Invisible Work Risk Assessment Framework

A Statistical Approach for Detecting Work Not Captured in Jira

---

Executive Summary

This document describes a statistical approach for **assessing the risk of invisible work**—the likelihood that tasks and activities are occurring outside of Jira, creating blind spots in planning, capacity estimation, and delivery forecasting.

The approach combines:

1. **Context-dependent scoring** using percentile comparisons against similar teams (primary)
2. **Absolute threshold guardrails** to catch extreme cases and provide baseline standards (secondary)
3. **Weighted category aggregation** to produce an overall risk level

This methodology balances statistical rigor with practical applicability, avoiding both the oversimplification of fixed thresholds and the contextual blindness of pure percentile-based approaches.

Risk Levels:

- **High Risk** (≤33rd percentile): Significant indicators of invisible work
- **Moderate Risk** (34-66th percentile): Some concerning patterns detected
- **Low Risk** (>66th percentile): Most work appears captured in Jira

---

1. Problem Statement

1.1 What is Invisible Work?

Invisible work encompasses any task, activity, or effort that consumes team capacity but is not reflected in the team's Jira instance. This includes:

- **Untracked tasks**: Work performed but never logged (ad-hoc requests, quick fixes, informal support)
- **Offline coordination**: Discussions, decisions, and planning happening in side channels
- **Shadow work**: Unofficial projects or initiatives running parallel to tracked work
- **Retroactive logging**: Work completed before being entered into the system, often with inaccurate timestamps

1.2 Why Detection is Non-Trivial

By definition, invisible work cannot be directly observed in the data. We must therefore rely on **leading indicators**—observable patterns in the data that correlate with the presence of untracked work. These indicators are statistical proxies, not direct measurements.

1.3 The Core Statistical Insight

If all work were captured in Jira, we would expect:

- Output variability to correlate with demand variability
- Consistent tool usage patterns over time
- Work entering through defined intake channels

Deviations from these expectations—particularly **unexplained variability**—signal that factors outside the system are influencing outcomes.

---

2. Theoretical Framework

2.1 The Unexplained Variability Hypothesis

Consider a team's throughput (items completed per sprint). In a perfectly tracked system:

```
Var(Output) ≈ f(Var(Demand), Var(Capacity), Var(Complexity))
```

Where:

- **Var(Demand)** = variability in incoming work requests
- **Var(Capacity)** = variability in available team capacity (PTO, attrition, etc.)
- **Var(Complexity)** = variability in work item complexity/size

If observed output variability significantly exceeds what these factors predict, the residual variance likely stems from **unobserved factors**—i.e., invisible work entering or leaving the system.

2.2 Why Context Matters

Different operational contexts produce legitimately different baseline variability:

Context Factor	Impact on Expected Variability
Work type (R&D vs. maintenance)	R&D inherently more variable
Team maturity	New teams show higher variability
Process methodology (Scrum vs. Kanban)	Scrum batching creates artificial variance
Industry (regulated vs. startup)	Regulated environments more predictable
Team size	Smaller teams have higher per-capita variance

A fixed threshold (e.g., "CoV > 0.5 is bad") ignores these realities and produces:

- **False positives** for legitimately variable contexts
- **False negatives** for contexts where even moderate variability is anomalous

2.3 The Case for Percentile-Based Primary Scoring

Comparing a team against similar peers addresses context implicitly:

- Teams are compared to others with similar operational characteristics
- The distribution of outcomes captures what's "normal" for that context
- Percentile rankings are robust to non-normal distributions (common in software metrics)

2.4 The Case for Absolute Threshold Guardrails

Pure percentile scoring has failure modes:

- **Organizational dysfunction**: If all teams have poor practices, poor becomes "normal"
- **Insufficient comparison data**: New or unique teams may lack valid peers
- **Extreme outliers**: Some values are objectively concerning regardless of context

Absolute thresholds provide a floor of standards and catch cases that percentile scoring might miss.

---

3. Proposed Methodology

3.1 Two-Layer Scoring Framework

```
┌─────────────────────────────────────────────────────────────┐
│  Layer 1: Percentile-Based Comparison (Primary Scoring)     │
│  ─────────────────────────────────────────────────────────  │
│  Rank each indicator against comparison group.              │
│  Flag if below 30th percentile.                             │
├─────────────────────────────────────────────────────────────┤
│  Layer 2: Absolute Threshold Guardrails (Secondary)         │
│  ─────────────────────────────────────────────────────────  │
│  Independent warnings for extreme values regardless of      │
│  percentile ranking.                                        │
└─────────────────────────────────────────────────────────────┘
```

3.2 Layer 1: Percentile-Based Comparison

For each indicator:

1. **Calculate the metric** for the target team
2. **Retrieve comparison group** distribution (see Section 4)
3. **Compute percentile rank** of target team within distribution
4. **Flag if below threshold** (default: 30th percentile)

Flagging logic:

```
def is_flagged(team_value, comparison_distribution, threshold_percentile=30):
    team_percentile = percentile_rank(team_value, comparison_distribution)

    # For metrics where LOWER is BETTER (e.g., stale items %)
    if metric.lower_is_better:
        return team_percentile >= (100 - threshold_percentile)
    # For metrics where HIGHER is BETTER (e.g., daily updates)
    else:
        return team_percentile <= threshold_percentile
```

Why 30th percentile?

- Balances sensitivity (catching issues) with specificity (avoiding false alarms)
- Roughly corresponds to "below average" while allowing for normal variance
- Can be adjusted based on organizational risk tolerance

3.3 Layer 2: Absolute Threshold Guardrails

Independent of percentile ranking, flag when:

Indicator Category	Absolute Threshold	Rationale
Variability Metrics		

Throughput CoV	> 1.2	StdDev exceeding mean indicates chaos
Stage time CoV	> 1.5	Extreme process inconsistency
Member throughput CoV	> 1.0	Major workload imbalances
Usage Metrics		
Stale in-progress items	> 70%	Majority of WIP abandoned
Daily updates per item	< 0.1	Near-total disengagement
Bulk change ratio	> 40%	Predominantly retroactive logging
Intake Metrics		
Mid-sprint additions	> 60%	Majority of work bypasses planning
Siloed items	> 70%	Minimal collaboration/accountability

---

4. Comparison Group Selection

The validity of percentile-based scoring depends critically on comparison group quality. Teams should be compared against peers with similar operational characteristics.

4.1 Recommended Stratification Factors

Factor 1: Team Size

Band	Size Range	Statistical Behavior
Small	2-4 members	High individual impact, volatile
Medium	5-9 members	Moderate variance
Large	10-20 members	Smoothed, more predictable
Very Large	20+ members	May have sub-team dynamics

Factor 2: Work Type / Domain

Work Type	Expected Characteristics
Product Development	Moderate variability, sprint-based patterns
Platform/Infrastructure	Lower variability, longer cycles
Support/Operations	High variability, reactive patterns
R&D/Innovation	High variability, exploratory work
Maintenance/BAU	Low variability, predictable workload

Factor 3: Process Methodology

Methodology	Pattern Implications
Scrum	Sprint-bounded metrics, planning-based intake
Kanban	Continuous flow, WIP-limited
Hybrid	Mixed patterns
None/Ad-hoc	High variability, weak signals

4.2 Minimum Comparison Group Size

Comparison Group Size	Reliability	Recommendation
< 10 teams	Poor	Fall back to broader group or absolute thresholds
10-30 teams	Moderate	Usable with caveats
30-100 teams	Good	Reliable percentile estimates
100+ teams	Excellent	High confidence

---

5. Indicator Specifications

5.1 Category 1: Unexplained Variability Indicators (7 indicators)

These indicators detect variance in output that cannot be explained by visible input variance.

Indicator	Description	Flag When
Throughput Variability	CoV(Throughput) / CoV(Demand)	Ratio > 2.0 or bottom 30%
Workflow Stage Time Variability	CoV of stage duration for same-size items	CoV > 1.5 or bottom 30%
Team Member Throughput Variability	CoV of individual throughput	CoV > 1.0 or bottom 30%
Estimation Variability	CoV of estimates vs actuals	Bottom 30%
In-Progress Items Variability	Sprint-to-sprint WIP count variability	Bottom 30%
Cycle Time Variability	CoV of overall cycle time	Bottom 30%
Demand-Output Correlation	Correlation between demand and output	Low correlation (<0.3)

5.2 Category 2: Infrequent Tool Use Indicators (7 indicators)

These indicators detect patterns suggesting Jira is not the primary work tracking mechanism.

Indicator	Description	Flag When

Stale In-Progress Items	% of items in progress > N days without update	> 70% or top 30%
Bulk Change Ratio	% of changes in bulk operations	> 40% or top 30%
Average Daily Updates	Updates per item per day	< 0.1 or bottom 30%
Comment Frequency	Comments per completed item	Bottom 30%
Status Update Frequency	Status changes per item	Bottom 30%
Evening/Weekend Updates	% of updates outside business hours	Top 30% (suspicious pattern)
Update Clustering	Temporal clustering of updates	High clustering (retroactive)

5.3 Category 3: Work Intake Process Indicators (3 indicators)

These indicators detect work bypassing formal intake channels.

Indicator	Description	Flag When
Mid-Sprint/Mid-Cycle Additions	% of items added after sprint start	> 60% or top 30%
Siloed Work Items	% of items with single contributor	> 70% or top 30%
Cross-Team Spillover	Work from outside the team's domain	Top 30%

---

6. Risk Level Calculation

6.1 Individual Indicator Scoring

Each indicator produces a percentile score (0-100):

```
def score_indicator(team_value, comparison_distribution):
    percentile = percentile_rank(team_value, comparison_distribution)

    # Normalize so higher score = worse (more risk)
    if metric.higher_is_worse:
        return percentile
    else:
        return 100 - percentile
```

6.2 Category Aggregation

Each category score = average of its indicator percentiles:

```
def score_category(indicator_scores):
    return np.mean(list(indicator_scores.values()))
```

6.3 Overall Risk Score

Categories are weighted by importance:

```
def calculate_risk_score(category_scores):
    weights = {
        'unexplained_variability': 0.40,  # Primary signal (Dark Matter)
        'infrequent_tool_use': 0.35,      # Strong secondary (Stale Data)
        'work_intake_process': 0.25       # Supporting signal (Shadow Work)
    }

    weighted_sum = sum(
        category_scores[cat] * weight
        for cat, weight in weights.items()
    )

    return weighted_sum
```

6.4 Risk Level Classification

Composite Percentile	Risk Level	Interpretation
≤33rd percentile	**High**	Significant indicators suggest invisible work is likely
34-66th percentile	**Moderate**	Some concerning patterns warrant investigation
>66th percentile	**Low**	Indicators suggest most work is captured in Jira

---

7. Statistical Considerations and Limitations

7.1 Data Quality Requirements

- **Minimum history:** 3 months of data recommended; 6 months preferred
- **Minimum activity:** Teams with < 10 items/month may have insufficient signal
- **Consistent configuration:** Workflow changes mid-period can distort metrics

7.2 Correlation Between Indicators

Several indicators are likely correlated (e.g., low daily updates correlates with high stale items).

Implications:

- Don't over-count the same underlying issue
- Report category scores, not just indicator counts
- Focus on the overall risk level, not individual flags

7.3 Known Limitations

1. **Proxy measures:** All indicators are proxies; false positives and negatives will occur
2. **Gaming potential:** Teams aware of metrics may optimize for metrics rather than behavior
3. **Context incompleteness:** Stratification factors don't capture all legitimate variance sources
4. **Benchmark data dependency:** Quality depends on breadth of comparison data

7.4 Validation Approach

To validate the model:

1. **Face validity:** Do flagged teams agree they have invisible work issues?
2. **Predictive validity:** Do high-risk scores correlate with delivery problems?
3. **Intervention response:** Do scores improve after process changes?

---

8. Implementation Recommendations

8.1 Phased Rollout

Phase 1: Foundation

- Implement percentile comparison and absolute thresholds
- Use broad comparison groups initially
- Gather feedback on face validity

Phase 2: Refinement

- Refine comparison group stratification based on data volume
- Adjust thresholds based on feedback
- Add confidence indicators for small comparison groups

Phase 3: Advanced

- Implement residual variability analysis when data permits
- Build predictive models linking scores to outcomes
- Personalized recommendations based on specific flag patterns

8.2 User Communication

- **Explain the methodology:** Users should understand they're being compared to similar teams
- **Show comparison group:** Display what teams they're being compared against
- **Confidence indicators:** Flag when comparison group is small or match is imperfect
- **Actionable framing:** Focus on "here's what you can investigate" not "you're bad"

---

9. Conclusion

The proposed approach—combining context-dependent percentile scoring with absolute threshold guardrails—provides a statistically sound framework for **assessing invisible work risk** while remaining practical to implement and interpret.

Key principles:

1. **Context matters:** Compare like with like
2. **Residuals over absolutes:** Focus on unexplained variance
3. **Multiple signals:** No single indicator is definitive
4. **Guardrails prevent absurdity:** Some values are objectively concerning
5. **Transparency builds trust:** Show users how risk levels are calculated
6. **Three-tier classification:** High, Moderate, Low risk levels provide actionable guidance

This framework should be treated as a starting point. Empirical validation with real teams will refine thresholds, weights, and stratification criteria over time.

Appendix A: Glossary

Term	Definition
CoV (Coefficient of Variation)	Standard deviation divided by mean; normalized variability measure
Percentile rank	Percentage of comparison group at or below a given value
Residual	Difference between observed value and expected value from model
Stratification	Dividing population into subgroups for fairer comparison
WIP (Work in Progress)	Items currently being actively worked on

---

Appendix B: Data Requirements

Data Point	Source	Required/Optional
Item created date	Jira	Required
Item resolved date	Jira	Required
Status transitions	Jira	Required
Item assignee	Jira	Required
Item size/estimate	Jira	Recommended
Sprint membership	Jira	Required (Scrum teams)
Comments/updates	Jira	Required
Team member list	Jira/Config	Required
Team metadata (size, type)	Config	Required

Document Version: 4.0 Last Updated: December 2024 Changes in v4.0: Simplified to 3 risk levels (High/Moderate/Low); removed survey calibration system; focus on percentile-based comparison