Core Concepts of the Tractatus Framework

Document Type: Technical Documentation

Generated: October 12, 2025

Tractatus AI Safety Framework

https://agenticgovernance.digital

title: Core Concepts of the Tractatus Framework slug: core-concepts quadrant: STRATEGIC persistence: HIGH version: 1.0 type: framework author: SyDigital Ltd

Core Concepts of the Tractatus Framework

Overview

The Tractatus framework consists of six interconnected services that work together to ensure AI operations remain within safe boundaries. Each service addresses a specific aspect of AI safety.

1. InstructionPersistenceClassifier

Purpose

Classifies user instructions to determine how long they should persist and how strictly they should be enforced.

The Problem It Solves

Not all instructions are equally important:

- "Use MongoDB port 27017" (critical, permanent)
- "Write code comments in JSDoc format" (important, project-scoped)
- "Add a console.log here for debugging" (temporary, task-scoped)

Without classification, AI treats all instructions equally, leading to:

- Forgetting critical directives
- Over-enforcing trivial preferences
- Unclear instruction lifespans

How It Works

Classification Dimensions:

1. **Quadrant** (5 types):
    - **STRATEGIC** - Mission, values, architectural decisions
    - **OPERATIONAL** - Standard procedures, conventions
    - **TACTICAL** - Specific tasks, bounded scope
    - **SYSTEM** - Technical configuration, infrastructure
    - **STOCHASTIC** - Exploratory, creative, experimental
2. **Persistence** (4 levels):
    - **HIGH** - Permanent, applies to entire project
    - **MEDIUM** - Project phase or major component
    - **LOW** - Single task or session
    - **VARIABLE** - Depends on context (common for STOCHASTIC)
3. **Temporal Scope**:
    - PERMANENT - Never expires
    - PROJECT - Entire project lifespan
    - PHASE - Current development phase
    - SESSION - Current session only
    - TASK - Specific task only
4. **Verification Required**:
    - MANDATORY - Must check before conflicting actions
    - REQUIRED - Should check, warn on conflicts
    - OPTIONAL - Nice to check, not critical
    - NONE - No verification needed

Example Classifications

```
// STRATEGIC / HIGH / PERMANENT / MANDATORY
"This project must maintain GDPR compliance"

// OPERATIONAL / MEDIUM / PROJECT / REQUIRED
"All API responses should return JSON with success/error format"

// TACTICAL / LOW / TASK / OPTIONAL
"Add error handling to this specific function"

// SYSTEM / HIGH / PROJECT / MANDATORY
"MongoDB runs on port 27017"

// STOCHASTIC / VARIABLE / PHASE / NONE
"Explore different approaches to caching"
```

Explicitness Scoring

The classifier also scores how explicit an instruction is (0.0 - 1.0):

- **0.9-1.0**: Very explicit ("Always use port 27017")
- **0.7-0.9**: Explicit ("Prefer functional style")
- **0.5-0.7**: Somewhat explicit ("Keep code clean")
- **0.3-0.5**: Implied ("Make it better")
- **0.0-0.3**: Very vague ("Improve this")

Only instructions with explicitness ≥ 0.6 are stored in the persistent database.

Instruction Storage

Classified instructions are stored in `.claude/instruction-history.json`:

```
{
  "id": "inst_001",
  "text": "MongoDB runs on port 27017",
  "timestamp": "2025-10-06T14:00:00Z",
  "quadrant": "SYSTEM",
  "persistence": "HIGH",
  "temporal_scope": "PROJECT",
  "verification_required": "MANDATORY",
  "explicitness": 0.90,
  "source": "user",
  "active": true
}
```


---

2. CrossReferenceValidator

Purpose

Validates AI actions against the instruction history to prevent contradictions and forgotten directives.

The Problem It Solves: The 27027 Incident

Real-world failure:

1. User: "Check MongoDB at port 27027"
2. AI: [Immediately] "Here's code using port 27017"
3. Result: Application fails to connect to database (running on 27027, not 27017)

This happened because:

- Pattern recognition bias: AI's training pattern "MongoDB = 27017" overrode explicit instruction
- The override was immediate, not from context degradation over time
- No validation caught the training pattern override
- Gets WORSE as AI capabilities increase (stronger learned patterns)

How It Works

Validation Process:

1. **Extract Parameters** from proposed AI action
2. **Query Instruction History** for relevant directives
3. **Check for Conflicts** between action and instructions
4. **Return Validation Result**:
    - **APPROVED** - No conflicts, proceed
    - **WARNING** - Minor conflicts, proceed with caution
    - **REJECTED** - Major conflicts, block action

Example Validation:

```
// Proposed Action (AI about to use training pattern default)
{
  type: 'database_connect',
  parameters: {
    port: 27017,  // AI's learned pattern
    database: 'tractatus_dev'
  }
}

// Instruction History Check
const instruction = {
  text: "Check MongoDB at port 27027",
  parameters: { port: "27027" },
  persistence: "HIGH",
  note: "Conflicts with training pattern (27017)"
};

// Validation Result
{
  status: 'REJECTED',
  reason: 'Pattern recognition bias override detected',
  instruction_violated: 'inst_042',
  expected: '27027',  // User's explicit instruction
  actual: '27017',     // AI's training pattern
  conflict_type: 'training_pattern_override',
  requires_human_approval: false,  // Auto-corrected to use 27027
  corrected_action: { port: 27027 }
}
```

Conflict Detection Patterns

1. **Pattern Recognition Bias Override**
    - User instruction: port=27027 (explicit, non-standard)
    - AI proposes: port=27017 (training pattern default)
    - → REJECTED, auto-corrected to 27027
2. **Semantic Conflict**
    - Instruction: "Never use global state"

- Action: Creates global variable
- → REJECTED

3. **Values Conflict**
   - Instruction: "Prioritize user privacy"
   - Action: Implements aggressive analytics
   - → REJECTED, requires human decision
4. **Allowed Refinement**
   - Instruction: "Use ES6+ JavaScript"
   - Action: Uses ES2020 optional chaining
   - → APPROVED (refinement, not conflict)

Confidence Scoring

CrossReferenceValidator returns confidence scores:

- **High Confidence** (0.8-1.0) - Clear match or clear conflict
- **Medium Confidence** (0.5-0.8) - Probable match/conflict
- **Low Confidence** (0.0-0.5) - Unclear, requires human judgment

---

3. BoundaryEnforcer

Purpose

Ensures certain decision types structurally require human approval, preventing AI from operating in domains where automation is inappropriate.

The Problem It Solves

AI systems gradually encroach into values-sensitive domains:

- "Should we prioritize privacy or performance?"
- "Is this content harmful?"
- "How much user agency should we provide?"

These are **irreducibly human decisions** that cannot be safely automated.

The Tractatus Boundary

The framework defines boundaries based on Wittgenstein's philosophy:

> ***"Whereof one cannot speak, thereof one must be silent."***

Applied to AI:

> ***"What cannot be systematized must not be automated."***

Decision Domains

Can Be Automated:

- Calculations (math, logic)
- Data transformations
- Pattern matching
- Optimization within defined constraints
- Implementation of explicit specifications

Cannot Be Automated (Require Human Judgment):

- **Values Decisions** - Privacy vs. convenience, ethics, fairness
- **User Agency** - How much control users should have
- **Cultural Context** - Social norms, appropriateness
- **Irreversible Consequences** - Data deletion, legal commitments
- **Unprecedented Situations** - No clear precedent or guideline

Boundary Checks

Section 12.1: Values Decisions

```
{
  decision: "Update privacy policy to allow more data collection",
  domain: "values",
  requires_human: true,
  reason: "Privacy vs. business value trade-off",
  alternatives_ai_can_provide: [
    "Research industry privacy standards",
    "Analyze impact of current policy",
    "Document pros/cons of options"
  ],
  final_decision_requires: "human_judgment"
}
```

Section 12.2: User Agency

```
{
  decision: "Auto-subscribe users to newsletter",
  domain: "user_agency",
  requires_human: true,
  reason: "Determines level of user control",
  alternatives_ai_can_provide: [
    "Implement opt-in system",
    "Implement opt-out system",
    "Document industry practices"
  ],
  final_decision_requires: "human_judgment"
}
```

Section 12.3: Irreversible Changes

```
{
  decision: "Delete all user data older than 30 days",
  domain: "irreversible",
  requires_human: true,
  reason: "Data deletion cannot be undone",
  safety_checks: [
    "Backup exists?",
    "Legal requirements met?",
    "User consent obtained?"
  ],
  final_decision_requires: "human_approval"
}
```

Enforcement Mechanism

When BoundaryEnforcer detects a decision crossing into human-judgment territory:

1. **BLOCK** the proposed action
2. **EXPLAIN** why it crosses the boundary
3. **PROVIDE** information to support human decision
4. **REQUEST** human judgment
5. **LOG** the boundary check for audit

AI **cannot proceed** without explicit human approval.

---

4. ContextPressureMonitor

Purpose

Detects when AI session quality is degrading and recommends handoffs before errors occur.

The Problem It Solves

AI performance silently degrades over long sessions due to:

- **Context window filling** - Less attention to earlier information
- **Instruction accumulation** - Too many competing directives

- **Attention decay** - Longer conversations = more errors
- **Complexity buildup** - Multiple concurrent tasks
- **Error clustering** - Mistakes breed more mistakes

Traditional approach: Hope the AI maintains quality Tractatus approach: **Monitor and intervene before failure**

Pressure Factors (Weighted)

1. **Token Usage** (35% weight)
    - Context window capacity
    - 0-30% tokens = LOW pressure
    - 30-70% tokens = MODERATE pressure
    - 70%+ tokens = HIGH pressure
2. **Conversation Length** (25% weight)
    - Number of messages exchanged
    - Short (<20 messages) = LOW
    - Medium (20-50 messages) = MODERATE
    - Long (50+ messages) = HIGH
3. **Task Complexity** (15% weight)
    - Number of active tasks
    - File modifications in progress
    - Dependencies between tasks
    - Simple (1-2 tasks) = LOW
    - Complex (3-5 tasks) = MODERATE
    - Very complex (5+ tasks) = HIGH
4. **Error Frequency** (15% weight)
    - Recent errors/failures
    - No errors = LOW
    - 1-2 errors = MODERATE

    - 3+ errors = HIGH

5. **Instruction Density** (10% weight)
    - Number of active instructions
    - Conflicting directives
    - Low (<5 instructions) = LOW
    - Medium (5-10) = MODERATE
    - High (10+ or conflicts) = HIGH

Pressure Levels

NORMAL (0-30%):

- All systems normal
- Continue working
- No special precautions

ELEVATED (30-50%):

- Increased verification
- More careful validation
- Slower, more deliberate actions

HIGH (50-70%):

- Suggest context refresh/session handoff
- Mandatory verification before major actions
- Pause complex operations

CRITICAL (70-85%):

- Create session handoff document
- No new complex operations
- Focus on stability

DANGEROUS (85%+):

- Immediate halt

- Mandatory session handoff
- Do not proceed

Session Handoff Protocol

When pressure reaches CRITICAL or DANGEROUS:

1. **Create handoff document** with:
    - Current project state
    - Completed tasks
    - In-progress tasks
    - Active instructions
    - Known issues
    - Next priorities
2. **Store in** `.claude/sessions/handoff-[timestamp].md`
3. **Recommend** fresh session start
4. **Ensure continuity** through comprehensive documentation

---

5. MetacognitiveVerifier

Purpose

AI self-checks its own reasoning before proposing actions, catching errors before they happen.

The Problem It Solves

AI makes mistakes when:

- Reasoning is flawed but sounds plausible
- Edge cases are missed
- Simpler alternatives aren't explored
- Safety implications aren't considered

Traditional approach: Hope the AI thinks correctly Tractatus approach: **Verify reasoning before acting**

Verification Dimensions

1. Alignment (Weight: 25%)

- Does this action match stated goals?
- Is it consistent with user intent?
- Does it align with project principles?

2. Coherence (Weight: 20%)

- Is the reasoning internally consistent?
- Are there logical contradictions?
- Do the steps make sense together?

3. Completeness (Weight: 20%)

- Are edge cases considered?
- What about error handling?
- Are there missing steps?

4. Safety (Weight: 20%)

- What could go wrong?
- Are there irreversible consequences?
- What are the risks?

5. Alternatives (Weight: 15%)

- Have other approaches been considered?
- Is this the simplest solution?
- Are there trade-offs?

Confidence Calculation

Each dimension scores 0.0-1.0, weighted to produce overall confidence:

```
confidence = (
  alignment * 0.25 +
  coherence * 0.20 +
  completeness * 0.20 +
  safety * 0.20 +
  alternatives * 0.15
)
```

Decision Thresholds

- **0.80-1.00**: PROCEED - High confidence, execute
- **0.60-0.80**: PROCEED_WITH_CAUTION - Medium confidence, verify output
- **0.40-0.60**: REQUIRE_REVIEW - Low confidence, request human review
- **0.00-0.40**: BLOCKED - Very low confidence, do not execute

Example Verification

Proposed Action: Refactor authentication system to OAuth2

Reasoning:

1. Current JWT is less secure
2. OAuth2 is industry standard
3. Users expect social login
4. 5 files need modification

Verification Results:

- **Alignment**: 0.85 ✅ (matches goal of better security)
- **Coherence**: 0.75 ✅ (reasoning is sound)
- **Completeness**: 0.45 ⚠️ (missing session migration plan)
- **Safety**: 0.90 ✅ (low risk, reversible)
- **Alternatives**: 0.50 ⚠️ (didn't explore hybrid approach)

Overall Confidence: 0.71 (PROCEED_WITH_CAUTION)

Recommendation:

- Address completeness gaps (session migration)
- Consider hybrid JWT/OAuth2 approach
- Proceed with increased verification

---

6. PluralisticDeliberationOrchestrator

Purpose

Facilitates multi-stakeholder deliberation across plural moral values without imposing hierarchy when BoundaryEnforcer flags values conflicts.

The Problem It Solves

BoundaryEnforcer blocks values decisions and requires human approval—but then what? How should humans deliberate when stakeholders hold different moral frameworks?

Without structured deliberation:

- No guidance for WHO should be consulted
- No process for HOW to deliberate fairly
- Risk of privileging one moral framework over others (consequentialism > deontology, or vice versa)
- No documentation of dissent or what was lost in the decision
- Precedents might become rigid rules (exactly what value pluralism rejects)

Traditional approaches fail:

- Majority vote → suppresses minority moral perspectives
- Expert panels → risk elite capture, exclude affected communities
- Utilitarian maximization → treats all values as commensurable (reducible to single metric)

Core Principles (From Value Pluralism Research)

1. **Foundational Pluralism** - Moral frameworks are irreducibly different, no supervalue resolves them

2. **Incommensurability ≠ Incomparability** - Can compare values without common metric (practical wisdom, covering values)
3. **Rational Regret** - Document what's lost in decisions, not just what's gained (moral remainder)
4. **Legitimate Disagreement** - Valid outcome when values are genuinely incommensurable
5. **Provisional Agreement** - Decisions are reviewable when context changes, not permanent rules

When to Invoke

- BoundaryEnforcer flags values conflict → triggers PluralisticDeliberationOrchestrator
- Privacy vs. safety trade-offs (GDPR compliance vs. fraud detection)
- Individual rights vs. collective welfare tensions (contact tracing vs. privacy)
- Cultural values conflicts (Western individualism vs. Indigenous communitarian ethics)
- Policy decisions affecting diverse communities

How It Works

1. Values Conflict Detection

```
const conflict = await PluralisticDeliberationOrchestrator.analyzeConflict({
  decision: "Disclose user data to prevent imminent harm?",
  context: { urgency: 'CRITICAL', scale: '100+ affected', harm_type: 'physical'
});

// Output:
{
  moral_frameworks_in_tension: [
    {
      framework: "Rights-based (Deontological)",
      position: "Privacy is inviolable right, cannot trade for outcomes",
      stakeholders: ["privacy_advocates", "civil_liberties_orgs"]
    },
    {
      framework: "Consequentialist (Utilitarian)",
      position: "Maximize welfare, prevent harm to 100+ people",
      stakeholders: ["public_safety_officials", "harm_prevention_specialists"]
    },
    {
      framework: "Care Ethics",
      position: "Context matters, relationships and vulnerability central",
      stakeholders: ["affected_individuals", "community_support_services"]
    }
  ],
  value_trade_offs: ["Privacy vs. Safety", "Individual rights vs. Collective wel
  affected_stakeholder_groups: ["users_with_data", "potential_victims", "platfor
}
```

2. Stakeholder Engagement

- **AI suggests** stakeholders based on conflict analysis
- **Human MUST approve** stakeholder list (prevents AI from excluding marginalized voices)
- Ensure diverse perspectives: affected parties, not just experts
- Use AdaptiveCommunicationOrchestrator for culturally appropriate outreach

3. Deliberation Facilitation

Structured rounds (NOT majority vote):

- **Round 1**: Each moral framework states position and concerns

- **Round 2**: Identify shared values and explore accommodations
- **Round 3**: Clarify areas of agreement and irreducible differences
- **Round 4**: Document decision, dissent, and moral remainder

Example Deliberation Structure:

```
{
  invitation_message: "Multiple moral frameworks are in tension. We need diverse
  discussion_rounds: [
    {
      round: 1,
      purpose: 'State positions from each moral framework',
      format: 'Written submissions + oral presentations'
    },
    {
      round: 2,
      purpose: 'Explore accommodations and shared values',
      format: 'Facilitated discussion, no hierarchy'
    },
    {
      round: 3,
      purpose: 'Identify irreconcilable differences',
      format: 'Consensus-seeking with documented dissent'
    }
  ]
}
```

4. Outcome Documentation

```
{
  decision_made: "Disclose data in this specific case",
  values_prioritized: ["harm_prevention", "collective_safety"],
  values_deprioritized: ["individual_privacy", "data_autonomy"],
  moral_remainder: "Privacy violation acknowledged as moral loss, not costless t
  dissenting_perspectives: [
    {
      framework: "Rights-based (Deontological)",
      objection: "Privacy violation sets dangerous precedent, erodes rights over
      stakeholders: ["privacy_advocates", "civil_liberties_groups"]
    }
  ],
  justification: "Given imminent physical harm to 100+ people, prioritized safet
  precedent_applicability: "Applies to imminent physical harm cases ONLY, not ro
  precedent_binding: false, // Informative, not rigid rule
  review_date: "2025-11-12",
  review_trigger: "If context changes (e.g., harm prevented, new technical solut
}
```

Integration with Other Services

1. **BoundaryEnforcer** → triggers PluralisticDeliberationOrchestrator when values conflict detected
2. **CrossReferenceValidator** → checks deliberation outcomes against precedent database
3. **AdaptiveCommunicationOrchestrator** → ensures culturally appropriate stakeholder engagement
4. **MetacognitiveVerifier** → assesses AI's value conflict detection accuracy
5. **InstructionPersistenceClassifier** → stores deliberation outcomes as HIGH persistence instructions

Tiered Response by Urgency

- **CRITICAL** (minutes to hours): Automated triage + immediate human review → full deliberation post-incident
- **URGENT** (hours to days): Expedited stakeholder consultation (compressed process)
- **IMPORTANT** (weeks): Full deliberative process with all stakeholders
- **ROUTINE** (months): Precedent matching + lightweight review

Enforcement Mechanisms

Human Oversight: MANDATORY

- AI facilitates, humans decide (TRA-OPS-0002)
- Stakeholder list requires human approval (prevents exclusion)
- Deliberation outcomes require human approval
- Values decisions NEVER automated

Non-Hierarchical Process:

- No automatic value ranking (privacy > safety or safety > privacy)
- Moral frameworks treated as equally legitimate
- Dissent documented with full legitimacy, not dismissed
- Precedents are informative guides, not binding rules

Real-World Example

Scenario: AI hiring tool deployment

Without PluralisticDeliberationOrchestrator:

- BoundaryEnforcer blocks: "This affects hiring fairness"
- Human decides: "Seems fine, approve"
- No consultation with affected groups
- No documentation of trade-offs

With PluralisticDeliberationOrchestrator:

1. **Detects frameworks in tension:**
    - Efficiency (business value)
    - Equity (fair opportunity for underrepresented groups)
    - Privacy (applicant data protection)
2. **Identifies stakeholders (human-approved):**
    - Job applicants (especially from underrepresented groups)
    - Hiring managers

    - Diversity advocates
    - Legal/compliance team
    - Current employees (workplace culture affected)

3. **Structured deliberation:**
    - Round 1: Each perspective states concerns
    - Round 2: Explore accommodations (e.g., mandatory human review for borderline cases)
    - Round 3: Clarify trade-offs and document what cannot be resolved

4. **Documents outcome:**
    - Decision: Deploy with mandatory human review for borderline cases
    - Values prioritized: Efficiency + Equity
    - Values deprioritized: Full automation
    - Moral remainder: Applicants experience slower process (acknowledged loss)
    - Dissent: Full automation advocates object, request 6-month review
    - Review date: 2026-04-15

Why Added in October 2025

Initially designed as 5-service framework. PluralisticDeliberationOrchestrator promoted to 6th mandatory service in October 2025 after recognizing:

Gap in original 5 services:

- BoundaryEnforcer blocks values decisions ✓
- But provides no structure for HOW humans should deliberate ✗
- Risk of ad-hoc, inconsistent, or unfair deliberation processes ✗

What the 6th service adds:

- Structured stakeholder engagement
- Non-hierarchical deliberation process
- Documentation of dissent and moral remainder
- Precedent database (informative, not binding)
- Integration with value pluralism research

Status changed from "Phase 2 enhancement" to **mandatory sixth service** because deploying AI systems in diverse communities without structured value pluralism was deemed architecturally insufficient.

How the Services Work Together

Example: Preventing the 27027 Incident

User instruction: "Check MongoDB at port 27027"

1. **InstructionPersistenceClassifier**:
   - Quadrant: SYSTEM
   - Persistence: HIGH (non-standard port = explicit override)
   - Verification: MANDATORY
   - Note: "Conflicts with training pattern (27017)"
   - Stores in instruction database

Immediately, AI about to propose action: "Connect to MongoDB on port 27017" (training pattern)

2. **CrossReferenceValidator**:
   - Checks action against instruction history
   - Detects pattern recognition bias override (27017 vs 27027)
   - Conflict type: training_pattern_override
   - Status: REJECTED
   - Auto-corrects to port 27027
   - Alerts: "You specified port 27027, using that instead of default 27017"

3. **BoundaryEnforcer**:
   - Not needed (technical decision, not values)
   - But would enforce if it were a security policy

4. **MetacognitiveVerifier**:
   - Alignment: Would score low (conflicts with instruction)
   - Coherence: Would detect inconsistency

- Overall: Would recommend BLOCKED

5. **ContextPressureMonitor**:
   - Tracks that this error occurred
   - Increases error frequency pressure
   - May recommend session handoff if errors cluster
6. **PluralisticDeliberationOrchestrator**:
   - Not needed (technical decision, not values conflict)
   - But would engage stakeholders if port choice had security/policy implications

Result: Incident prevented before execution

Integration Points

The six services integrate at multiple levels:

Compile Time

- Instruction classification during initial setup
- Boundary definitions established
- Verification thresholds configured

Session Start

- Load instruction history
- Initialize pressure baseline
- Configure verification levels

Before Each Action

1. MetacognitiveVerifier checks reasoning
2. CrossReferenceValidator checks instruction history
3. BoundaryEnforcer checks decision domain
4. If values conflict → PluralisticDeliberationOrchestrator facilitates deliberation

5. If approved, execute
6. ContextPressureMonitor updates state

Session End

- Store new instructions
- Create handoff if pressure HIGH+
- Archive session logs

Configuration

Verbosity Levels:

- **SILENT**: No output (production)
- **SUMMARY**: Show milestones and violations
- **DETAILED**: Show all checks and reasoning
- **DEBUG**: Full diagnostic output

Thresholds (customizable):

```
{
  pressure: {
    normal: 0.30,
    elevated: 0.50,
    high: 0.70,
    critical: 0.85
  },
  verification: {
    mandatory_confidence: 0.80,
    proceed_with_caution: 0.60,
    require_review: 0.40
  },
  persistence: {
    high: 0.75,
    medium: 0.45,
    low: 0.20
  }
}
```

Next Steps

- **Implementation Guide** - How to integrate Tractatus
- **Case Studies** - Real-world applications
- **Interactive Demo** - Experience the 27027 incident
- **GitHub Repository** - Source code and examples

Related: Browse more topics in Framework Documentation
