ENKIDU

MIOVSKI GROUP

undefined

Glossary

Key terms used throughout this guide.

Qualitative parameter A judgment-based value assigned to a pain point — criticality, priority, urgency, scope, sentiment, evidence density — that this guide renders deterministic and defensible.

Pain point A discrete problem, challenge, gap, or unmet need explicitly expressed by an interviewee and supported by verbatim citation.

Rubric anchor An explicit descriptor for a specific level of a specific parameter, against which the agent must match transcript evidence.

Cohen's Kappa (kappa) A statistic measuring agreement between two raters (here, two agent runs, or an agent versus a human coder) beyond chance.

Fleiss' Kappa (kappa_F) A generalization of Cohen's Kappa to more than two raters — used to validate multi-pass ensemble scoring.

MCDA Multi-Criteria Decision Analysis — a weighted additive model that combines several independently-scored parameters into one composite priority score.

Bayesian updating Progressive refinement of a probability distribution over possible scores as each new interview is processed.

Evidence chain The structured record linking every assigned score back to the specific transcript citations that justify it.

Module (M1-M5) One stage of the implementation pipeline — ingestion, extraction, scoring, MCDA computation, and audit-package generation.

Prompt hash A SHA-256 hash of the exact system prompt used, stored so a scoring run can be reproduced precisely.

Abstract

Analyzing interview transcripts to extract and prioritize pain points requires assigning qualitative values — criticality, priority, urgency, impact — that are, by nature, subjective. For a governed, auditable process, three problems must be solved at once: subjectivity (two analysts may score the same transcript differently), repeatability (will the same input produce the same result tomorrow?), and traceability (can every score be traced to specific evidence?). This guide answers all three.

It presents four methods of increasing mathematical rigour — rubric-anchored deterministic scoring, multi-pass ensemble scoring with statistical convergence, weighted multi-criteria decision analysis, and Bayesian hierarchical scoring — each with its worked example and its governance metric. It then provides a complete implementation guide for the analysis agent: a five-module pipeline from transcript ingestion through audit-package generation, with system prompts, code, and a governance-compliance checklist. The recommended approach is a hybrid: rubric-anchored scoring feeding an MCDA composite, validated by ensemble reliability, with the same evidence-and-provenance discipline MIOVSKI GROUP applies across its analytical work.

> The mathematical tools transform what is typically a subjective exercise into a disciplined, auditable analytical process — every score traceable, reproducible, and defensible.

1. The Statistical Dilemma

When analyzing interview transcripts to extract pain points — like those from the CWS branch engagement interviews — analysts must assign qualitative values such as criticality, priority, urgency, and impact. The core dilemma is threefold:

Subjectivity
Two human analysts reading the same transcript may assign different criticality scores. How do we make an AI agent's scoring deterministic and defensible?

Repeatability
If the same transcript is processed again tomorrow, will the agent produce the same results? What mathematical guarantees can we provide?

Traceability
For data-governance and ethics audits, every score must be traceable to specific evidence in the source transcripts. How do we build an audit chain?

This guide presents four distinct options for solving this problem, each with increasing mathematical rigour, along with complete instructions for implementing the analysis agent using the Claude API.

2. Framework of Qualitative Parameters

Before selecting a scoring method, the parameters themselves must be defined. Based on the ECCC pain-point structure and the CWS interview data, the following parameter taxonomy is recommended:

Criticality — scale 1-5 (Low to Critical)
Degree to which the pain point threatens mandate delivery or legal compliance. Governance use: risk register, audit evidence.

Priority — P1-P4 (Immediate to Deferred)
Recommended order of remediation based on impact vs. effort. Governance use: roadmap sequencing.

Frequency — count (n) + percentage
How often the pain point was independently cited across interviews. Governance use: statistical weight.

Scope — Narrow / Moderate / Wide / Enterprise
Number of branches, directorates, or roles affected. Governance use: investment justification.

Urgency — scale 1-5 (Low to Immediate)
Time sensitivity — is there a deadline, legal trigger, or degrading condition? Governance use: sprint planning.

Sentiment Intensity — -1.0 to +1.0 (normalized)
Emotional weight of language used by interviewees. Governance use: change-management risk.

Evidence Density — Sparse / Moderate / Rich
Volume and specificity of supporting evidence per pain point. Governance use: confidence scoring.

2.1 Why these parameters matter for governance

Each parameter maps to a distinct governance requirement. Criticality satisfies risk-based audit requirements under TBS policy frameworks. Priority enables defensible resource-allocation decisions. Frequency provides statistical grounding — a pain point cited by 8 of 10 interviewees carries more weight than one cited by 1 of 10. Scope determines whether a solution needs enterprise-level investment or can be addressed locally. Sentiment intensity, while seemingly soft, is a leading indicator of adoption resistance and change-management risk.

3. Option A — Rubric-Anchored Deterministic Scoring

Concept. This approach pre-defines a detailed scoring rubric with explicit anchor descriptions for every level of every parameter. The AI agent is constrained to select from these anchors and must cite the specific transcript evidence that matches each anchor. It is the simplest approach to implement and the easiest to audit.

3.1 The criticality rubric

1 — Low
Inconvenience only. No impact on mandate delivery, compliance, or service levels. Workarounds exist and are sustainable. Triggers: "nice to have", "minor annoyance", "workaround is fine".

2 — Moderate
Measurable inefficiency. Staff time is wasted, but deadlines are still met. No compliance risk. Triggers: "takes extra time", "manual process", "could be better".

3 — High
Recurring disruption to operations. Deadlines are at risk. Staff morale is affected. Data-quality concerns exist. Triggers: "frequent setbacks", "years of delays", "frustrating", cost figures cited.

4 — Severe
Direct threat to mandate delivery. Compliance deadlines have been missed or are imminent. Data integrity compromised. Triggers: "cannot comply", "legal risk", "outdated system failing".

5 — Critical
Systemic failure. Public safety, legal obligations, or ministerial commitments are at immediate risk. Triggers: "system down", "cannot deliver", "ministerial liability", "public harm".

3.2 Mathematical basis — Cohen's Kappa for repeatability

To validate that the rubric produces repeatable results, inter-rater reliability is measured using Cohen's Kappa (kappa). Here, two separate runs of the AI agent (or an AI run versus a human coder) are treated as the two raters:

```
kappa = (P_o - P_e) / (1 - P_e)
```

where P_o is the observed proportion of agreement (how often both raters assigned the same score) and P_e is the expected proportion of agreement by chance.

Worked example from CWS data. Suppose 20 pain points are extracted from the CWS interviews and scored twice by the agent (Run A and Run B). The confusion matrix:

```
              RunB:2  RunB:3  RunB:4   Total
RunA: 2          5       1       0       6
RunA: 3          0       8       1       9
RunA: 4          0       1       4       5
Total            5      10       5      20
```

```
P_o = (5 + 8 + 4) / 20 = 0.85
```

```
P_e = (6x5 + 9x10 + 5x5) / 20² = (30 + 90 + 25) / 400 = 0.3625
```

```
kappa = (0.85 - 0.3625) / (1 - 0.3625) = 0.4875 / 0.6375 = 0.764
```

A kappa of 0.764 indicates "substantial agreement" on the Landis-Koch scale. For governance purposes, a target kappa >= 0.80 ("almost perfect") is recommended. If the agent falls below this threshold, the rubric anchors need refinement.

< 0.00 Poor (less than chance)

0.00 - 0.20 Slight agreement

0.21 - 0.40 Fair agreement

0.41 - 0.60 Moderate agreement

0.61 - 0.80 Substantial agreement

0.81 - 1.00 Almost perfect agreement

3.3 Traceability — the evidence chain

For every score the agent assigns, it must produce a structured evidence record:

pain_point_id Unique identifier (e.g., PP-CWS-017).

parameter Which parameter is being scored (e.g., Criticality).

score The numerical value assigned (e.g., 4).

rubric_anchor_matched The exact anchor text from the rubric that was matched.

source_citations[] Array of {interview_id, speaker, paragraph_number, verbatim_quote}.

confidence Agent's self-assessed confidence (0.0-1.0) based on evidence density.

reasoning Free-text explanation of why this anchor was selected over adjacent ones.

timestamp / model_version / prompt_hash ISO 8601 timestamp; the Claude model version used; SHA-256 hash of the system prompt, enabling exact reproduction.

4. Option B — Multi-Pass Ensemble Scoring with Statistical Conver

Concept. Instead of relying on a single agent pass, this approach runs the same transcript through the agent N times (typically N = 5 to 7) with controlled temperature variation. The final score is derived from the statistical distribution of results, not any single run.

4.1 Central tendency and dispersion

For each pain point p and parameter k, across N runs, a vector of scores is collected:

```
S_pk = {s_1, s_2, ..., s_N}
```

The final score is the median (robust to outliers), and the confidence measure is derived from the inverse of the coefficient of variation:

```
Score_pk = median(S_pk)   |   CV_pk = sigma(S_pk) / mean(S_pk)   |   Confidence_pk
= 1 - CV_pk
```

Worked example. Consider the pain point "Outdated e-Permitting tool (ColdFusion)." Across 5 runs, the agent assigns criticality scores S = {4, 4, 3, 4, 4}:

```
Median = 4, Mean = 3.8, Std Dev (sigma) = 0.447
```

```
CV = 0.447 / 3.8 = 0.118   |   Confidence = 1 - 0.118 = 0.882 (88.2%)
```

The final criticality is 4 (Severe) with 88.2% confidence — and that confidence score becomes part of the audit trail.

Consensus threshold. Define a consensus threshold: if the interquartile range (IQR) of scores across runs exceeds 1.0, the pain point is flagged for human review — the governance safety net:

```
If IQR(S_pk) > 1.0  ->  FLAG for human adjudication
```

4.2 Fleiss' Kappa for multi-rater agreement

When N > 2 runs are used, Fleiss' Kappa generalizes Cohen's Kappa:

```
kappa_F = (Pbar - Pbar_e) / (1 - Pbar_e)
```

where Pbar is observed agreement, Pbar = (1 / (N n (n-1))) sum_i sum_j n_ij (n_ij - 1), and Pbar_e = sum_j (p_j)² is chance agreement, with p_j the proportion of all ratings in category j. For the ensemble method to satisfy governance requirements, target kappa_F >= 0.75 across all parameter types.

5. Option C — Weighted Multi-Criteria Decision Analysis (MCDA)

Concept. This approach computes a single composite Priority Score from multiple independently-scored parameters using a weighted additive model. The weights are established by stakeholders (e.g., CWS directors) before the analysis begins, ensuring the prioritization reflects organizational values rather than algorithmic bias.

5.1 The weighted additive model

```
Priority(p) = sum_{k=1..K} w_k x normalized_score_k(p)
```

where K is the number of parameters, w_k are the stakeholder-assigned weights (summing to 1.0), and scores are min-max normalized to [0, 1]:

```
normalized_score_k(p) = (raw_score_k(p) - min_k) / (max_k - min_k)
```

Recommended weight configuration:

Criticality — 0.30
Mandate and compliance risk dominates. Source: DG-level validation.

Frequency — 0.20
Statistical recurrence indicates systemic issue. Source: calculated from data.

Scope — 0.20
Enterprise-wide issues get priority. Source: org-chart mapping.

Urgency — 0.15
Time-sensitive items need acceleration. Source: interview context.

Sentiment Intensity — 0.10
Change-management risk indicator. Source: NLP extraction.

Evidence Density — 0.05
Confidence modifier. Source: citation count.

5.2 Worked example — full MCDA calculation

Pain point: "Lack of centralized data repository" (cited by several interviewees).

```
Parameter     Raw    Min  Max   Norm    Weighted
Criticality    4      1    5   0.75   0.75x0.30 = 0.225
Frequency    7/10     0    1   0.70   0.70x0.20 = 0.140
Scope        4(Ent)   1    4   1.00   1.00x0.20 = 0.200
Urgency        3      1    5   0.50   0.50x0.15 = 0.075
Sentiment    -0.6     -1   +1   0.20   0.20x0.10 = 0.020
Evidence     Rich(3)  1    3   1.00   1.00x0.05 = 0.050
```

```
Priority Score = 0.225 + 0.140 + 0.200 + 0.075 + 0.020 + 0.050 = 0.710
```

5.3 Priority classification thresholds

0.75 - 1.00 -> P1 (Immediate)
Address in current planning cycle; escalate to DG.

0.50 - 0.74 -> P2 (High)
Include in next roadmap iteration.

0.25 - 0.49 -> P3 (Medium)
Schedule for future consideration; monitor.

0.00 - 0.24 -> P4 (Low)
Document and revisit annually.

The "Lack of centralized data repository" scores 0.710, placing it at P2 (High). This is mathematically defensible, traceable to specific interview evidence, and the weights were pre-approved by stakeholders — satisfying all three governance requirements.

5.4 Sensitivity analysis

To demonstrate robustness, perform Monte Carlo sensitivity analysis by varying weights +/-10% and recalculating priorities. If the priority classification changes for a given pain point, it is flagged as weight-sensitive and requires stakeholder discussion:

```
For each simulation i = 1..1000: w_k' = w_k + uniform(-0.1 w_k, +0.1 w_k), then
re-normalize; Priority_i(p) = sum w_k' x normalized_score_k(p). Stability =
P(priority class unchanged across 1000 simulations).
```

6. Option D — Bayesian Hierarchical Scoring with Progressive Upd

Concept. This is the most mathematically rigorous option. Instead of assigning a single point score, the agent maintains a probability distribution over possible scores for each pain point. As more interviews are processed, the distribution is updated using Bayes' theorem, progressively narrowing uncertainty.

6.1 Bayes' theorem applied to pain-point scoring

```
P(Score = c | Evidence) = P(Evidence | Score = c) x P(Score = c) / P(Evidence)
```

where P(Score = c) is the prior probability that the pain point has criticality c (initialized as uniform, 1/5 for each level 1-5); P(Evidence | Score = c) is the likelihood of observing this evidence given criticality c (derived from the rubric anchors); and P(Evidence) is the marginal likelihood (normalizing constant).

6.2 Sequential updating across interviews

After processing interview 1 for pain point p, we have a posterior distribution. This becomes the prior for interview 2:

```
After Interview 1: P1(c|E1) proportional to P(E1|c) x P0(c)
```

```
After Interview 2: P2(c|E2) proportional to P(E2|c) x P1(c|E1)
```

```
After Interview n: Pn(c|En) proportional to P(En|c) x P(n-1)(c|E(n-1))
```

6.3 Worked example — Bayesian updating for "e-Permitting Outdated Tool"

Prior (uniform): P0(c) = [0.20, 0.20, 0.20, 0.20, 0.20] for scores 1-5. Interview 1 mentions "6 years of frequent setbacks," "built in ColdFusion," and "not looked at as transformative." The likelihood function, based on rubric-anchor matching:

```
P(E1|c=1)=0.01, P(E1|c=2)=0.05, P(E1|c=3)=0.20, P(E1|c=4)=0.55, P(E1|c=5)=0.19
```

```
Unnormalized posterior: [0.002, 0.010, 0.040, 0.110, 0.038]  ->  Normalized
P1(c|E1): [0.010, 0.050, 0.200, 0.550, 0.190]
```

After Interview 1, the MAP (Maximum A Posteriori) estimate is Score 4 (Severe) with P = 0.55.

Interview 2 confirms the issue and adds "critical data is often siloed, outdated, or handled through inefficient, manual processes," further concentrating the posterior:

```
Updated P2(c|E2): [0.002, 0.015, 0.140, 0.680, 0.163]
```

The MAP estimate remains 4 but confidence has increased from 55% to 68%. Each interview is an observation that refines the distribution.

6.4 Governance advantages

The Bayesian approach has unique governance properties. Every intermediate posterior is stored, creating a full audit trail showing how the score evolved with each interview. The credible interval (analogous to a confidence interval) quantifies uncertainty. If the 90% credible interval spans more than 2 score levels, the pain point is flagged as requiring additional evidence or human adjudication.

7. Comparison of Options

Complexity
A Rubric: Low. B Ensemble: Medium. C MCDA: Medium-High. D Bayesian: High.

Repeatability
A: High (deterministic). B: Very High (statistical). C: High (formula-based). D: Very High (distributional).

Traceability
A: direct citation. B: median + distribution. C: full decomposition. D: full posterior chain.

Audit readiness
A: Excellent. B: Excellent. C: Excellent. D: Best-in-class.

Mathematical rigour
A: Moderate. B: High. C: High. D: Highest.

Implementation effort
A: 1-2 days. B: 3-5 days. C: 1 week. D: 2 weeks.

Best for
A: quick pilots. B: production systems. C: multi-stakeholder buy-in. D: evolving evidence bases.

Governance metric
A: Cohen's kappa >= 0.80. B: Fleiss' kappa >= 0.75. C: sensitivity stability >= 90%. D: 90% CI width <= 2 levels.

7.1 Recommended approach

For ECCC's immediate needs, a hybrid of Options A and C is recommended: use rubric-anchored scoring (Option A) to independently score each parameter, then feed those scores into the MCDA framework (Option C) to produce composite priority scores. This provides the best balance of rigour, auditability, and practical implementation time. Option B's ensemble method can be added later as a

validation layer, and Option D's Bayesian approach is ideal if the interview corpus will grow over time.

8. Building the Agent — Implementation Guide

Architecture overview. The agent is composed of five modules that form a pipeline. Each module is a distinct Claude API call with a specialized system prompt, structured output schema, and audit logging.

M1 — Transcript Ingestion
Parse raw interview transcripts into structured segments. Output JSON: {speaker, section, paragraphs[]}.

M2 — Pain Point Extraction
Identify discrete pain points with citations. Output JSON: {pain_point_id, description, citations[]}.

M3 — Parameter Scoring
Apply rubric to score each parameter for each pain point. Output JSON: {scores{}, evidence_chains{}, confidence{}}.

M4 — MCDA Computation
Calculate composite priority using the weighted model. Output JSON: {priority_score, priority_class, decomposition{}}.

M5 — Audit Package Generation
Compile the full audit trail for governance review. Output JSON + XLSX: complete evidence package.

8.1 Module M1 — Transcript Ingestion

The system prompt enforces structured output and prevents hallucination:

```
SYSTEM PROMPT - MODULE M1: TRANSCRIPT INGESTION
You are a transcript structuring agent for ECCC's Canadian
Wildlife Service digital transformation initiative.

TASK: Parse the provided interview transcript into structured JSON.

RULES:
1. Every output field must trace to a specific paragraph in the source.
2. Do NOT infer, summarize, or add content not present in the source.
3. Preserve the speaker's exact words in the 'verbatim' field.
4. Assign a sequential paragraph ID to every substantive paragraph.
5. Classify each section by its topic category from this controlled
   vocabulary: [Core Functions, Data & Information Management,
   Service Experience, Tools & Systems, Skills & Capacity,
   Processes & Automation, Collaboration & Governance, Pain Points]

OUTPUT SCHEMA:
{
  "interview_id": "string (INT-CWS-YYYYMMDD-LastName)",
  "date": "ISO 8601",
  "speaker": {"name": "string", "role": "string", "branch": "string"},
  "sections": [
```

```
    { "section_id": "string",
      "topic_category": "string (from controlled vocabulary)",
      "paragraphs": [
        { "para_id": "string (P001, P002...)",
          "verbatim": "string (exact text from transcript)",
          "contains_pain_point_signal": boolean } ] } ],
  "metadata": {
    "total_paragraphs": number,
    "pain_point_signals_detected": number,
    "processing_timestamp": "ISO 8601" }
}
```

8.2 Module M2 — Pain Point Extraction

```
SYSTEM PROMPT - MODULE M2: PAIN POINT EXTRACTION
You are a pain point extraction agent for ECCC's CWS branch.
INPUT: Structured transcript JSON from Module M1.
TASK: Identify all discrete pain points mentioned in the transcript.

RULES:
1. A pain point is a problem, challenge, frustration, gap, or unmet
   need explicitly expressed by the interviewee.
2. Each pain point must be supported by at least one verbatim citation.
3. Do NOT create pain points from implications or inferences.
4. If a pain point matches an existing one in the provided
   pain_point_registry, link to it rather than creating a duplicate.
5. Assign each pain point to one or more branches from the org structure.

PAIN POINT TAXONOMY (use for classification):
- Data Management: Data silos, legacy systems, data quality
- Tools & Systems: Outdated tools, poor UX, system limitations
- Process & Bureaucracy: Excessive approvals, red tape, manual processes
- Security & Compliance: Information security, classification confusion
- Skills & Capacity: Training gaps, technical skill shortages
- Collaboration: Silos, poor cross-team coordination
- Communication: Information flow, multiple channels, messaging

OUTPUT SCHEMA:
{
  "pain_points": [
    { "pain_point_id": "PP-CWS-NNN",
      "title": "string (concise, 5-10 words)",
      "description": "string (1-3 sentences)",
      "taxonomy_category": "string (from taxonomy above)",
      "affected_branches": ["string (branch codes)"],
      "citations": [
        { "interview_id": "string", "para_id": "string",
          "verbatim_quote": "string", "speaker": "string" } ],
      "linked_existing_id": "string or null",
      "extraction_confidence": number (0.0-1.0) } ]
}
```

8.3 Module M3 — Parameter Scoring Engine

This is the most critical module. It contains the full rubric and requires the agent to produce evidence chains:

```
SYSTEM PROMPT - MODULE M3: PARAMETER SCORING ENGINE
You are a scoring agent that applies a predefined rubric to assign
qualitative parameter values to pain points from ECCC CWS transcripts.
INPUT: Pain point JSON from M2 + full structured transcript from M1.
```

```
RUBRICS (you MUST use these exact definitions):

=== CRITICALITY (1-5) ===
1-Low: Inconvenience only. No mandate/compliance impact.
2-Moderate: Measurable inefficiency. Deadlines met, staff time wasted.
3-High: Recurring operational disruption. Deadlines at risk.
4-Severe: Direct threat to mandate. Compliance at risk.
5-Critical: Systemic failure. Public safety/legal/ministerial risk.

=== URGENCY (1-5) ===
1-Low: No time pressure. Can be addressed in 12+ months.
2-Moderate: Should be addressed within 6-12 months.
3-High: Active degradation. Should be addressed within 3-6 months.
4-Severe: Deadline approaching or conditions worsening. 1-3 months.
5-Immediate: Legal deadline, system failure, or safety issue. Now.

=== SCOPE ===
1-Narrow: Affects a single team or role.
2-Moderate: Affects a directorate (multiple teams).
3-Wide: Affects an entire branch (e.g., all of CWS).
4-Enterprise: Affects multiple branches or the entire department.

=== SENTIMENT INTENSITY (-1.0 to +1.0) ===
-1.0: Extreme frustration, anger, hopelessness
-0.5: Clear frustration, resigned tone
 0.0: Neutral, factual description
+0.5: Cautiously optimistic despite challenges
+1.0: Enthusiastic, sees opportunity

=== EVIDENCE DENSITY ===
1-Sparse: Single mention, no specifics.
2-Moderate: Multiple mentions OR one detailed account.
3-Rich: Multiple mentions across interviews with specific examples.

SCORING RULES:
1. Score ONLY based on evidence present in the citations.
2. For each score, identify the SPECIFIC rubric anchor that matches.
3. If evidence supports two adjacent levels, choose the LOWER.
4. Provide a reasoning chain explaining your selection.
5. Rate your own confidence (0.0-1.0) based on evidence clarity.

OUTPUT SCHEMA:
{
  "scored_pain_points": [
    { "pain_point_id": "string",
      "scores": {
        "criticality": {"value": int, "anchor_matched": "string"},
        "urgency": {"value": int, "anchor_matched": "string"},
        "scope": {"value": int, "anchor_matched": "string"},
        "sentiment_intensity": {"value": float, ...},
        "evidence_density": {"value": int, "anchor_matched": "string"} },
      "evidence_chains": { "criticality": [{"citation_ref": "string",
        "reasoning": "string"}], ... },
      "overall_confidence": float,
      "flags": ["string ('ADJACENT_LEVEL_TIE', 'LOW_EVIDENCE')"] } ],
  "scoring_metadata": {
    "rubric_version": "string", "model_version": "string",
    "prompt_hash": "string (SHA-256)", "timestamp": "ISO 8601",
    "temperature": float }
}
```

8.4 Module M4 — MCDA Computation

This module is deterministic and does not require an LLM call. It is pure computation that takes the scores from M3 and applies the weighted additive model:

```
PYTHON - MODULE M4: MCDA PRIORITY CALCULATOR
WEIGHTS = {
    "criticality": 0.30, "frequency": 0.20, "scope": 0.20,
    "urgency": 0.15, "sentiment_intensity": 0.10, "evidence_density": 0.05
}
RANGES = {
    "criticality": (1, 5), "frequency": (0, 1), "scope": (1, 4),
    "urgency": (1, 5), "sentiment_intensity": (-1, 1), "evidence_density": (1, 3)
}
PRIORITY_THRESHOLDS = [
    (0.75, "P1-Immediate"), (0.50, "P2-High"),
    (0.25, "P3-Medium"), (0.00, "P4-Low")
]

def normalize(value, min_val, max_val):
    return (value - min_val) / (max_val - min_val)

def compute_priority(scores: dict) -> dict:
    weighted_sum = 0.0
    decomposition = {}
    for param, weight in WEIGHTS.items():
        raw = scores[param]
        min_v, max_v = RANGES[param]
        norm = normalize(raw, min_v, max_v)
        contribution = norm * weight
        weighted_sum += contribution
        decomposition[param] = { "raw": raw, "normalized": round(norm, 3),
            "weight": weight, "contribution": round(contribution, 4) }
    priority_class = "P4-Low"
    for threshold, label in PRIORITY_THRESHOLDS:
        if weighted_sum >= threshold:
            priority_class = label; break
    return { "priority_score": round(weighted_sum, 4),
        "priority_class": priority_class, "decomposition": decomposition }
```

8.5 Module M5 — Audit Package Generation

The audit package is the governance deliverable. It bundles all intermediate results into a traceable, versioned package:

audit_manifest.json List of all files, their SHA-256 hashes, timestamps, and model versions.

raw_transcripts/ Original interview documents (immutable, hash-verified).

structured_transcripts/ M1 outputs — parsed JSON with paragraph IDs.

extracted_pain_points/ M2 outputs — pain points with citation chains.

scoring_results/ M3 outputs — all parameter scores with evidence chains and confidence.

priority_rankings/ M4 outputs — MCDA results with full decomposition.

rubric_version.json / weight_configuration.json The exact rubric used (version-controlled); the MCDA weights with approval metadata (who approved, when).

system_prompts/ All system prompts used, with SHA-256 hashes for exact reproduction.

reliability_report.json Cohen's kappa / Fleiss' kappa results from validation runs.

9. API Implementation — Step by Step

Prerequisites. An Anthropic API key, Python 3.10+, and the anthropic Python SDK. The agent can also be built in Node.js or called from a Claude Code environment.

9.1 Core agent orchestrator

```
PYTHON - CORE AGENT ORCHESTRATOR
import anthropic, hashlib, json
from datetime import datetime, timezone

client = anthropic.Anthropic()   # Uses ANTHROPIC_API_KEY env var
MODEL = "claude-sonnet-4-20250514"

def hash_prompt(prompt: str) -> str:
    return hashlib.sha256(prompt.encode()).hexdigest()

def call_module(system_prompt, user_content, temperature=0.0) -> dict:
    prompt_hash = hash_prompt(system_prompt)
    timestamp = datetime.now(timezone.utc).isoformat()
    response = client.messages.create(
        model=MODEL, max_tokens=8192,
        temperature=temperature,          # 0.0 for determinism
        system=system_prompt,
        messages=[{"role": "user", "content": user_content}])
    result = json.loads(response.content[0].text)
    result["_audit"] = {
        "model": MODEL, "prompt_hash": prompt_hash,
        "temperature": temperature, "timestamp": timestamp,
        "input_tokens": response.usage.input_tokens,
        "output_tokens": response.usage.output_tokens,
        "stop_reason": response.stop_reason }
    return result

def run_pipeline(transcript_text, pain_point_registry, weights) -> dict:
    m1 = call_module(M1_SYSTEM_PROMPT,
                     f"Parse this transcript:\n\n{transcript_text}")
    m2 = call_module(M2_SYSTEM_PROMPT, json.dumps({
        "structured_transcript": m1,
        "pain_point_registry": pain_point_registry }))
    m3 = call_module(M3_SYSTEM_PROMPT, json.dumps({
        "pain_points": m2["pain_points"],
        "structured_transcript": m1 }))
    m4 = compute_mcda_priorities(m3["scored_pain_points"], weights)
    return { "pipeline_run_id": f"RUN-{datetime.now():%Y%m%d-%H%M%S}",
        "modules": {"m1_ingestion": m1, "m2_extraction": m2,
                    "m3_scoring": m3, "m4_priority": m4} }
```

9.2 Ensemble validation (Option B)

To implement the multi-pass ensemble for validation, wrap the scoring module in a loop with varying temperatures:

```
PYTHON - ENSEMBLE SCORING WITH STATISTICAL CONVERGENCE
import numpy as np

def ensemble_score(pain_point_json, transcript_json, n_runs=5) -> dict:
    temperatures = [0.0, 0.1, 0.2, 0.1, 0.0][:n_runs]
    all_scores = {param: [] for param in PARAMETERS}
    for temp in temperatures:
        result = call_module(M3_SYSTEM_PROMPT, json.dumps({
```

```
            "pain_points": [pain_point_json],
            "structured_transcript": transcript_json }), temperature=temp)
        for param in PARAMETERS:
            all_scores[param].append(
                result["scored_pain_points"][0]["scores"][param]["value"])
    consensus = {}
    for param, scores in all_scores.items():
        arr = np.array(scores)
        consensus[param] = {
            "median": float(np.median(arr)), "mean": float(np.mean(arr)),
            "std": float(np.std(arr)),
            "cv": float(np.std(arr)/np.mean(arr)) if np.mean(arr) else 0,
            "iqr": float(np.percentile(arr,75) - np.percentile(arr,25)),
            "needs_human_review": bool(
                np.percentile(arr,75) - np.percentile(arr,25) > 1.0) }
    return consensus
```

9.3 Reliability testing

Run this to calculate Cohen's Kappa between two scoring runs and validate the kappa >= 0.80 governance threshold:

```
PYTHON - COHEN'S KAPPA CALCULATOR
from sklearn.metrics import cohen_kappa_score
import json

def validate_reliability(run_a_path, run_b_path, parameter) -> dict:
    with open(run_a_path) as f: run_a = json.load(f)
    with open(run_b_path) as f: run_b = json.load(f)
    scores_a = [pp["scores"][parameter]["value"] for pp in run_a["scored_pain_points"]]
    scores_b = [pp["scores"][parameter]["value"] for pp in run_b["scored_pain_points"]]
    kappa = cohen_kappa_score(scores_a, scores_b, weights="linear")
    if   kappa >= 0.81: interpretation = "Almost perfect"
    elif kappa >= 0.61: interpretation = "Substantial"
    elif kappa >= 0.41: interpretation = "Moderate"
    else: interpretation = "Fair or lower"
    return { "parameter": parameter, "cohens_kappa": round(kappa, 4),
        "interpretation": interpretation,
        "meets_governance_threshold": kappa >= 0.80,
        "recommendation": ("PASS" if kappa >= 0.80 else
            "FAIL: Refine rubric anchors for this parameter.") }
```

10. Data Governance and Ethics Audit Compliance

The audit-trail architecture. Every decision the agent makes must be traceable through an unbroken chain of evidence: Source Transcript -> Parsed Paragraph (M1) -> Pain Point + Citation (M2) -> Score + Rubric Anchor + Evidence Chain (M3) -> Priority + Decomposition (M4) -> Audit Package with Hashes & Timestamps (M5). This is the same chain-of-custody discipline the Origi trust framework applies to AI systems.

10.1 Governance requirements checklist

Reproducibility
temperature=0.0, SHA-256 prompt hashing, pinned model version. Verify: re-run with same inputs and prompts; compare output hashes.

Traceability

Every score links to verbatim citations with paragraph IDs. Verify: trace any priority score back to source transcript text.

Transparency
Full system prompts are versioned and included in the audit package. Verify: auditor can read the exact instructions given to the AI.

Bias detection
Ensemble detects inconsistencies; conservative scoring rule (lower level on ties). Verify: Fleiss' kappa across runs; monitor for systematic over/under-scoring.

Human override
All flagged items (IQR > 1.0, low confidence) require human adjudication. Verify: override log with justification stored in the audit package.

Data minimization
Only structured outputs are retained; raw API logs are ephemeral. Verify: audit package contains only necessary governance artifacts.

Version control
Rubric, weights, and prompts are version-controlled with semantic versioning. Verify: Git history or version manifest in the audit package.

Informed consent
Interviewees are informed that AI tools will assist in analysis. Verify: consent documentation referenced in the audit manifest.

10.2 Ethics considerations specific to interview analysis

Speaker anonymization
The agent should not use speaker names in the final priority outputs unless explicitly authorized. Use role-based identifiers (e.g., "Dir. SAR Implementation") instead.

Sentiment scores and power dynamics
A director expressing frustration carries different organizational weight than a junior analyst. The agent should not conflate positional authority with evidence quality — all citations are weighted equally regardless of the speaker's seniority.

Preventing confirmation bias
The rubric's conservative scoring rule (choose the lower level on ties) prevents the agent from inflating criticality. The ensemble method provides a statistical check against single-run anomalies.

Cultural sensitivity
Interview language varies by cultural background. Sentiment analysis should recognize that understated language (common in Canadian government communication) may mask high criticality; the rubric anchors include both explicit and understated trigger patterns.

11. Practical Walkthrough — CWS Interview Data

Three pain points extracted from the CWS interview transcripts demonstrate the full pipeline.

11.1 Outdated e-Permitting tool

Source: (Dir. SAR Implementation, June 9) — "Outdated e-permitting application built in ColdFusion. The DMTS does not help with this." Additional source references "outdated, difficult-to-navigate data systems."

```
Criticality     4 (Severe)   Unsupported tech (ColdFusion); 6+ yrs failed modernization
Urgency         4 (Severe)   Legal obligation under the Species at Risk Act
Scope           3 (Wide)     Entire CWS branch permitting across regions
Sentiment      -0.5          Resigned tone ("not looked at as transformative")
Evidence        2 (Moderate) Two interviewees, specific technical details
Frequency       2/10 (0.20)  Mentioned in 2 of the sampled interviews
```

```
Priority = 0.30(0.75)+0.20(0.20)+0.20(0.67)+0.15(0.75)+0.10(0.25)+0.05(0.50) =
0.562 -> P2-High
```

11.2 Information security confusion

Source (June 5) — "Staff are unsure about handling, storing, and sharing sensitive or secret documents." Additional sources corroborate across branches.

```
Criticality     3 (High)      Compliance risk but no active breach reported
Urgency         3 (High)      Active risk of improper handling; degrading
Scope           4 (Enterprise) Across multiple branches and roles
Sentiment      -0.4           Frustration + resignation
Evidence        3 (Rich)      Three interviewees (burner phones, GCSI, SharePoint)
Frequency       3/10 (0.30)   Mentioned in 3 of the sampled interviews
```

```
Priority = 0.30(0.50)+0.20(0.30)+0.20(1.00)+0.15(0.50)+0.10(0.30)+0.05(1.00) =
0.565 -> P2-High
```

11.3 Excessive process bureaucracy

Source: (Dir. Priority Species, June 10) — "40 steps to update website content"; "two-year delays in simple tasks like launching a webpage."

```
Criticality     3 (High)      Impedes public-facing service delivery
Urgency         3 (High)      Ongoing drain; cumulative impact significant
Scope           3 (Wide)      Multiple teams publishing content; branch-wide
Sentiment      -0.6           Strong frustration ("need a PhD to use it")
Evidence        3 (Rich)      Specific step counts, timelines, examples
Frequency       2/10 (0.20)   One interview but with extensive detail
```

```
Priority = 0.30(0.50)+0.20(0.20)+0.20(0.67)+0.15(0.50)+0.10(0.20)+0.05(1.00) =
0.469 -> P3-Medium
```

12. Implementation Roadmap

Phase 1 — Rubric Calibration (1 week)
Finalized rubric with stakeholder-approved anchors; weight configuration signed off by DGs.

Phase 2 — Agent Development (2 weeks)
Modules M1-M5 implemented and unit-tested; system prompts version-controlled.

Phase 3 — Pilot Run (1 week)
Process 5 transcripts; compute kappa scores; validate against human coders.

Phase 4 — Calibration (1 week)
Refine rubric anchors where kappa < 0.80; tune prompts; re-run pilot.

Phase 5 — Production Run (2 weeks)
Process all interview transcripts; generate full audit packages.

Phase 6 — Governance Review (1 week)
Present audit packages to ethics board; document human overrides.

Phase 7 — Reporting (1 week)
Generate pain-point priority report for DG/ADM; populate the Citations sheet.

13. Conclusion

This guide has presented four mathematically grounded approaches for determining qualitative parameters from interview transcripts, each with increasing statistical rigour. The recommended hybrid approach — rubric-anchored scoring fed into MCDA — provides the optimal balance of repeatability, traceability, and practical implementation. The complete agent pipeline, from transcript ingestion through audit-package generation, ensures that every criticality score, every priority ranking, and every qualitative judgment can be traced to specific evidence, reproduced with identical inputs, and defended in a governance audit.

The mathematical tools — Cohen's Kappa for inter-rater reliability, the weighted additive model for multi-criteria prioritization, Bayesian updating for evolving evidence, and Monte Carlo sensitivity analysis for robustness — transform what is typically a subjective exercise into a disciplined, auditable analytical process. It is the same conviction that runs through the MIOVSKI GROUP analytical frameworks: a score is worth only the evidence chain behind it.

END OF DOCUMENT