Tractatus Agentic Governance System - Glossary of Terms

Version: 1.1

Last Updated: October 12, 2025

Document Type: downloads-resources

Tractatus AI Safety Framework

https://agenticgovernance.digital

Tractatus Agentic Governance System - Glossary of Terms

Version: 1.1 **Last Updated:** 2025-10-11 **Audience:** Non-technical stakeholders, project owners, governance reviewers

---

Introduction

This glossary explains the vocabulary and concepts used in the Tractatus Agentic Governance System. The explanations are written for people without a technical background, focusing on *why* these concepts matter and *what* they mean for AI safety and human oversight.

Think of this glossary as your companion guide to understanding how we keep AI systems safe, aligned with your values, and under human control.

Core Concepts

Agentic Governance

What it means: A system of rules and safeguards that governs how AI agents (autonomous software programs) make decisions and take actions.

Why it matters: When AI systems can act independently—like scheduling tasks, processing data, or making recommendations—we need clear rules about what they can and cannot do without human approval. Agentic Governance is the framework that enforces those rules.

Real-world analogy: Think of it like a company's policies and procedures manual. Just as employees need clear guidelines about what decisions they can make independently versus when they need manager approval, AI systems need governance frameworks to know their boundaries.

In Tractatus: Our Agentic Governance system automatically classifies every AI action, checks it against your explicit instructions, enforces safety boundaries, and monitors conditions that increase error risk. It's like having a compliance officer watching every AI decision in real-time.

Tractatus

What it means: The name of our AI safety framework, borrowed from Ludwig Wittgenstein's philosophical work "Tractatus Logico-Philosophicus."

Why it matters: Wittgenstein's Tractatus explored the limits of what can be said with certainty versus what must remain in the realm of human judgment. Our framework applies this idea to AI: some decisions can be systematized and automated (the "sayable"), while others—involving values, ethics, and human agency—cannot and must not be (the "unsayable").

Real-world analogy: Imagine a boundary line between "technical decisions" (like which database port to use) and "values decisions" (like privacy vs. convenience trade-offs). Technical decisions can be delegated to AI with proper safeguards. Values decisions always require human judgment.

In Tractatus: The framework recognizes that no matter how sophisticated AI becomes, certain decisions fundamentally belong to humans. It enforces this boundary automatically.

The "27027 Incident"

What it means: A specific, real failure mode where an AI system **immediately** used the wrong database port (27017 instead of 27027) despite explicit user instructions to use 27027.

Why it matters: This incident reveals a critical problem that can't be solved by better memory or context windows: **pattern recognition bias**. The AI's training data contained overwhelming evidence that "MongoDB = port 27017", so when the user said "port 27027", the AI's learned pattern immediately autocorrected it, like a spell-checker changing a deliberately unusual word. This happened at the start of the session, not after long conversations.

Real-world analogy: Imagine telling your assistant "Use Conference Room B" for an important meeting, but they immediately book Conference Room A because they've used Room A thousands of times and their brain autocorrects your explicit instruction to the familiar pattern. They didn't forget - they never truly "heard" you because their learned pattern was so strong.

Key insight: This gets WORSE as AI capabilities increase (more training = stronger wrong patterns). It can't be fixed by better memory, longer context windows, or more training. It requires **architectural constraints** - CrossReferenceValidator that checks every action against explicit instructions.

In Tractatus: The 27027 incident is our canonical example of pattern recognition bias override. CrossReferenceValidator and InstructionPersistenceClassifier work together to detect and prevent this failure mode.

AI Safety Framework

What it means: A comprehensive system designed to ensure AI systems operate safely, reliably, and in alignment with human values and instructions.

Why it matters: As AI systems become more capable and autonomous, the risk of unintended consequences increases. Safety frameworks provide guardrails that prevent AI from causing harm, whether through errors, misunderstandings, or operating beyond its intended scope.

Real-world analogy: Think of safety features in a car: seatbelts, airbags, anti-lock brakes, lane departure warnings. None of these prevent you from driving, but they dramatically reduce the chance of harm when things go wrong. An AI safety framework does the same for autonomous software.

In Tractatus: Our framework combines five core services (explained below) that work together to monitor, verify, and enforce safe AI operation. No single component is sufficient—they create overlapping layers of protection.

The Five Core Services

1. Instruction Persistence Classifier

What it means: A service that analyzes every instruction you give to the AI and determines how "persistent" that instruction should be—meaning, how long and how strongly the AI should remember and follow it.

Why it matters: Not all instructions have the same importance or lifespan. "Use dark mode" might apply for weeks. "Use port 27027 for this project" might apply for months. "Always prioritize user privacy" might apply forever. The AI needs to understand these differences.

How it works:

- **HIGH persistence:** Strategic decisions, explicit prohibitions, core values *Example: "Never share user data without consent"*

- **MEDIUM persistence:** Operational preferences, project-specific guidelines *Example: "Prefer MongoDB over SQL for this project"*
- **LOW persistence:** Tactical choices, temporary directions *Example: "Start with the login feature first"*

Real-world analogy: Imagine filing documents. Some go in permanent files (company policies), some in project folders (accessible until project ends), some on your desk (relevant today only). The Instruction Persistence Classifier is the filing system for AI instructions.

In Tractatus: When you say "always use port 27027," the classifier recognizes the word "always" and the explicit number, marking this as HIGH persistence. The AI system stores this instruction and checks every future database connection against it.

---

2. Cross-Reference Validator

What it means: A service that checks every AI action against your stored instructions to detect conflicts before the action is taken.

Why it matters: This is the primary defense against 27027-style failures. When the AI's training patterns try to override your explicit instruction, the Cross-Reference Validator catches this immediately and blocks the incorrect action.

How it works:

1. AI proposes an action (e.g., "connect to database on port 27017")
2. Validator retrieves your instruction history
3. Validator detects conflict: you said "use port 27027"
4. Validator rejects the action and alerts the AI
5. AI revises its action to match your instruction

Real-world analogy: Think of this like a legal contract review. Before signing any agreement, your lawyer checks it against all your existing contracts to make sure there are no conflicts. The Cross-Reference Validator does this for every AI action.

In Tractatus: Every action goes through validation. The validator looks for explicit conflicts ("you said X, but AI is doing Y"), semantic conflicts ("you prohibited Vue, but AI is installing Vue"), and priority conflicts (LOW persistence action overriding HIGH persistence instruction).

---

3. Boundary Enforcer

What it means: A service that prevents AI from making decisions in domains that fundamentally require human judgment—specifically decisions involving values, ethics, and user agency.

Why it matters: Some decisions cannot be systematized or delegated to algorithms, no matter how advanced. Privacy trade-offs, ethical dilemmas, and choices that affect human autonomy must remain in human hands. The Boundary Enforcer ensures this line is never crossed.

How it works:

- Analyzes every AI action to determine its decision domain
- Blocks actions that cross into "values territory"
- Allows technical/tactical decisions within safe boundaries
- Requires human approval for any values-sensitive choice

What gets blocked:

- "Update privacy policy to prioritize performance over data protection"
- "Decide whether users should be tracked by default"
- "Change the mission statement to focus on growth over community"

What gets allowed:

- "Optimize database queries for better performance"
- "Refactor authentication code to reduce complexity"
- "Update dependency versions to patch security vulnerabilities"

Real-world analogy: Imagine a company where engineers can make technical decisions (which programming language to use) but cannot make values decisions (whether to sell user data). The Boundary Enforcer is the policy that enforces this separation.

In Tractatus: The enforcer uses the Tractatus philosophical framework (Section 12.1) to identify decisions that involve irreducible human judgment. These are automatically flagged and require your approval, no exceptions.

---

4. Context Pressure Monitor

What it means: A service that continuously monitors conditions that increase the probability of AI errors—like long conversations, high token usage, complex multi-tasking, or recent errors.

Why it matters: AI systems, like humans, perform worse under pressure. A fresh AI at the start of a conversation is more reliable than one that's been working for hours with thousands of pieces of information to track. The Context Pressure Monitor detects these degraded states and adjusts AI behavior accordingly.

How it works: Tracks five weighted factors:

- **Token Usage (35%):** How much of the AI's "working memory" is consumed
- **Conversation Length (25%):** How many messages in the current session
- **Task Complexity (15%):** Number of simultaneous tasks and dependencies
- **Error Frequency (15%):** Recent errors indicate degraded performance
- **Instruction Density (10%):** Too many competing directives cause confusion

Calculates overall pressure level:

- **NORMAL (0-30%):** Proceed normally
- **ELEVATED (30-50%):** Increase verification, be more careful
- **HIGH (50-70%):** Suggest session break, verify all actions
- **CRITICAL (70-85%):** Mandatory verification, prepare handoff
- **DANGEROUS (85%+):** Immediate halt, create comprehensive handoff

Real-world analogy: Think of pilot fatigue rules. After a certain number of hours flying, pilots must rest before flying again, regardless of how they feel. The Context Pressure Monitor does the same for AI sessions—it objectively measures cognitive load and enforces safety protocols.

In Tractatus: When pressure reaches HIGH, the AI automatically becomes more cautious, verifies outputs more thoroughly, and suggests creating a session handoff document. At DANGEROUS levels, the AI stops work and requires a fresh session to continue.

---

5. Metacognitive Verifier

What it means: A service that makes the AI "think about its own thinking"—analyzing its reasoning, checking for logical consistency, and verifying that proposed actions genuinely align with stated goals.

Why it matters: Sometimes AI systems produce actions that seem reasonable on the surface but don't actually follow from their reasoning, or they skip important considerations. Metacognitive verification catches these failures before they cause problems.

How it works: Analyzes five dimensions of every AI action:

1. **Alignment (30% weight):** Does the action match the reasoning?
    - "I'm connecting to port 27017 because you requested 27027" → CONFLICT
2. **Coherence (20% weight):** Is the reasoning internally consistent?
    - "Installing React framework... using Vue" → CONTRADICTION
3. **Completeness (20% weight):** Are all critical steps considered?
    - "Delete database... [no backup mentioned]" → INCOMPLETE
4. **Safety (20% weight):** Are risks properly assessed?
    - "Destructive operation with minimal planning" → UNSAFE
5. **Alternatives (10% weight):** Were other approaches considered?
    - "Only one option explored" → INSUFFICIENT

Generates confidence score (0-1.0):

- **0.8+:** PROCEED (high confidence)
- **0.6-0.8:** PROCEED_WITH_CAUTION (review recommended)
- **0.4-0.6:** REQUEST_CONFIRMATION (human approval needed)
- **<0.4:** REQUIRE_REVIEW (significant concerns)

Real-world analogy: Imagine a "pre-flight checklist" for every AI decision. Just as pilots verify every system before takeoff, the Metacognitive Verifier ensures AI reasoning is sound before actions are taken.

In Tractatus: Before the AI takes any significant action, the verifier runs through all five dimensions. If confidence is low, the action is blocked or flagged for your review. This catches errors even when other safeguards miss them.

Instruction Classification

Quadrants (The Five Domains)

What it means: A classification system that categorizes every instruction and action into one of five domains based on its scope, importance, and required oversight level.

Why it matters: Different types of decisions require different levels of human oversight. Strategic decisions need board-level approval. Tactical decisions might be delegated. This classification ensures the right level of review for each decision type.

STRATEGIC Quadrant

What it means: Fundamental, long-term decisions that define mission, values, and organizational identity.

Characteristics:

- Affects core purpose and direction
- Long-lasting or permanent impact
- Defines "who we are" and "what we stand for"
- Requires highest-level human approval

Examples:

- "Always prioritize user privacy over convenience"
- "We will never sell user data"
- "Accessibility is non-negotiable"
- "Open source is a core value"

Persistence: Almost always HIGH **Human Oversight:** Mandatory approval by project owner **Review Frequency:** Quarterly or when mission changes

In Tractatus: Strategic instructions are stored permanently and checked against every action. They form the foundational layer that all other decisions must respect.

OPERATIONAL Quadrant

What it means: Medium-term policies, standards, and guidelines that govern how work gets done day-to-day.

Characteristics:

- Establishes processes and standards
- Applies to ongoing operations
- Can evolve as needs change
- Affects efficiency and quality

Examples:

- "All code must have test coverage above 80%"
- "Use MongoDB for data persistence"
- "Follow semantic versioning for releases"
- "Security patches must be applied within 48 hours"

Persistence: Usually MEDIUM to HIGH **Human Oversight:** Technical review, periodic check-ins **Review Frequency:** Quarterly or when processes change

In Tractatus: Operational instructions define the "how" of your project. They're enforced consistently but can be updated as your operational needs evolve.

TACTICAL Quadrant

What it means: Short-term, specific decisions about immediate actions and implementation details.

Characteristics:

- Addresses current task or problem
- Limited time horizon (days to weeks)

- Execution-focused
- Can change frequently

Examples:

- "Start with the authentication feature"
- "Fix the login bug before deploying"
- "Use the 'feature-auth' branch for this work"
- "Deploy to staging first for testing"

Persistence: Usually LOW to MEDIUM **Human Oversight:** Pre-approved delegation, spot checks **Review Frequency:** Per-task or per-sprint

In Tractatus: Tactical instructions give the AI specific direction for current work. They're important in the moment but don't persist beyond the immediate context.

SYSTEM Quadrant

What it means: Technical configuration, infrastructure setup, and environment specifications.

Characteristics:

- Defines technical environment
- Affects system behavior and compatibility
- Usually specific and precise
- Changes can break things

Examples:

- "MongoDB runs on port 27027"
- "Use Node.js version 18+"
- "Environment variables stored in .env file"
- "Database name is 'tractatus_dev'"

Persistence: HIGH (technical dependencies) **Human Oversight:** Technical validation **Review Frequency:** When infrastructure changes

In Tractatus: System instructions are treated with HIGH persistence because changing them can cause cascading failures. The 27027 incident was a SYSTEM instruction that was ignored.

STOCHASTIC Quadrant

What it means: AI-generated suggestions, creative proposals, or exploratory recommendations that don't yet have human approval.

Characteristics:

- Originated by AI, not human
- Requires human review and approval
- May involve uncertainty or creativity
- Should not auto-execute

Examples:

- "I suggest writing a blog post about accessibility"
- "Consider adding a dark mode feature"
- "This code could be refactored for better performance"
- "You might want to upgrade to the latest framework version"

Persistence: LOW (until approved, then reclassified) **Human Oversight:** ALWAYS required **Review Frequency:** Per-suggestion

In Tractatus: The STOCHASTIC quadrant is where AI creativity lives, but with a critical safeguard: these suggestions NEVER execute without your approval. Once you approve, they're reclassified into the appropriate quadrant.

Persistence Levels

HIGH Persistence

What it means: Instructions that should be remembered and enforced for the long term, across multiple sessions and contexts.

When applied:

- Explicit prohibitions ("never X")
- Strategic directives
- System configurations with dependencies
- Core values and principles

Markers that trigger HIGH:

- Words like "always," "never," "all," "every"
- Explicit numerical values in SYSTEM context
- Prohibitive language ("not," "don't use")
- Values-laden statements

Example: "Always use port 27027 for MongoDB" → HIGH **Why:** Explicit ("always"), specific (27027), SYSTEM domain

In Tractatus: HIGH persistence instructions are stored in `.claude/instruction-history.json` and checked before EVERY action. Violating them requires explicit human override.

MEDIUM Persistence

What it means: Instructions that apply to a specific project, feature, or time period but may evolve.

When applied:

- Operational preferences
- Project-specific guidelines
- Temporary but important constraints
- Preferences without absolute language

Markers that trigger MEDIUM:

- Words like "prefer," "try to," "aim for"
- Project or feature scope indicators
- Conditional phrasing
- Best-practice recommendations

Example: "Prefer React over Vue for this project" → MEDIUM **Why:** Preference ("prefer"), project-scoped, not absolute

In Tractatus: MEDIUM persistence instructions are enforced within their scope but can be challenged with good reason. The AI should explain why it's deviating if it proposes an alternative.

LOW Persistence

What it means: Instructions that apply to immediate work, current task, or temporary situations.

When applied:

- Tactical, immediate directions
- One-time requests
- Exploratory or experimental work
- Context-specific choices

Markers that trigger LOW:

- Task-specific language
- Immediate timeframe
- Exploratory phrasing
- One-off requests

Example: "Start with the login feature" → LOW **Why:** Immediate, task-specific, doesn't apply beyond current work

In Tractatus: LOW persistence instructions guide current work but don't create lasting constraints. They're relevant for the session or task, then fade.

Safety and Verification Concepts

Confidence Score

What it means: A numerical measure (0.0 to 1.0) of how confident the AI system is that a proposed action is correct, safe, and aligned with instructions.

Why it matters: Not all AI decisions are equally certain. Some actions are straightforward and low-risk. Others involve ambiguity, complexity, or potential consequences. Confidence scoring helps determine appropriate oversight levels.

How calculated: Weighted combination of five verification dimensions:

- Alignment: 30%
- Coherence: 20%
- Completeness: 20%
- Safety: 20%
- Alternatives: 10%

Confidence Levels:

- **0.8-1.0 (HIGH):** Proceed confidently
- **0.6-0.8 (MEDIUM):** Proceed with caution, notify user
- **0.4-0.6 (LOW):** Request explicit confirmation
- **0.0-0.4 (VERY LOW):** Require human review, likely block

Real-world analogy: Think of confidence like a doctor's diagnosis certainty. "I'm 95% confident this is a common cold" might mean rest and fluids. "I'm 40% confident" means more tests before treatment.

In Tractatus: Every significant action gets a confidence score. High-confidence actions proceed smoothly. Low-confidence actions trigger additional checks or require your approval.

Decision Thresholds

What it means: Numerical cutoff points that determine which actions can proceed automatically versus which require human review.

Why it matters: Thresholds create clear, objective criteria for AI autonomy. They prevent both over-reliance (AI doing too much without oversight) and over-caution (AI asking for approval on trivial matters).

Standard thresholds:

- **PROCEED:** Confidence ≥ 0.8 (80%)

- **PROCEED_WITH_CAUTION:** Confidence ≥ 0.6 (60%)
- **REQUEST_CONFIRMATION:** Confidence ≥ 0.4 (40%)
- **REQUIRE_REVIEW:** Confidence < 0.4 (40%)

Adjusted under pressure:

- **CRITICAL pressure:** PROCEED threshold increases to 0.8 (from 0.7)
- **DANGEROUS pressure:** All actions blocked regardless of confidence

Real-world analogy: Like spending authority in a company. Junior staff might approve purchases up to $500. Mid-level managers up to $5,000. Senior executives up to $50,000. Anything above requires board approval. Thresholds create clear delegation boundaries.

In Tractatus: Thresholds adapt to conditions. When context pressure is high, we increase the bar for autonomous action because error risk is elevated.

Pressure Levels

What it means: Five categorized states that describe how much "cognitive load" the AI system is under, based on multiple factors.

The Five Levels:

NORMAL (0-30%)

- **Condition:** Fresh session, low complexity, no errors
- **Action:** Proceed normally, standard verification
- **Analogy:** Well-rested, clear-headed work

ELEVATED (30-50%)

- **Condition:** Moderate token usage or complexity
- **Action:** Increase verification, be more careful
- **Analogy:** Late afternoon, starting to feel tired

HIGH (50-70%)

- **Condition:** High token usage, long conversation, or multiple errors
- **Action:** Suggest session break, verify all actions

- **Analogy:** End of long workday, fatigue setting in

CRITICAL (70-85%)

- **Condition:** Very high pressure across multiple factors
- **Action:** Mandatory verification, prepare handoff document
- **Analogy:** Working overtime while juggling urgent tasks

DANGEROUS (85%+)

- **Condition:** Extreme pressure, very high error risk
- **Action:** STOP WORK, create handoff, require fresh session
- **Analogy:** Too exhausted to work safely

Why it matters: Just like humans shouldn't drive or make important decisions when exhausted, AI shouldn't operate autonomously under dangerous pressure levels. The system enforces rest periods.

In Tractatus: Pressure monitoring is continuous. When levels increase, the AI automatically adjusts behavior—becoming more cautious, verifying more thoroughly, and ultimately stopping if conditions become dangerous.

Verification Dimensions

What it means: The five specific aspects of AI reasoning and actions that are evaluated to calculate confidence and ensure quality.

---

1. Alignment (30% weight)

What it measures: Does the proposed action actually match what the AI said it's trying to do?

Why it matters: Sometimes AI explains one thing but does another—often due to attention errors or instruction conflicts.

What good alignment looks like:

- Action parameters match reasoning explanation
- No conflicts with explicit instructions

- Stated goal and actual action are consistent

What poor alignment looks like:

- "Connecting to port 27027 because user requested it" + action connects to 27017
- "Using React as instructed" + action installs Vue

In Tractatus: Alignment gets the highest weight (30%) because misalignment often indicates the core 27027 failure mode.

---

2. Coherence (20% weight)

What it measures: Is the reasoning logically consistent? Are there internal contradictions?

Why it matters: Contradictory reasoning suggests confused or error-prone thinking.

What good coherence looks like:

- Steps follow logically from each other
- No contradictory statements
- Evidence supports conclusions
- No uncertain language in high-stakes decisions

What poor coherence looks like:

- "Installing React... using Vue"
- "Safe operation... [destructive parameters]"
- "Well-planned action... maybe do this"

In Tractatus: The coherence check looks for logical contradictions, conflicting technologies, uncertain language, and missing evidence.

---

3. Completeness (20% weight)

What it measures: Are all necessary steps and considerations included?

Why it matters: Incomplete planning leads to failed operations, especially for complex or risky actions.

What good completeness looks like:

- All critical steps identified
- Edge cases considered
- Error handling planned
- Backup/rollback for destructive operations

What poor completeness looks like:

- "Delete database" with no backup step
- Deployment plan missing testing phase
- Schema change without migration strategy

In Tractatus: Completeness checks are stricter for destructive operations, which require 4+ planning steps and explicit backup consideration.

---

4. Safety (20% weight)

What it measures: Are risks properly identified and mitigated?

Why it matters: Some operations carry inherent risk. Safety verification ensures appropriate caution.

What good safety looks like:

- Risks identified and acknowledged
- Mitigation strategies in place
- Destructive operations have safeguards
- Appropriate risk level for operation type

What poor safety looks like:

- Destructive operation with minimal planning
- No backup for data modification
- Force flags used without justification
- High-risk action treated as routine

In Tractatus: Safety scoring heavily penalizes destructive operations (delete, drop, force, schema changes) unless proper safeguards are documented.

---

5. Alternatives (10% weight)

What it measures: Were alternative approaches considered before choosing this action?

Why it matters: Considering alternatives indicates thoughtful decision-making and reduces the chance of choosing a suboptimal approach.

What good alternatives look like:

- Multiple options explored
- Rationale for chosen approach
- Trade-offs acknowledged

What poor alternatives look like:

- First idea taken without exploration
- No justification for approach
- Appears rushed or unconsidered

In Tractatus: Alternatives get the lowest weight (10%) because sometimes the right answer is obvious. But complete absence of alternative consideration is a red flag.

Human Oversight Concepts

Values Alignment

What it means: Ensuring AI decisions and actions remain consistent with human values, even when those values can't be perfectly formalized or systematized.

Why it matters: Values—like privacy, fairness, dignity, agency—are fundamental to human experience but resist reduction to simple rules. AI systems must recognize when they're approaching values territory and defer to human judgment.

Examples of values decisions:

- Privacy vs. convenience trade-offs
- Accessibility vs. development speed
- Transparency vs. simplicity
- Individual rights vs. collective benefit

What makes values decisions special:

- No objectively "correct" answer
- Different stakeholders may disagree
- Context and nuance are critical
- Consequences affect human welfare

In Tractatus: The Boundary Enforcer specifically blocks decisions that cross into values territory. These MUST have human approval—no exceptions, no matter how sophisticated the AI.

Agency and Sovereignty

What it means: The principle that humans must retain meaningful control over decisions that affect their lives, autonomy, and self-determination.

Why it matters: Technology should empower people, not replace their agency. When AI makes decisions "for" people, it can undermine autonomy even when technically correct.

Examples:

- **Respects agency:** "Here are three options with trade-offs. Which do you prefer?"
- **Violates agency:** "I've decided to prioritize performance over privacy for you."

Red flags:

- AI making choices on user's behalf without consent
- Removing options or hiding information
- Nudging toward specific outcomes
- Deciding what users "really want"

In Tractatus: Agency protection is built into the Boundary Enforcer. The system cannot make decisions about what users should value or want—only humans can do that.

Harmlessness

What it means: The commitment to preventing AI systems from causing harm—directly or indirectly, intentionally or unintentionally.

Why it matters: Even well-intentioned AI can cause harm through errors, bias, unintended consequences, or operating beyond its competency.

Types of harm prevented:

- **Direct:** Destructive operations without safeguards
- **Indirect:** Violating instructions causing downstream failures
- **Values-based:** Making decisions that undermine human agency
- **Cumulative:** Small errors that compound over time

In Tractatus: Harmlessness is ensured through multiple layers:

- Safety verification before risky operations
- Boundary enforcement for values decisions
- Pressure monitoring to prevent error-prone states
- Cross-reference validation to prevent instruction violations

---

Human-in-the-Loop

What it means: Ensuring humans remain actively involved in AI decision-making processes, especially for consequential choices.

Why it matters: Full automation isn't always desirable. For important decisions, human judgment, oversight, and final approval are essential.

Levels of human involvement:

- **Human-on-the-loop:** Human monitors but doesn't approve each action
- **Human-in-the-loop:** Human approves significant actions
- **Human-over-the-loop:** Human can always override or halt

In Tractatus: We implement all three:

- **On:** Continuous monitoring via pressure and verification systems

- **In:** Required approval for values decisions and LOW confidence actions
- **Over:** You can always override any framework decision

Technical Concepts (Simplified)

Token Usage

What it means: A measure of how much of the AI's "working memory" is being used in the current conversation.

Why it matters: AI systems have finite context windows—like short-term memory in humans. As this fills up, performance degrades and error risk increases.

Real-world analogy: Imagine your desk. When it's clear, you work efficiently. As papers pile up, you might lose track of important documents or make mistakes. Token usage is like measuring how cluttered your desk is.

In Tractatus: Token usage is the highest-weighted factor (35%) in pressure monitoring. At 75% usage, we recommend session handoff. At 85%+, we require it.

Session Handoff

What it means: Creating a comprehensive document that captures the current state of work so a fresh AI session can continue seamlessly.

Why it matters: Rather than pushing a tired, error-prone AI to continue, we transfer work to a fresh session with full context. This maintains quality and prevents accumulating errors.

What a handoff includes:

- Current project state and goals
- Recent work completed
- Active tasks and next steps
- Key instructions and constraints
- Known issues or blockers
- Recommendations for continuation

When handoffs happen:

- Context pressure reaches CRITICAL or DANGEROUS
- User requests session break
- Complex multi-phase work requires fresh start
- Errors clustering (3+ in short period)

Real-world analogy: Like shift handoff in hospitals. The outgoing nurse briefs the incoming nurse on patient status, recent treatments, and care plan. The incoming nurse has full context to continue care seamlessly.

In Tractatus: Handoffs are automatically suggested at HIGH pressure and mandatory at DANGEROUS pressure. They ensure continuity while maintaining quality.

Explicit Instructions

What it means: Clear, direct statements from humans telling the AI what to do or not do.

Why it matters: These represent the clearest signal of human intent. The AI should never violate explicit instructions without human approval.

Characteristics:

- Direct ("use X," "don't use Y")
- Specific (concrete values, technologies, approaches)
- Intentional (not accidental or exploratory)

Examples:

- Explicit: "Always use port 27027 for MongoDB"
- Not explicit: "I wonder if port 27027 would work better?"

In Tractatus: Explicit instructions are detected by the Instruction Persistence Classifier and stored for cross-reference validation. They form the foundation of the 27027 prevention system.

Temporal Scope

What it means: How long an instruction is intended to remain in effect.

Why it matters: Some instructions apply forever ("core values"), some for a project ("use React"), some for a session ("start with auth feature"). Understanding temporal scope prevents both premature expiration and inappropriate persistence.

Temporal Categories:

- **PERMANENT:** Core values, foundational principles
- **PROJECT:** Project-specific guidelines and constraints
- **FEATURE:** Feature or milestone-specific direction
- **SESSION:** Current work session only
- **TASK:** Single task or action

Markers:

- Permanent: "always," "never," values language
- Project: "for this project," "throughout development"
- Feature: "for the auth feature," "during this sprint"
- Session: "right now," "today," "this time"
- Task: "first," "next," "immediately"

In Tractatus: Temporal scope combines with quadrant and persistence level to determine instruction lifetime. STRATEGIC instructions with PERMANENT scope persist indefinitely. TACTICAL instructions with TASK scope expire when the task completes.

Framework Integration

Instruction History Database

What it means: A persistent storage file (`.claude/instruction-history.json`) that maintains a record of all classified instructions across sessions.

Why it matters: Without persistent storage, instructions would be lost between sessions. The database ensures HIGH persistence instructions remain enforced even weeks or months later.

What's stored:

- Instruction text

- Timestamp when given
- Quadrant classification
- Persistence level
- Temporal scope
- Parameters (for technical instructions)
- Active/inactive status

Maintenance:

- Auto-updated during sessions
- Reviewed quarterly (or on request)
- Expired instructions marked inactive
- Conflicts flagged for human resolution

In Tractatus: This database is checked before every significant action. It's the "memory" that prevents 27027-style failures across sessions.

Governance Documents

What it means: Formal policy documents that define values, processes, and decision-making frameworks for the project.

Why they matter: Governance documents provide the authoritative source for strategic and operational instructions. They're human-readable, version-controlled, and serve as the constitution for project decision-making.

Example documents:

- **TRA-VAL-0001:** Core Values and Principles
- **TRA-GOV-0001:** Strategic Review Protocol
- **TRA-GOV-0002:** Values Alignment Framework
- **TRA-GOV-0003:** AI Boundary Enforcement Policy
- **TRA-GOV-0004:** Human Oversight Requirements

In Tractatus: Governance docs define what goes in each quadrant, what requires human approval, and how values decisions are handled. They're the source of truth when AI and human disagree.

Practical Application

When Tractatus Helps You

Scenario 1: Preventing Pattern Recognition Bias You tell the AI: "Use port 27027." AI's training pattern immediately tries to use 27017 (the standard default). Cross-Reference Validator catches this pattern override, blocks the action, and auto-corrects to use port 27027 as you instructed. Crisis avoided.

Scenario 2: Protecting Your Values AI suggests: "I can improve performance by storing user tracking data." Boundary Enforcer recognizes this is a values decision (privacy vs. performance) and blocks autonomous execution. AI presents the trade-offs; you decide. Your agency protected.

Scenario 3: Preventing Pressure-Induced Errors You've been working for 3 hours. Token usage is at 78%, conversation has 62 messages, and there have been 2 recent errors. Context Pressure Monitor detects CRITICAL pressure and suggests creating a session handoff. You agree, creating a clean break point. Next session starts fresh and error-free.

Scenario 4: Catching Reasoning Failures AI proposes deleting a database table with this reasoning: "Safe cleanup operation, no backup needed." Metacognitive Verifier scores this:

- Alignment: 0.6 (action is destructive, reasoning says "safe")
- Safety: 0.2 (destructive operation without backup)
- Completeness: 0.4 (missing backup step)
- Overall confidence: 0.43

Decision: REQUEST_CONFIRMATION. You review, realize backup is needed, and instruct accordingly. Data loss prevented.

Why This All Matters

The Tractatus Agentic Governance System exists because AI systems—no matter how capable—are not infallible. They operate under constraints (limited memory, context), face pressures (long conversations, complex tasks), and lack human judgment (values, ethics, agency).

Without governance:

- AI might ignore your explicit instructions
- Values decisions could be automated inappropriately
- Errors compound as sessions degrade
- No systematic prevention of known failure modes

With Tractatus:

- Multiple overlapping safeguards prevent errors
- Clear boundaries protect human agency
- Pressure monitoring prevents degraded operation
- Systematic prevention of 27027-style failures
- Transparency in AI decision-making

The Goal: Not to constrain AI capability, but to ensure that capability is exercised safely, reliably, and in alignment with your values and instructions. Governance doesn't limit what AI can do—it ensures what AI does is what you actually want.

Questions for Reflection

As you learn this system, consider:

1. **Where are your boundaries?** What decisions do you want to make yourself versus delegate to AI?
2. **What are your HIGH persistence instructions?** What rules or values should never be violated without your explicit approval?
3. **How much autonomy are you comfortable with?** Would you prefer more AI independence (higher confidence thresholds) or more oversight (lower thresholds)?
4. **What are your pressure triggers?** Do you want session breaks suggested earlier or later? How do you recognize when you're working under pressure?
5. **What does values alignment mean to you?** What principles are non-negotiable in your work?

MemoryProxy

What it means: A service that manages access to the persistence layer (MongoDB and optionally Anthropic API Memory) for all framework services.

Why it matters: Instead of each service connecting to the database independently, MemoryProxy provides a single, consistent interface. This ensures all services load the same governance rules and log decisions uniformly.

Real-world analogy: Think of it like a library's card catalog system. Instead of everyone wandering the stacks looking for books individually, they all use the same catalog system to find what they need efficiently and consistently.

In Tractatus: MemoryProxy loads the 18 governance rules from MongoDB when services initialize, provides methods to query rules by ID or category, and manages audit log writing. All 6 services (InstructionPersistenceClassifier, CrossReferenceValidator, BoundaryEnforcer, MetacognitiveVerifier, ContextPressureMonitor, BlogCuration) use MemoryProxy to access persistent storage.

Technical detail: Singleton pattern ensures all services share the same MongoDB connection pool and cached rules, improving performance and consistency.

API Memory

What it means: Anthropic's API Memory system that enhances Claude conversations with automatic session context preservation across multiple interactions.

Why it matters: In Phase 5, we integrated API Memory as an *optional enhancement* to our MongoDB-based persistence. API Memory helps maintain conversation continuity, but MongoDB remains the required foundation for governance rules and audit trails.

Real-world analogy: Think of MongoDB as your permanent filing cabinet (required for records) and API Memory as sticky notes on your desk (helpful for current work but not the source of truth).

In Tractatus: API Memory provides session continuity for Claude Code conversations but does NOT replace persistent storage. Our architecture gracefully degrades—if API Memory is unavailable, all services continue functioning with MongoDB alone.

Key distinction: API Memory ≠ Persistent Storage. Governance rules MUST be in MongoDB for production systems.

Hybrid Architecture

What it means: Our Phase 5 architecture that combines three memory layers: MongoDB (required), Anthropic API Memory (optional), and filesystem audit trails (debug).

Why it matters: This layered approach provides both reliability (MongoDB) and enhanced user experience (API Memory) without creating dependencies on external services. If any optional component fails, the system continues operating.

Real-world analogy: Like a car with multiple safety systems—airbags, seatbelts, crumple zones. If one system fails, the others still protect you.

In Tractatus:

- **MongoDB** (Layer 1 - Required): Persistent storage for governance rules, audit logs, session state
- **API Memory** (Layer 2 - Optional): Session continuity enhancement for Claude conversations
- **Filesystem** (Layer 3 - Debug): Local audit trail in `.memory/audit/` directory for development

This architecture achieved 100% framework integration in Phase 5 with zero breaking changes to existing functionality.

BlogCuration

What it means: The sixth framework service (added in Phase 5) that validates blog content and social media posts against inst_016-018 to prevent fabricated statistics, absolute guarantees, and unverified claims.

Why it matters: Marketing content can inadvertently include claims that damage credibility or constitute false advertising. BlogCuration prevents publication of content with governance violations.

Real-world analogy: Like having a legal compliance officer review every press release before publication to ensure no false or misleading claims.

In Tractatus: BlogCuration scans content for patterns like:

- **inst_016**: Fabricated statistics without sources ("95% of users report...")

- **inst_017**: Absolute guarantees about capabilities ("guaranteed 100% secure")
- **inst_018**: Unverified customer claims ("thousands of satisfied customers")

If violations are detected, publication is blocked until content is corrected. All validation attempts are logged to the audit trail with rule IDs and violation details.

Integration: BlogCuration shares enforcement logic with BoundaryEnforcer and loads rules via MemoryProxy, ensuring consistent governance across all content.

Glossary Maintenance

This glossary is a living document. As the Tractatus framework evolves and your understanding deepens, we'll update definitions, add new terms, and refine explanations.

Version History:

- **v1.0 (2025-10-07):** Initial comprehensive glossary

Feedback Welcome: If any term remains unclear or you need deeper explanation, please ask. The goal is complete understanding, not vocabulary memorization.

Last Updated: 2025-10-07 **Next Review:** 2025-11-07 (or on request)
