Outcome-Dimension Contribution Matrix

Purpose

This document defines how each of the 14 dimensions contributes to each of the 7 outcomes. The mappings are based on **logical necessity analysis** - asking "what must be true about the Jira data for this outcome to be achievable?"

Methodology

Each dimension-outcome relationship is categorized as:

Category	Description	Weight Range
Required	Outcome is impossible or severely compromised without this dimension	0.20 - 0.30
Important	Significantly enables the outcome	0.10 - 0.20
Supporting	Helps but not essential	0.05 - 0.10
Not Applicable	No meaningful logical connection	0 (excluded)

Weights within each outcome sum to 1.0.

Outcome 1: Delivery Commitments

Question: "Can we use our Jira data to make reliable commitments about what will be delivered when?"

What this requires: The ability to predict delivery dates based on capacity, velocity, and scope.

Dimension	Category	Weight	Reasoning
estimationCoverage	Required	0.25	**You cannot commit to delivery dates without estimates.** Unestimated work makes capacity planning impossible. If 40% of work has no points, velocity-based forecasting is guesswork. This is definitionally required.
sizingConsistency	Required	0.20	**Inconsistent estimates make velocity unstable.** If "5 points" means 2 days for one person and 2 weeks for another, historical velocity cannot predict future delivery. Commitment reliability requires calibrated estimates.
workCaptured	Important	0.15	**Hidden work causes commitment overruns.** If significant effort happens outside Jira, commitments underestimate true remaining work. However, you *can* make commitments on visible work alone—they'll just be wrong.
informationHealth	Important	0.15	**Missing details cause mid-sprint discovery.** Tickets without acceptance criteria or context lead to scope creep during execution. Commitments assume understood scope; incomplete information undermines this.
sprintHygiene	Supporting	0.10	**Carryover patterns indicate planning accuracy.** High carryover suggests commitments routinely exceed capacity. Historical sprint health informs future commitment reliability, but doesn't directly enable/disable planning.
dataFreshness	Supporting	0.08	**Stale data affects velocity calculations.** If completion dates are delayed in Jira, velocity metrics are slightly off. Impact is real but secondary to estimation fundamentals.
backlogDiscipline	Supporting	0.07	**Unrefined backlog slows planning.** If top backlog items lack estimates or details, sprint planning becomes grooming. Affects planning efficiency, not whether commitments are possible.

Dimensions excluded (Not Applicable):

- workHierarchy: Epic linkage doesn't affect sprint-level delivery predictions
- teamCollaboration: Discussion patterns don't influence forecast accuracy
- blockerManagement: Affects execution, not planning ability
- collaborationFeatureUsage: Feature adoption doesn't impact commitment reliability
- automationOpportunities: Efficiency concern, not planning concern
- configurationEfficiency: Setup complexity doesn't affect forecasting
- issueTypeConsistency: Type usage doesn't impact delivery predictions

---

Outcome 2: Progress Tracking

Question: "Can we use our Jira data to track progress towards achieving a goal?"

What this requires: Real-time visibility into work state for burndowns, dashboards, and status reporting.

Dimension	Category	Weight	Reasoning
dataFreshness	Required	0.30	**Stale status makes dashboards lie.** If tickets show "In Progress" but work finished yesterday, burndowns are wrong. Progress tracking is fundamentally about current state—freshness is definitionally required.
workCaptured	Required	0.25	**Progress only shows what's in Jira.** If 30% of work is invisible, progress dashboards show 70% of reality. You cannot track what isn't recorded. This is definitionally required.
sprintHygiene	Important	0.18	**Last-day completion spikes indicate uncertain progress.** If 80% of work completes on day 10, progress was unknowable on days 1-9. Clean completion patterns enable meaningful mid-sprint tracking.
workHierarchy	Important	0.15	**Without linkage, rollup fails.** Epic and initiative progress requires stories linked to epics linked to initiatives. Missing hierarchy makes portfolio-level progress invisible.
configurationEfficiency	Supporting	0.07	**Too many statuses obscure progress.** Is "In Review" 70% done or 95% done? If "In Testing" differs from "QA" differs from "Validation," progress interpretation requires tribal knowledge.
informationHealth	Supporting	0.05	**Sparse descriptions make progress hard to interpret.** "What does 'done' mean for this ticket?" Without context, progress percentages lack meaning. Minor factor compared to freshness and capture.

Dimensions excluded (Not Applicable):

- estimationCoverage: Progress can be tracked by count, status, or % complete without estimates
- sizingConsistency: Estimate quality doesn't affect whether you can see current state
- teamCollaboration: Discussion doesn't affect progress visibility
- blockerManagement: Blockers affect execution speed, not progress visibility
- collaborationFeatureUsage: Collab features don't impact tracking
- automationOpportunities: Efficiency, not visibility
- backlogDiscipline: Backlog is future work, not current progress
- issueTypeConsistency: Type consistency helps aggregation but isn't required for tracking

---

Outcome 3: Productivity Measurement

Question: "Can we use our Jira data to measure our productivity?"

What this requires: Accurate throughput/velocity metrics that reflect actual team output.

Dimension	Category	Weight	Reasoning
workCaptured	Required	0.25	**Productivity metrics only count visible work.** If engineers spend 40% of time on work not in Jira, productivity appears 40% lower than reality. You cannot measure what isn't recorded. Definitionally required.
estimationCoverage	Required	0.22	**Velocity requires estimates.** Without story points, "productivity" becomes ticket counting—a 5-minute config change equals a 2-week feature. Points enable meaningful throughput measurement.
sizingConsistency	Required	0.20	**Inconsistent estimates make comparisons unfair.** If Team A sizes aggressively (small points) and Team B sizes conservatively, velocity comparisons are meaningless. Fair measurement requires calibration.
dataFreshness	Important	0.15	**Stale data delays throughput recognition.** If completions aren't recorded promptly, velocity metrics lag reality. Affects measurement accuracy but not whether measurement is possible.
sprintHygiene	Supporting	0.10	**Carryover complicates velocity calculation.** If the same ticket spans 3 sprints, which sprint gets the "credit"? Clean sprint boundaries enable cleaner velocity attribution.
teamCollaboration	Supporting	0.08	**Collaboration patterns affect attribution.** Single-contributor issues enable clear individual measurement; highly collaborative work complicates "who did what." Minor factor.

Dimensions excluded (Not Applicable):

- workHierarchy: Epic linkage doesn't affect throughput metrics
- informationHealth: Description quality doesn't impact velocity calculation
- blockerManagement: Blockers affect what gets done, not measurement of what did
- collaborationFeatureUsage: Feature usage doesn't impact productivity metrics
- automationOpportunities: Efficiency potential vs. current measurement
- configurationEfficiency: Setup issues don't affect throughput math
- issueTypeConsistency: Type usage doesn't fundamentally affect measurement
- backlogDiscipline: Backlog state doesn't affect completed work measurement

---

Outcome 4: Continuous Improvement

Question: "Can we use our Jira data to determine how we might improve our processes?"

What this requires: Historical patterns, trends, and data that reveal bottlenecks and improvement opportunities.

Dimension	Category	Weight	Reasoning
sprintHygiene	Required	0.25	**Sprint patterns reveal process health.** Carryover trends, velocity stability, and scope change frequency are primary inputs to retrospectives. Clean sprint data enables meaningful improvement analysis.
dataFreshness	Important	0.18	**Stale data masks bottlenecks.** If cycle time metrics are based on delayed status updates, you can't see where work actually gets stuck. Current state visibility is necessary for bottleneck identification.
estimationCoverage	Important	0.15	**Estimation accuracy analysis requires estimates.** "Are we getting better at sizing?" requires historical estimate vs. actual comparisons. Without estimates, this improvement dimension is invisible.
sizingConsistency	Important	0.12	**Calibration issues indicate team alignment needs.** If estimates vary wildly, the team isn't aligned on work understanding. Consistency metrics reveal a specific improvement opportunity.
automationOpportunities	Important	0.12	**Repetitive work reveals automation potential.** Identifying manual patterns that could be automated is itself an improvement insight. This dimension directly surfaces improvement opportunities.
teamCollaboration	Supporting	0.10	**In-Jira discussions create retrospective material.** Comments on blockers, delays, and issues provide qualitative data for process review. Silent tickets leave no trail for improvement analysis.
workCaptured	Supporting	0.08	**Hidden work patterns reveal process gaps.** If certain work types consistently stay outside Jira, that's an improvement target. But improvement analysis can proceed on visible work alone.

Dimensions excluded (Not Applicable):

- workHierarchy: Structure doesn't drive improvement insights
- informationHealth: Description quality doesn't reveal process patterns
- blockerManagement: Blockers are execution events, not process patterns
- collaborationFeatureUsage: Feature adoption is separate from process improvement
- configurationEfficiency: Setup optimization is separate from process improvement
- issueTypeConsistency: Type usage doesn't reveal improvement opportunities
- backlogDiscipline: Backlog health is a planning concern, not retrospective material

---

Outcome 5: Collaboration Effectiveness

Question: "Are we using Jira to collaborate effectively?"

What this requires: Evidence that Jira serves as a collaboration hub, not just a task dump.

Dimension	Category	Weight	Reasoning
teamCollaboration	Required	0.30	**This IS the outcome.** Comment density, multi-contributor issues, and discussion patterns directly measure whether collaboration happens in Jira. Definitionally required.
collaborationFeatureUsage	Required	0.25	**Feature adoption indicates collaboration intent.** @mentions, issue links, and watchers are collaboration mechanisms. Low usage means Jira isn't being used for coordination. Directly measures collaboration.
blockerManagement	Important	0.18	**Blocker flagging is collaborative behavior.** Actively flagging blockers and dependencies signals team awareness and coordination. Silent blockers indicate collaboration gaps.
configurationEfficiency	Supporting	0.12	**Bloated configuration discourages engagement.** If creating a ticket requires 15 fields, people avoid Jira. Lean setup promotes usage; heavy setup creates friction against collaboration.
automationOpportunities	Supporting	0.10	**Manual repetitive work displaces collaboration.** If people spend time on automatable tasks, they have less capacity for meaningful collaboration. Freeing time enables engagement.
informationHealth	Supporting	0.05	**Rich ticket content invites engagement.** Well-documented tickets prompt questions and discussion; empty tickets don't spark collaboration. Minor factor.

Dimensions excluded (Not Applicable):

- estimationCoverage: Estimates don't affect collaboration patterns
- sizingConsistency: Sizing calibration is separate from collaboration
- workCaptured: Work capture is about visibility, not collaboration
- dataFreshness: Freshness is about accuracy, not collaboration
- workHierarchy: Hierarchy is structure, not collaboration
- sprintHygiene: Sprint practices are methodology, not collaboration
- backlogDiscipline: Backlog health is planning, not collaboration
- issueTypeConsistency: Type usage doesn't affect collaboration

---

Outcome 6: Portfolio Planning

Question: "Is our Jira data reliable enough to be used in portfolio-level planning or decision-making?"

What this requires: Data that accurately rolls up from tasks to strategic initiatives for executive visibility.

Dimension	Category	Weight	Reasoning
workHierarchy	Required	0.28	**Without linkage, rollup is impossible.** Portfolio visibility requires stories → epics → initiatives chain. If stories aren't linked to epics, strategic progress cannot be computed. Definitionally required.
estimationCoverage	Important	0.18	**Portfolio capacity planning requires estimates.** Resource allocation across initiatives needs effort estimates. Missing points at story level cascade to missing data at portfolio level.
workCaptured	Important	0.15	**Portfolio views only show tracked work.** If 30% of effort is invisible, portfolio dashboards undercount by 30%. Strategic decisions based on partial data are misinformed.
informationHealth	Important	0.12	**Stakeholders need context.** Epic and initiative descriptions must be clear for executive consumption. Sparse details undermine confidence in portfolio reporting.
issueTypeConsistency	Supporting	0.10	**Cross-team aggregation requires consistency.** If Team A's "Epic" is Team B's "Feature," portfolio rollups are apples-to-oranges. Consistent categorization enables accurate aggregation.
dataFreshness	Supporting	0.10	**Strategic decisions require current state.** If portfolio dashboards show last week's status, resource reallocation decisions are based on stale information.
sizingConsistency	Supporting	0.07	**Cross-team comparisons need calibration.** If Team A sizes small and Team B sizes large, initiative comparisons are skewed. Less critical than at team level, but still relevant.

Dimensions excluded (Not Applicable):

- teamCollaboration: Team-level concern, not portfolio visibility
- blockerManagement: Execution-level, not strategic visibility
- collaborationFeatureUsage: Feature adoption doesn't affect portfolio rollup
- automationOpportunities: Efficiency, not portfolio visibility
- configurationEfficiency: Setup issues don't affect strategic reporting
- sprintHygiene: Sprint-level concern, not portfolio planning
- backlogDiscipline: Backlog health is team-level, not portfolio

---

Outcome 7: Risk Detection

Question: "Can we use our Jira data to identify risks and blockers early?"

What this requires: Early warning signals that surface problems before they become emergencies.

Dimension	Category	Weight	Reasoning
blockerManagement	Required	0.30	**This IS the outcome.** Blocker flagging, dependency tracking, and impediment management directly measure risk detection capability. If blockers aren't flagged, you learn about delays at deadline. Definitionally required.
dataFreshness	Required	0.25	**Stale data hides emerging risks.** If a ticket has been stuck for 5 days but status hasn't been updated, the risk is invisible. You cannot detect risks in data you don't have. Definitionally required.
teamCollaboration	Important	0.18	**Risks discussed in Jira become visible.** Comments flagging concerns, delays, or dependencies create an early warning trail. Silent tickets hide problems until escalation.
collaborationFeatureUsage	Important	0.15	**Links surface dependencies.** Issue links showing "blocks/blocked by" relationships make dependency risks visible. @mentions route attention to problems. Underused features = missed warnings.
workHierarchy	Supporting	0.07	**Hierarchy shows impact scope.** A blocked story is one thing; knowing it blocks an epic which blocks a release is another. Linkage enables understanding of risk magnitude.
workCaptured	Supporting	0.05	**Hidden work can harbor hidden risks.** If significant work is outside Jira, risks in that work are invisible. Minor factor compared to detection mechanisms.

Dimensions excluded (Not Applicable):

- estimationCoverage: Estimates don't reveal risks
- sizingConsistency: Sizing quality doesn't affect risk detection
- informationHealth: Description quality is separate from risk flagging
- issueTypeConsistency: Type usage doesn't affect early warning
- sprintHygiene: Sprint health is performance, not risk detection
- backlogDiscipline: Backlog is future work, not current risks
- automationOpportunities: Efficiency, not risk detection
- configurationEfficiency: Setup issues don't affect warning systems

---

Summary Matrix

Dimension	Commitments	Progress	Productivity	Improvement	Collaboration	Portfolio	Risk Detection
estimationCoverage	0.25	-	0.22	0.15	-	0.18	-
sizingConsistency	0.20	-	0.20	0.12	-	0.07	-
workCaptured	0.15	0.25	0.25	0.08	-	0.15	0.05
informationHealth	0.15	0.05	-	-	0.05	0.12	-
dataFreshness	0.08	0.30	0.15	0.18	-	0.10	0.25
workHierarchy	-	0.15	-	-	-	0.28	0.07
issueTypeConsistency	-	-	-	-	-	0.10	-
sprintHygiene	0.10	0.18	0.10	0.25	-	-	-
backlogDiscipline	0.07	-	-	-	-	-	-
teamCollaboration	-	-	0.08	0.10	0.30	-	0.18
blockerManagement	-	-	-	-	0.18	-	0.30
collaborationFeatureUsage	-	-	-	-	0.25	-	0.15
automationOpportunities	-	-	-	0.12	0.10	-	-
configurationEfficiency	-	0.07	-	-	0.12	-	-

Dimension Coverage Analysis

How many outcomes does each dimension contribute to?

Dimension	# Outcomes	Outcomes
dataFreshness	6	Commitments, Progress, Productivity, Improvement, Portfolio, Risk
workCaptured	6	Commitments, Progress, Productivity, Improvement, Portfolio, Risk
estimationCoverage	4	Commitments, Productivity, Improvement, Portfolio
sprintHygiene	4	Commitments, Progress, Productivity, Improvement
sizingConsistency	4	Commitments, Productivity, Improvement, Portfolio
teamCollaboration	4	Productivity, Improvement, Collaboration, Risk
workHierarchy	3	Progress, Portfolio, Risk
informationHealth	3	Commitments, Progress, Portfolio
blockerManagement	2	Collaboration, Risk
collaborationFeatureUsage	2	Collaboration, Risk
automationOpportunities	2	Improvement, Collaboration
configurationEfficiency	2	Progress, Collaboration
issueTypeConsistency	1	Portfolio
backlogDiscipline	1	Commitments

Observation: `dataFreshness` and `workCaptured` are foundational—they contribute to almost every outcome. `issueTypeConsistency` and `backlogDiscipline` are specialized—each supports only one outcome strongly.

How to Use This Matrix

For Admin Configuration

When an admin sets outcome priorities:

```
Admin sets:
  Commitments:    High (weight = 3)
  Progress:       Medium (weight = 2)
  Productivity:   Low (weight = 1)
  ... etc

System computes dimension weights:
  For each dimension d:
    raw_weight[d] = Σ (outcome_priority[o] × contribution[d][o])

  Then normalize so all dimension weights sum to 1.0
```

Example Calculation

If admin prioritizes: Commitments (3), Progress (2), others (1)

For `estimationCoverage`:

```
= (3 × 0.25) + (2 × 0) + (1 × 0.22) + (1 × 0.15) + (1 × 0) + (1 × 0.18) + (1 × 0)
= 0.75 + 0 + 0.22 + 0.15 + 0 + 0.18 + 0
= 1.30
```

For `dataFreshness`:

```
= (3 × 0.08) + (2 × 0.30) + (1 × 0.15) + (1 × 0.18) + (1 × 0) + (1 × 0.10) + (1 × 0.25)
= 0.24 + 0.60 + 0.15 + 0.18 + 0 + 0.10 + 0.25
= 1.52
```

After computing all dimensions, normalize to sum to 1.0.

Validation Checklist

For each outcome, verify:

- ☐ Required dimensions: Would the outcome be impossible without this?
- ☐ Important dimensions: Does this significantly enable the outcome?
- ☐ Supporting dimensions: Does this help but isn't essential?
- ☐ Excluded dimensions: Is there truly no logical connection?
- ☐ Weights sum to 1.0 within each outcome

Change Log

Date	Author	Change
2026-01-27	Claude (draft)	Initial matrix based on logical necessity analysis
		Awaiting human review and validation