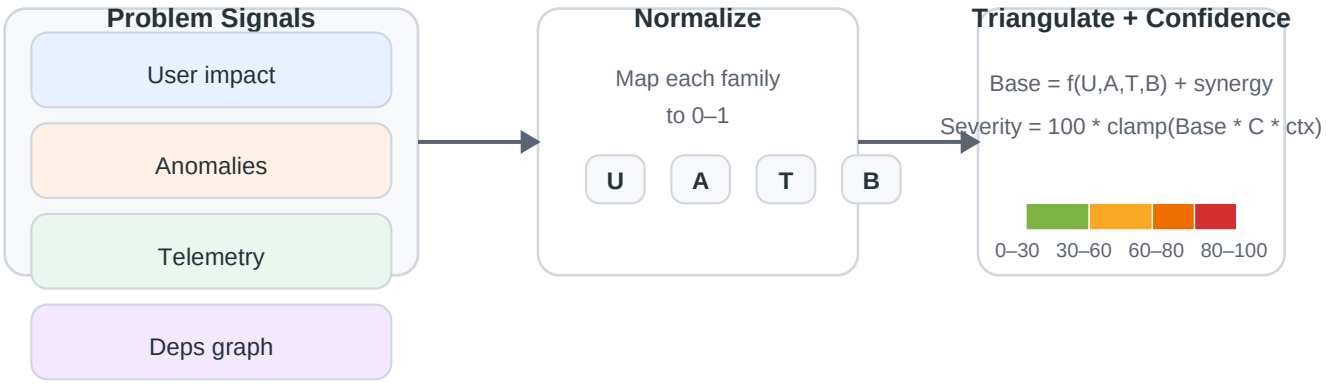
Alert Risk Assessment Classification

Triangulated Risk Score (0-100) that unifies user impact, anomalies, telemetry, and dependency blast radius



• **Objective:** Produce a single severity score (0-100) for each alert by fusing heterogeneous signals into interpretable evidence dimensions.

• **Design principle:** A real incident usually lights up multiple evidence families. A noisy single sensor should not page on its own.

• **Core model:** Severity = 100 x clamp((Impact + Evidence + BlastRadius) x Confidence x Context).

Outputs supported: numeric score, plus your own thresholds for RED vs non-RED routing (immediate vs time-boxed action).

1. Problem and goals

AIOps360 receives multiple problem-signal inputs with different noise profiles and different meanings. The goal is to unify them into a severity score that (a) tracks real user harm, (b) reflects technical plausibility, (c) accounts for potential blast radius via dependencies, and (d) reduces false positives via confidence gating.

Primary outcome: a stable 0-100 score that can drive alert routing and prioritization.

Secondary outcomes: explainability ("why this is high"), tunability, and a path to ML without labeled incidents.

2. Signal families consumed

Family	Examples in your system	What it tells you
User impact (UI)	Synthetics/RUM, user friction, essential services, customer journeys	Direct evidence of user harm / workflow degradation
Anomaly detections (AD)	App log anomalies, Dynatrace (DtM) anomalies	Technical symptoms; often early but can be noisy
SRE telemetry (SRE)	MCPs for logs/metrics/traces, SLO/error budgets, env context (M12), SNOW changes/incidents	Ground truth of system health + operational context
Dependency graph (DG)	One-hop fan-out (dependents) and fan-in (upstreams)	Potential blast radius and propagation risk

Current signal-to-alert flow (as provided)

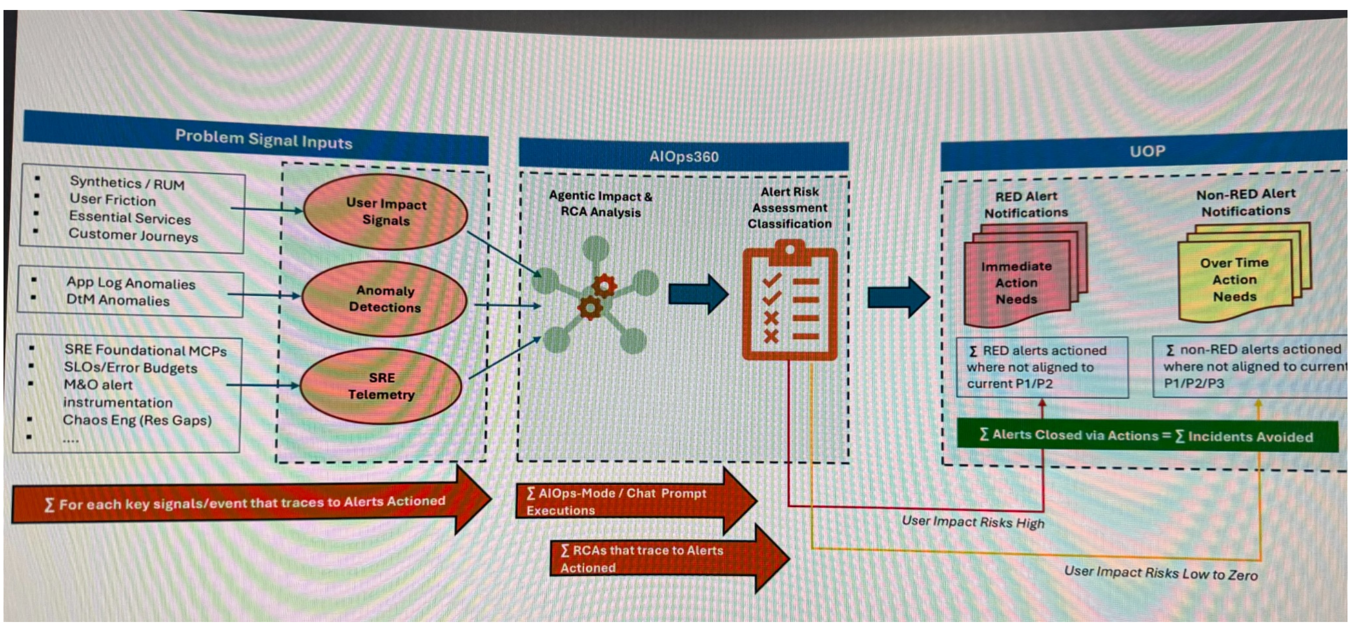


Figure: Problem-signal inputs -> AIOps360 agentic impact/RCA -> alert risk classification -> UOP notifications.

3. Normalize all raw signals into 0-1 evidence scores

The scoring framework works best when every raw signal is first mapped into a comparable 0-1 severity scale. This avoids apples-to-oranges weighting (e.g., raw log counts vs. journey failure rate).

Recommended mapping patterns

• **Sigmoid mapping:** converts a metric that has a meaningful threshold into a smooth 0-1 severity.

• **Z-score mapping:** uses deviation from baseline when absolute thresholds vary by service.

• **Persistence weighting:** multiplies by a factor that increases with continuous duration (reduces transient noise).

• **Noisy-OR fusion:** treats multiple sensors as corroborating evidence: one strong sensor can be enough; many sensors compound.

Example transforms (illustrative)

Use these as templates; calibrate thresholds per org/service where needed:

```
sigmoid(x) = 1 / (1 + exp(-x)) u_syn = sigmoid(a*(error_rate - e0)) u_fric =
sigmoid(a*(pct_users_impacted - p0)) * corr(frontend_error, backend_service) A = 1 -
(1 - a_log) * (1 - a_dtm) U = 1 - Π(1 - u_i) (Noisy-OR across impact sensors)
```

4. Four evidence dimensions: U, A, T, B

Instead of mixing everything into one opaque score, maintain four interpretable evidence dimensions and combine them later.

U - User Impact

• Noisy-OR across synthetics/RUM, user friction, and customer-journey failure signals.

• Apply an essential-service multiplier as a criticality factor rather than a raw signal.

• Goal: measure user harm in a way that quickly rises when users feel it.

A - Anomaly Evidence

• Fuse app-log anomalies and Dynatrace anomalies, then weight by persistence.

• Goal: detect early technical symptoms while limiting one-off spikes.

T - Telemetry and Context Evidence

• Fuse SLO burn-rate, metrics, traces, and operational context (recent change, active incident, environment signals).

• Goal: represent 'system is truly unhealthy' evidence that SRE teams trust.

B - Blast Radius

• Compute a dependency-based risk proxy from one-hop fan-out and (optionally) critical dependent weight.

• Goal: boost severity when a service can affect many others or key workflows.

5. Confidence gating: punish single-sensor hallucinations

Confidence is a multiplicative factor (0-1) that rewards corroboration across categories and persistence over time. This is the main lever that reduces false positives without needing historical labels.

```
C_sources = (# categories with meaningful evidence) / 4 C_persist =
sigmoid(p*(duration_minutes - 5)) C_fresh = 1.0 if signals are fresh else 0.6 C =
clamp(0.35*C_sources + 0.45*C_persist + 0.20*C_fresh)
```

Interpretation: a loud log anomaly alone stays modest unless it persists or is corroborated by user impact or telemetry.

6. Final severity score: triangulation + synergy

Compute a base score from U/A/T/B, add a synergy bonus when user impact aligns with technical evidence, then apply confidence and context.

```
Base = 0.55*U + 0.30*(0.6*T + 0.4*A) + 0.15*B Base = clamp(Base + 0.20*(U * (T + A)/2))
# synergy bonus Severity = 100 * clamp(Base * C * ContextMultiplier)
```

ContextMultiplier is a small factor to incorporate known operational conditions (e.g., active change/deploy slightly increases risk; approved maintenance slightly decreases risk). Keep it narrow (e.g., 0.85-1.15) to avoid dominating evidence.

Suggested routing bands (optional)

• 0-30: watch / informational

• 30-60: investigate soon (likely non-RED)

• 60-80: high risk (prioritize)

• 80-100: RED (immediate action)

7. ML extensions without labeled incidents

You can improve accuracy over time even without ground-truth severity labels by using weak supervision and unsupervised learning.

Option A - Weak-label learning (best ROI): treat "alert actioned", "ticket created", or "incident opened" as proxy labels. Train a lightweight model (logistic regression / gradient boosting) to predict action probability from U/A/T/B and supporting features.

Option B - Unsupervised novelty: compute a per-service feature vector and score "weirdness" via Isolation Forest or autoencoder reconstruction error. Fuse it as a second opinion (e.g., 70% rule score +

30% novelty score).

Option C - Cross-signal agreement: self-supervised model predicts one family (e.g., synthetics) from others (metrics/traces/logs). When prediction error spikes, elevate risk because normal relationships no longer explain the system state.

8. Implementation notes and guardrails

• **Keep U/A/T/B explainable:** store the top contributing signals and their normalized values alongside the score.

• **Calibrate per service:** baseline-based transforms (z-scores) reduce the need for global thresholds.

• **Measure outcomes:** even without incident labels, log whether the alert was actioned, time-to-ack, and downstream ticket/incident creation.

• **Control noise:** persistence and source-coverage should be mandatory before RED escalation (unless U is extreme).

• **Audit drift:** changes in synthetics configuration, journey definitions, or telemetry pipelines can shift score distributions.

9. Worked example (illustrative numbers)

Suppose an alert shows moderate user impact plus strong telemetry corroboration and a high fan-out service. One possible set of normalized evidence values is U=0.70, A=0.40, T=0.80, B=0.60. If confidence C=0.85 and context multiplier is 1.10 (active change window), the severity lands in the high-risk range.

```
Base = 0.55*0.70 + 0.30*(0.6*0.80 + 0.4*0.40) + 0.15*0.60 = 0.66 Synergy =
0.20*(0.70*((0.80+0.40)/2)) = 0.084 -> Base' ~ 0.744 Severity = 100 * (0.744 * 0.85 *
1.10) ~ 69.6
```

Note: this example demonstrates the intended behavior: user impact + telemetry alignment lifts the score; confidence and context can nudge the final routing without overriding core evidence.

Appendix: Minimal data contract (suggested fields)

```
alert_id, service_id, timestamp_window U components: synthetics_fail_rate,
journey_fail_rate, friction_pct_users, essentiality_flag A components: log_anomaly_score,
dtm_anomaly_score, anomaly_duration T components: slo_burn_rate, metric_spike_scores,
trace_latency_delta, change_flag, incident_flag B components: fan_out, fan_in,
critical_dependent_weight Derived: U, A, T, B, C, context_multiplier, final_severity,
explanation_top_factors[]
```