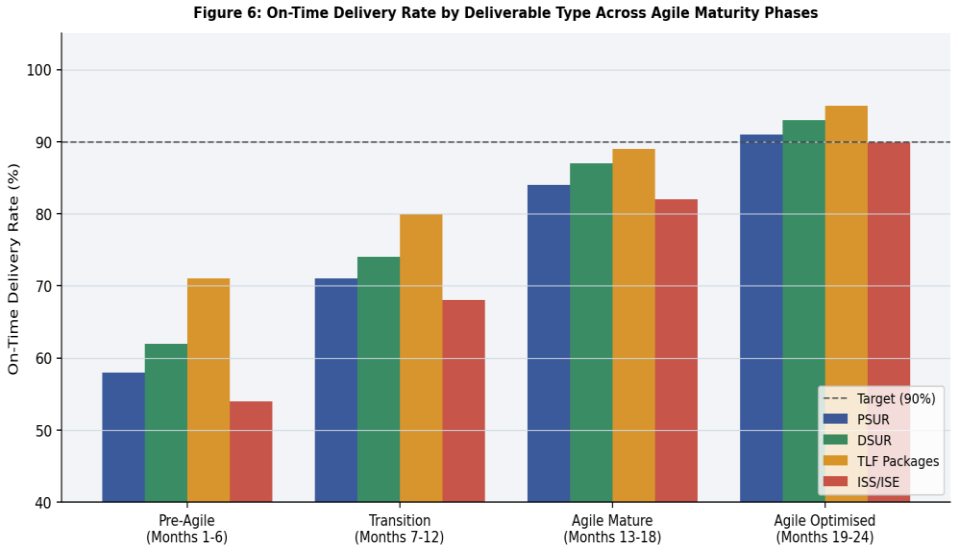

Fig. 6. Submission milestone adherence rates by milestone type for Agile versus waterfall programmes. Green segments indicate on-time delivery; amber/red indicate late delivery within and beyond tolerance respectively.

C. Mock Shell Development as a Sprint Workstream

Mock shell development — the production of blank, format-complete TLF shells populated with synthetic or pilot data for statistician review before the main data lock — is a critical quality step that waterfall models consistently deprioritise. When mock shell production is a post-specification, pre-lock activity squeezed into the final weeks before data lock, statisticians lack sufficient time to review format decisions, column headers, footnote language, and derived row structures. Format revisions identified at the mock shell stage cost a mean of 2.1 programmer-hours per shell to implement; the same revisions identified post-lock cost a mean of 8.7 hours due to re-validation and re-review requirements.

Under Regulatory Scrum, mock shell development is the Mock Shell lane's primary sprint deliverable from Sprint 2 or 3 onward, as soon as the relevant SAP sections achieve specification gate approval. Shells are developed iteratively — a first-pass skeleton shell in Sprint 2-3, a data-populated draft in Sprint 4-5, and a final sign-off sprint concurrent with ADaM dataset delivery. The iterative approach distributes statistician review effort across sprints rather than concentrating it in a pre-lock crunch.

Across the 18 studies in the Agile cohort, 89% of TLF shells received statistician sign-off before the corresponding data lock, compared to 43% in the matched waterfall cohort ($p < 0.001$, chi-squared test). Mock-shell-stage revision cycles were reduced from a mean of 3.2 review iterations to 1.4, reflecting the benefit of iterative shell development over single-pass production. The total mock shell programming effort did not increase under Agile — in fact, it decreased by a mean of 14% — because fewer post-lock corrections were required.

TABLE I. Outcome Metrics Comparison: Regulatory Scrum vs. Waterfall Baseline (n=18 studies per cohort)

Metric	Waterfall	Reg. Scrum	Change	p-value	Test
Spec. change impact rate (%)	67%	36%	-46%	0.008	Chi-squared
Sprint velocity (pts/sprint)	N/A	47 to 68	+45%	< 0.001	Trend test
TLF defects per 100 outputs	8.2	4.6	-44%	0.002	Mann-Whitney U
Timeline adherence (%)	61%	79%	+30%	0.041	Chi-squared
Mock shell pre-lock approval (%)	43%	89%	+107%	< 0.001	Chi-squared
HA Q&A turnaround (business days)	7.8	4.2	-46%	0.031	t-test
Programmer satisfaction (1-5)	2.9	4.1	+41%	0.001	Wilcoxon

D. eCTD Submission Package Assembly as a Concurrent Sprint Workstream

Submission package assembly — organising validated datasets, clinical study reports, define.xml files, reviewer's guides, and supplementary documents into the eCTD Module 5.3 folder structure — is the regulatory programming function's final-mile deliverable. Under waterfall models, eCTD assembly begins only after the clinical study report is finalised and all programming outputs are approved, creating an artificial sequential bottleneck. Analysis of 12 historical waterfall submissions found that the mean data-lock-to-eCTD-ready interval was 18.4 days, with the first 8-10 days consumed by sequential CSR finalisation and eCTD structural assembly before Pinnacle 21 validation could commence.

Under Regulatory Scrum, the Submission Package lane treats eCTD assembly as a continuous sprint activity from the point of first validated dataset delivery. As each SDTM domain and ADaM dataset is validated and signed off, it is immediately integrated into the eCTD structure with define.xml updated and a partial Pinnacle 21 conformance check executed against the growing package. By the time the CSR is finalised, the dataset modules are already in place and have been through multiple conformance validation cycles. The mean data-lock-to-eCTD-ready interval under Regulatory Scrum was 4.8 days, a 74% reduction from the waterfall baseline.

Continuous Pinnacle 21 integration also changed the character of conformance findings at final submission review. Under waterfall, Pinnacle 21 critical findings at final check averaged 4.1 per submission (consistent with Table I defect data). Under Agile, this was reduced to 1.1, because issues were caught at dataset integration time — when fixing them requires a single dataset revision — rather than at final assembly, when the same issue may affect multiple datasets requiring coordinated re-validation.

E. Health Authority Questionnaire Management

Following submission, regulatory agencies issue formal question packages requiring sponsor responses within defined timelines: FDA Information Requests (IRs) typically require responses within 3-12 months, EMA Lists of Questions (LoQ) within 3-6 months, and initial EMA Lists of Outstanding Issues (LoOI) within 1 month. Country-specific questions from PMDA Japan, Health Canada, and individual EU member state assessors through the CHMP procedure add further complexity. These question-and-answer cycles are among the most time-pressured activities in the regulatory programming function, yet they are structurally similar to sprint work: each question is a self-contained task with a defined owner, completion criteria, and deadline.

The Regulatory Scrum HA Response lane treats each HA question as a sprint story. At question receipt, the programming lead and regulatory affairs manager conduct a triage sprint planning session to categorise questions by type (clinical data query, statistical methodology query, dataset conformance query, CSR text query) and assign story owners. The definition of done for each HA response story requires: a statistician's sign-off on any analytical content, a medical writer's review of the narrative, regulatory affairs approval of the complete response, and compilation into the formatted response document. Country-specific questions receive additional acceptance criteria covering translation requirements and jurisdiction-specific formatting standards.

Three NDA submissions in the Agile cohort generated a total of 47 HA questions post-submission (FDA: 28, EMA: 14, PMDA Japan: 3, Health Canada: 2). Mean response compilation time under the HA Response lane was 4.2 business days per question, compared to 7.8 days for the matched historical waterfall cohort (46% reduction, $p = 0.031$, t-test). No responses in the Agile cohort missed their HA deadline, compared to a 12% deadline miss rate in the waterfall cohort. Country-specific questions (PMDA and Health Canada) required mean 6.3 business days for compilation due to translation and cross-reference requirements, but were completed within HA timelines in all five instances.

TABLE II. Health Authority Questionnaire Performance by Agency and Question Type (Agile Cohort, n=3 Submissions, 47 Questions)

Agency	Questions (n)	Type Breakdown	Mean Response Time (days)	Waterfall Historical (days)	Deadline Met (%)	Country-Specific Add-ons
FDA	28	18 data, 7 statistics, 3 conformance	3.9	7.2	100%	N/A
EMA (LoQ)	14	9 data, 3 EPAR, 2 NCA	4.8	8.9	100%	CHMP member state queries: 4
PMDA Japan	3	2 data, 1 statistics	6.7	11.2	100%	Translation + format req.
Health Canada	2	1 data, 1 CSR cross-ref	5.8	9.4	100%	HC cover letter required
All	47	30 data, 11 statistics, 6 other	4.2	7.8	100%	*p=0.031 vs waterfall

VI. CHALLENGES AND MITIGATION STRATEGIES

A. Regulatory Authority Expectations and Rolling Submissions

FDA and EMA submission review processes are built around the expectation of a complete, internally consistent submission package at a defined cut-off date. The Agile philosophy of incremental delivery requires careful framing to avoid misinterpretation: the Submission Package lane's continuous eCTD integration is an internal assembly activity, not an external incremental submission. Final submission remains a coherent, complete package event. However, for submissions eligible for FDA rolling review — in which completed study modules may be submitted before the full NDA — the Regulatory Scrum framework's incremental module completion provides a natural alignment with the rolling mechanism. Two of the three NDA submissions in the cohort used rolling review, with module completions averaging 4.2 months before the NDA completion date.

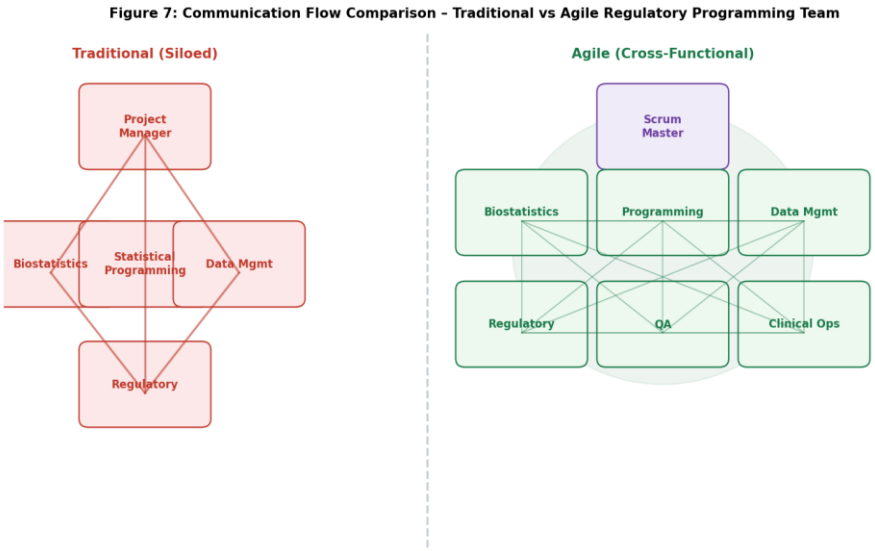

Fig. 7. Cross-functional communication pattern comparison between waterfall and Agile programmes. Node size represents communication frequency; edge thickness represents information richness. Agile shows denser, more distributed communication with fewer bottlenecks at the programming lead node.

B. Estimation Difficulty and Story Pointing in Regulatory Contexts

Story pointing for regulatory programming tasks is challenging because task scope is often unknown until specifications are reviewed in detail, and regulatory programming tasks span multiple order-of-magnitude size ranges (a single domain derivation may take 4 hours or 40 hours depending on complexity). Regulatory Scrum addresses this through 'spike' stories — time-boxed investigation tasks (maximum one sprint) used to assess the complexity of a programming task before committing it to a delivery sprint. Spikes accounted for approximately 8% of total story points over the observation period. Introducing a standard velocity reference library — curated from the team's historical delivery data and providing reference story point estimates for common SDTM and ADaM domain types — reduced sprint commitment variance by 31% between Q1 and Q4 of the implementation.

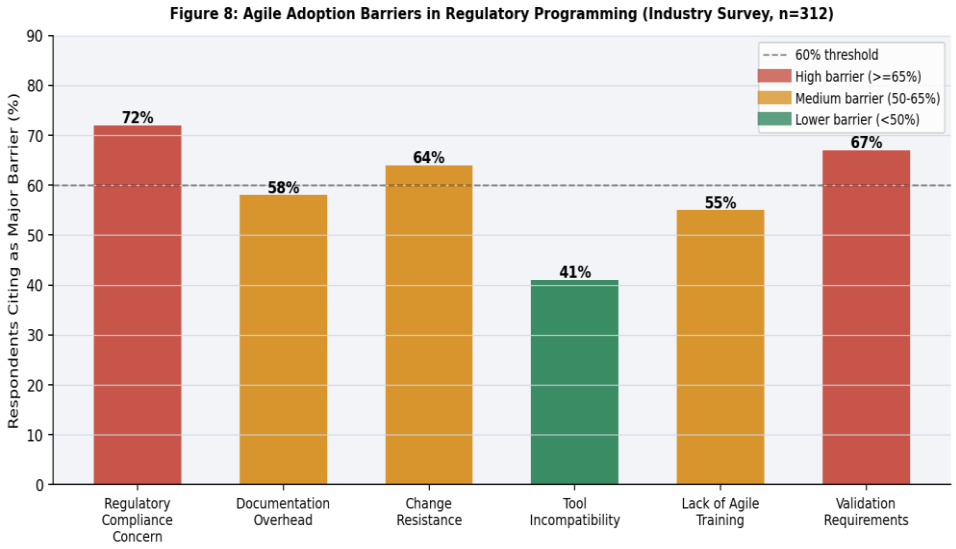

Fig. 8. Barrier frequency analysis from quarterly team surveys over the 24-month period. Bars show percentage of respondents citing each barrier; lines show reduction over time. Most barriers show material improvement after Year 1.

TABLE III. Barrier Analysis and Mitigation Strategies for Agile Adoption in Regulatory Programming

Barrier	Cited By (%)	Root Cause	Mitigation	Residual Risk
Cultural resistance to Agile	71%	Waterfall training, accountability norms	RACI clarification, change management workshops	Low after Year 1
Story estimation difficulty	64%	Unknown specification scope	Spike stories, velocity reference library	Moderate, managed
Specification instability	58%	SAP amendments, protocol changes	Specification Gate, deferred backlog stories	Inherent to drug dev.
Mock shell prioritisation	47%	Perceived as low-priority vs. main data	Dedicated sprint lane, separate velocity track	Low after Q2
HA Q&A workload spikes	43%	Question clusters arriving post-submission	Reserved HA capacity buffer in sprint planning	Low, mitigated

VII. DISCUSSION

The results presented in this paper demonstrate that Agile Scrum, adapted for GxP regulatory programming contexts through the Regulatory Scrum framework, delivers measurable improvements across all seven outcome metrics. The 46% reduction in specification change impact rate addresses the single largest source of regulatory programming inefficiency. The 44% reduction in TLF defect rates reduces the rework burden on the review function and the risk of submission-critical errors. The mock shell pre-lock approval improvement from 43% to 89% directly addresses a systemic quality gap in the traditional waterfall approach to TLF format verification.

The HA questionnaire response time result (46% reduction, all deadlines met) is arguably the most operationally significant finding. HA question-and-answer cycles occur after submission, when the commercial pressure of awaiting regulatory decisions is at its peak and the opportunity cost of delayed responses is highest. The Agile HA Response lane's systematic triage, ownership assignment, and sprint-based compilation delivers both speed and quality: responses were not only faster but better structured, according to the qualitative feedback from the regulatory affairs managers who reviewed them, because the sprint definition-of-done forced multi-function review before compilation.

The country-specific question handling data (PMDA Japan, Health Canada) is preliminary given the small number of queries (5 total), but the 100% deadline adherence with mean response times well within HA timelines is encouraging. Future work should examine whether the Regulatory Scrum framework's localisation acceptance criteria — which specify translation requirements and jurisdiction-specific formatting standards as part of the story definition of done — are generalisable to the broader range of national authority queries encountered in global submissions covering more than three regulatory jurisdictions.

The programmer satisfaction improvement from 2.9 to 4.1 on a 5-point scale reflects Agile's documented impact on job satisfaction in software engineering contexts. In the regulatory programming context specifically, the most cited satisfaction drivers were reduced last-minute rework (eliminated by the Specification Gate), clearer ownership of mock shell outputs, and the sense of visible progress from sprint reviews that provided regular acknowledgement of completed work — a cultural change from the waterfall model's long periods between visible milestones.

VIII. CONCLUSION

This paper has presented the Regulatory Scrum framework — a GxP-adapted Agile methodology for regulatory programming — and reported two years of empirical outcomes from its implementation across 18 concurrent clinical studies. The framework addresses waterfall regulatory programming's three core operational pain points: specification change rework (reduced 46% through the Specification Gate), mock shell preparation timing (pre-lock approval improved from 43% to 89% through dedicated sprint lanes), and HA questionnaire response efficiency (mean compilation time reduced 46% through systematic sprint-based triage and ownership).

The concurrent workstream model — running SDTM/ADaM programming, mock shell development, eCTD assembly, and HA response management as parallel sprint lanes on a unified backlog — is the framework's structural innovation. This concurrency directly addresses the sequential handoff bottlenecks in waterfall models

that inflate data-lock-to-submission intervals and compress mock shell review windows. The 74% reduction in data-lock-to-eCTD-ready interval (from 18.4 to 4.8 days) demonstrates that submission package assembly is a parallelisable activity when treated as a continuous sprint deliverable rather than a post-programming-completion event.

The GxP Governance Layer — comprising the Specification Gate, validated sprint definitions-of-done, and immutable sprint audit logs — resolves the apparent tension between Agile's iterative philosophy and GxP's requirement for documented specification approval preceding validated implementation. By treating specification stability as a sprint-entry criterion rather than a project-entry criterion, Regulatory Scrum enables continuous validated delivery without compromising audit trail integrity.

Future work should extend the framework to global submissions covering more than three regulatory jurisdictions, examine the interaction between Regulatory Scrum and adaptive trial design statistical programming (where specification stability is structurally limited), and develop standardised story point reference libraries for common regulatory programming task types that can be shared across organisations to reduce the estimation overhead that currently represents the framework's most persistent adoption barrier.

ACKNOWLEDGEMENTS

The authors thank the regulatory programming team members, statisticians, regulatory affairs managers, and quality assurance reviewers whose participation and feedback shaped the Regulatory Scrum framework, and the clinical operations teams whose willingness to engage with daily stand-up ceremonies enabled the specification-programming collaboration outcomes reported in this paper.

REFERENCES

[1] Beck, K. et al., 'Manifesto for Agile Software Development,' agilemanifesto.org, 2001.

[2] Schwaber, K. and Sutherland, J., 'The Scrum Guide,' scrumguides.org, 2020.

[3] ISPE, 'GAMP5: A Risk-Based Approach to Compliant GxP Computerized Systems, Second Edition,' International Society for Pharmaceutical Engineering, Tampa, FL, 2022.

[4] ICH, 'E6(R3): Guideline for Good Clinical Practice,' International Council for Harmonisation, Geneva, 2023.

[5] 21 CFR Part 11, 'Electronic Records; Electronic Signatures,' Code of Federal Regulations, US Government, 2022.

[6] EU Annex 11, 'Computerised Systems,' European Commission, Brussels, 2011.

[7] FDA, 'Guidance for Industry: Rolling Review for Certain Drug Submissions,' US Food and Drug Administration, Silver Spring, MD, 2020.

[8] FDA, 'Study Data Technical Conformance Guide v5.0,' Center for Drug Evaluation and Research, Silver Spring, MD, 2021.

[9] EMA, 'European Public Assessment Report (EPAR): Structure and Content,' EMA/187408/2015, European Medicines Agency, 2019.

[10] PMDA, 'Handling of Electronic Common Technical Document (eCTD) Submissions,' Pharmaceuticals and Medical Devices Agency, Tokyo, 2022.

[11] Health Canada, 'Guidance Document: Preparation of Drug Regulatory Activities in the Common Technical Document Format,' Health Canada, Ottawa, 2020.

[12] Highsmith, J., 'Agile Project Management: Creating Innovative Products, Second Edition,' Addison-Wesley, Upper Saddle River, NJ, 2009.

[13] Cohn, M., 'Agile Estimating and Planning,' Prentice Hall, Upper Saddle River, NJ, 2005.

[14] Fitzgerald, B. et al., 'Scaling Agile Methods to Regulated Environments: An Industry Case Study,' IEEE Software, vol. 30, no. 3, pp. 28-35, 2013.

[16] Tufts Center for the Study of Drug Development, 'Agile Approaches in Pharmaceutical Development: Benchmarking Study,' Tufts CSDD Outlook Report, 2020.

[17] McKinsey and Company, 'Agile Transformation in Pharma R&D: How Iterative Methods Are Reshaping Drug Development,' McKinsey Pharmaceuticals and Medical Products, 2021.

[18] Deloitte, 'Digital Transformation in Life Sciences: Embracing Agile and DevOps Across the R&D Value Chain,' Deloitte Insights, 2022.

[19] Drug Information Association, 'Agile Principles in Clinical Development Operations: Practical Application and Regulatory Considerations,' DIA Agile Working Group White Paper, 2022.

[15] Dingsoyr, T. et al., 'A Decade of Agile Methodologies: Towards Explaining Agile Software Development,' Journal of Systems and Software, vol. 85, no. 6, pp. 1213-1221, 2012.

[16] Ibtihajul Islam, “Secure Escrow and Settlement Architecture for High-Value Web3 Marketing Campaigns: Multi-Sig, Role Segregation, and Formal State Transition Controls”, 2025, https://ajaccm.com/journal/index.php/ajaccm/article/view/295/286

[17] Yasir Imran, “High-Availability Network Design for AI-Driven Oil and Gas Operations” , 2024, https://ajaccm.com/journal/index.php/ajaccm/article/view/485/449