Operationalizing Percentile-Based Agile Release Forecasting Without Story Points

Berk Kibarer – berkkibarer@gmail.com
Independent Researcher
An Industrial Experience Report | May 2026

Abstract

Many agile organizations struggle to translate internal delivery progress into reliable stakeholder-facing release commitments. In the observed organization, release planning discussions frequently revolved around story-point calibration and velocity interpretation, while uncertainty itself remained difficult to communicate outside engineering teams. To address this problem, the organization gradually introduced a lightweight forecasting routine based on direct effort logging and percentile-based release forecasting.

The approach combined weekly effort tracking, export-based extraction from Jira and Trello, and Monte Carlo simulation to generate probabilistic release forecasts. External release commitments were anchored to the 90th percentile forecast rather than optimistic single-date targets. The routine was used operationally for approximately 170 weeks across 16 release cycles in a large product organization operating in the communications domain.

This paper focuses on the operationalization of the forecasting process rather than statistical novelty. It describes how the routine was introduced, how forecasting outputs became part of recurring governance discussions, what forms of organizational resistance emerged during adoption, and how the forecasting system was integrated into release management practices. Because detailed industrial forecast logs were no longer accessible after the author left the organization, the paper reports aggregated operational observations rather than full industrial datasets.

The experience suggests that percentile-based governance can provide a practical alternative to story-point-centered release commitments when organizations are willing to maintain lightweight operational discipline and communicate uncertainty explicitly.

Keywords
Agile forecasting; Monte Carlo simulation; release governance; uncertainty management; industrial experience report; software delivery

1. Introduction

Software delivery organizations are frequently expected to provide calendar-based commitments while internally managing delivery through abstract estimation systems such as story points. In practice, this often creates communication friction. Engineering teams discuss velocity trends and relative sizing, while stakeholders ask for dates, delivery confidence, and schedule risk.

The work described in this paper emerged from repeated operational problems in release governance meetings inside a large product organization. Delivery discussions regularly shifted toward defending estimation scales instead of discussing uncertainty and operational risk. Forecasts existed, but confidence around those forecasts remained difficult to communicate consistently across engineering and non-engineering stakeholders.

Over time, the organization gradually moved away from story-point-centered release forecasting and introduced a simpler operational process. Work effort was tracked directly in person-days, throughput histories were extracted from operational systems, and release forecasts were generated using percentile distributions instead of deterministic target dates.

The most important organizational change was not the simulation model itself. The larger shift involved anchoring external commitments to conservative percentile forecasts rather than optimistic “most likely” delivery dates. This changed how release discussions were conducted, how uncertainty was communicated, and how scope tradeoffs were negotiated during delivery.

This paper reports that experience from an operational perspective. The contribution is not a new forecasting algorithm, but a description of how a lightweight percentile-based forecasting process was introduced, stabilized, and incorporated into recurring release governance.

2. Organizational Context

The forecasting routine was developed within a large product-based organization operating in the communications sector. The team maintained a central test automation platform serving multiple internal delivery groups. The organization is intentionally anonymized because operational release history and internal delivery data cannot be disclosed publicly.

The core delivery team typically consisted of 9–12 engineers, including developers, testers, and developer-in-test roles. Work followed a weekly planning cadence and releases generally occurred every two to three months.

Jira was used for backlog and workflow management, while Trello was used for operational bookkeeping and effort logging. Instead of introducing heavy process customization into existing tools, the organization adopted an export-based integration model. Weekly exports were normalized into a compact dataset containing completed effort, work-item state, completion timing, and indicators for unplanned work.

The forecasting routine was not introduced as a formal organizational transformation initiative. It started as a lightweight reporting mechanism intended to improve release visibility within a single operational

area. Over time, the reports became part of recurring release governance discussions and gradually influenced stakeholder expectations around release commitments.

3. Forecasting Model and Operational Flow

The forecasting process operated as a recurring sprint-level routine consisting of four main stages:

1. extraction of operational data from Jira and Trello,
2. normalization and validation of sprint records,
3. simulation of future completion dates,
4. governance review and release discussion.

The operational overhead remained intentionally small. Data extraction and preprocessing were automated, while the governance review generally required less than one hour each sprint.

The model was based on actual logged effort rather than relative story-point estimation. At release start, the total release effort was declared in person-days. During each sprint, the system computed completed effort, unplanned work additions, and effective throughput. Remaining release effort was updated after each sprint using:

remaining_effort = previous_remaining_effort − (velocity − scope_added)

where velocity represented completed logged effort during the sprint and scope_added represented work introduced after sprint start.

Throughput was normalized by effective working days:

throughput_pd_daily = (velocity − scope_added) / effective_days

This normalization allowed the model to compare sprints of different lengths while accounting for weekends and holidays.

A simplified throughput formulation used during simulation is shown below.

$$\textbf{throughput}_t = \beta_0 + \beta_1(\textbf{prev\_daily\_rate}_{t-1}) + \beta_2(\textbf{unplanned\_fraction}_t) + \beta_3(\textbf{percent\_bug}_t) + \varepsilon_t$$

The regression model operated on sprint-level aggregates rather than individual tasks. Historical sprint metrics were used to estimate throughput behavior and operational volatility. The forecasting system used five principal features:

- previous sprint throughput,
- unplanned work fraction,

• bug-fix ratio,
• throughput volatility,
• scope added mid-sprint.

Monte Carlo simulation was then used to generate distributions of possible release completion dates. Each simulation proceeded sprint-by-sprint. For every simulated sprint, the model sampled a plausible throughput value using both regression output and residual noise derived from historical behavior. Simulated sprint capacity was subtracted from remaining release effort until completion.

The implementation executed 5,000 simulation runs per forecast update. Percentile outputs such as P50 and P90 were then extracted from the resulting completion-date distribution.

The operational purpose of Monte Carlo simulation was not mathematical sophistication. Deterministic release dates had proven unstable under changing delivery conditions, particularly when unplanned work or operational interruptions appeared. Percentile-based forecasting allowed the organization to communicate uncertainty explicitly rather than hiding schedule risk behind unofficial buffers.

4. Operational Governance and Forecast Usage

Forecasting outputs became part of a recurring governance cycle.

At the end of each sprint, operational data was exported from Jira and validated for missing or inconsistent effort entries. Sprint metrics were then recomputed and the forecasting pipeline executed automatically. The resulting report included remaining effort, throughput trends, percentile forecasts, and operational risk indicators.

The organization primarily consumed three percentiles:

• P50 as the internal planning baseline,
• P75 for intermediate risk review,
• P90 as the external release commitment boundary.

In practice, P90 became the operational commitment target communicated to stakeholders. When the P90 forecast moved significantly later between sprint reviews, governance discussions focused on identifying root causes and determining corrective actions. Typical responses included scope reduction, stabilization work, postponement of non-critical features, or explicit schedule buffers.

The forecasting process also introduced escalation rules. Significant outward movement in P90 forecasts triggered additional release reviews. Elevated throughput volatility or unusually high unplanned work fractions were treated as operational warning signals requiring investigation.

A typical forecasting cycle followed the workflow below:

• sprint closes Friday afternoon,
• Jira/Trello export generated,
• validation and cleanup performed,
• forecasting pipeline executed,
• governance review conducted,
• stakeholder summary distributed before the next sprint begins.

The entire cycle usually completed within approximately one hour.

Feature	Definition	Operational Meaning	Typical Range
prev_daily_rate	Previous sprint throughput	Delivery momentum	0.1–15 pd/day
unplanned_fraction	scope_added / velocity	Planning disruption	0–1
percent_bug	Bug-fix effort ratio	Quality burden	0–1
throughput_cv_w	Rolling throughput volatility	Predictability	0–2+
scope_added	Mid-sprint added effort	Reactive workload	0–50+ pd

Table 1. Main operational forecasting features used in the simulation model.

5. Adoption Challenges and Organizational Friction

The technical implementation was considerably easier than the organizational adoption.

Several forms of resistance appeared during the first release cycles. Some stakeholders viewed P90 commitments as unnecessarily pessimistic because they preferred aggressive “best-case” delivery dates. Engineering teams occasionally delayed effort logging until sprint end, reducing forecast stability. In some situations, percentile forecasts were interpreted as deterministic promises instead of evolving risk indicators.

The organization addressed these issues incrementally rather than through formal process enforcement. Weekly cutoffs were introduced for missing effort entries. Forecast review became a standing item in release meetings. Over time, discussions gradually shifted away from defending estimation scales toward discussing uncertainty and delivery risk directly.

Another concern involved metric misuse. Team members worried that effort tracking could eventually become a performance evaluation mechanism. To reduce this risk, metrics were reviewed at team level rather than individual level and forecasting artifacts were excluded from personnel evaluation discussions.

The system also implemented several practical anti-gaming controls. Scope additions were extracted automatically from Jira labels rather than manually reported. Carryover work remained visible between sprints. Bug-fix ratios acted as indirect quality signals. Governance reviews investigated unusual improvements in forecast stability or throughput without corresponding operational changes.

The routine became substantially more stable after approximately three release cycles. By that stage, percentile terminology and escalation rules had become part of normal release governance discussions.

6. Observed Outcomes

Detailed industrial forecast logs were no longer accessible after the author left the organization. As a result, this paper reports aggregated operational observations rather than raw historical datasets.

Across 16 releases during the observed policy period, release completion relative to the final communicated P90 forecast generally remained within day-scale deviation ranges. Most releases were completed within several days of the communicated commitment boundary. Earlier operational periods before adoption showed substantially larger release drift, commonly measured in weeks rather than days.

The most significant operational change was not forecast precision alone. The more important change involved governance behavior. Release discussions became less focused on defending estimates and more focused on identifying uncertainty sources, stabilizing delivery conditions, and negotiating scope tradeoffs earlier in the delivery cycle.

The forecasting routine also created a shared language for discussing delivery risk. Percentile-based forecasting made uncertainty easier to communicate outside engineering teams because it provided explicit probability-oriented framing rather than abstract velocity interpretation.

The approach was not equally effective under all conditions. Forecast quality degraded during periods of major organizational disruption, high turnover, or unusually volatile delivery patterns. Teams performing highly exploratory work generated wider forecast intervals and less stable throughput histories.

Figure 1. Aggregated release slip distribution relative to the final communicated P90 commitment.

7. Discussion

Several operational lessons emerged from the experience.

First, percentile forecasting proved useful primarily because it changed release conversations. The forecasting mathematics itself was relatively simple. The larger impact came from replacing optimistic single-date commitments with explicit uncertainty ranges.

Second, lightweight operational discipline mattered more than model sophistication. Forecast quality depended heavily on consistent effort logging and stable sprint review routines. Missing or delayed operational data affected forecast reliability more than changes in simulation parameters.

Third, percentile forecasts should not be presented as guarantees of accuracy. They were most effective when treated as evolving risk indicators that helped teams discuss scope, stabilization work, and operational buffers more explicitly.

Fourth, the approach appeared better suited to environments with recurring delivery patterns than highly exploratory research-oriented work. Teams experiencing major workflow instability generated substantially wider forecast intervals and less stable throughput histories.

Finally, the forecasting routine introduced cultural changes beyond scheduling itself. Teams gradually became more comfortable discussing uncertainty explicitly instead of negotiating confidence through optimistic estimates.

8. Threats to Validity and Limitations

This paper reports the experience of a single organization and therefore should not be interpreted as a statistically generalizable evaluation.

A second limitation involves data availability. Detailed industrial forecast logs could not be published after the author's departure from the organization. The paper therefore relies on aggregated operational observations and release-level summaries.

The baseline comparison between earlier release periods and the observed policy period is observational rather than controlled. Organizational maturity, staffing changes, evolving engineering practices, and operational process improvements may also have influenced delivery behavior over time.

The forecasting model itself was intentionally lightweight. Alternative statistical approaches could potentially improve predictive performance under different operational conditions. The purpose of the work was not to demonstrate algorithmic superiority, but to evaluate whether a lightweight percentile-based governance process could function sustainably in day-to-day release management.

Finally, the forecasting process assumed reasonably stable operational delivery behavior within the rolling historical window. Highly exploratory or heavily disrupted teams may therefore experience lower

forecast stability than teams with more consistent delivery patterns.

9. Conclusion

This paper described the introduction of a percentile-based release forecasting routine in a large agile organization. The operational process combined direct effort logging, automated extraction from existing work-management systems, and Monte Carlo simulation to generate probabilistic release forecasts.

Over time, the forecasting routine became part of recurring release governance. The most important organizational change was the adoption of percentile-based commitments, particularly the use of P90 forecasts for external release communication.

The experience suggests that lightweight probabilistic forecasting can help agile organizations communicate uncertainty more effectively than traditional velocity-centered release planning discussions. At the same time, the experience also showed that organizational adoption, operational discipline, and governance behavior matter more than forecasting mathematics alone.

Future work could evaluate similar forecasting approaches across multiple organizations and compare percentile-based governance with traditional velocity-based release management under more controlled conditions.

References

1. Cohn, M. Agile Estimating and Planning. Prentice Hall, 2005.
2. Vacanti, D. Actionable Agile Metrics for Predictability. ActionableAgile Press, 2015.
3. Magennis, T. Forecasting and Simulating Software Development Projects. Focused Objective, 2011.
4. Magennis, T. When Will It Be Done? Lean-Agile Forecasting to Answer Your Customers' Most Important Question. Focused Objective, 2016.
5. Jørgensen, M., and Shepperd, M. A systematic review of software development cost estimation studies. IEEE Transactions on Software Engineering, 33(1), 33–53, 2007.
6. Hastie, T., Tibshirani, R., and Friedman, J. The Elements of Statistical Learning. Springer, 2009.
7. Efron, B., and Tibshirani, R. An Introduction to the Bootstrap. Chapman & Hall/CRC, 1994.
8. Schwaber, K., and Sutherland, J. The Scrum Guide, 2020.
9. Seabold, S., and Perktold, J. Statsmodels: Econometric and Statistical Modeling with Python. Proceedings of the 9th Python in Science Conference, 2010.
10. Harrell, F. Regression Modeling Strategies. Springer, 2015.

Data & Code Availability: Synthetic dataset generator, source code, and a sample anonymized report artifact: https://github.com/berkkibarer/agile_release_forecast | Archived version with DOI: https://doi.org/10.5281/zenodo.20055007