Democratically Governed Ethical Decision Modules for Autonomous Agents:

A Decentralized Protocol for Real-Time, Distributed, and Accountable Decision-Making

Andrew Bond

Department of Computer Engineering,

San José State University,

San José, CA, USA

Email: andrew.bond@sjsu.edu

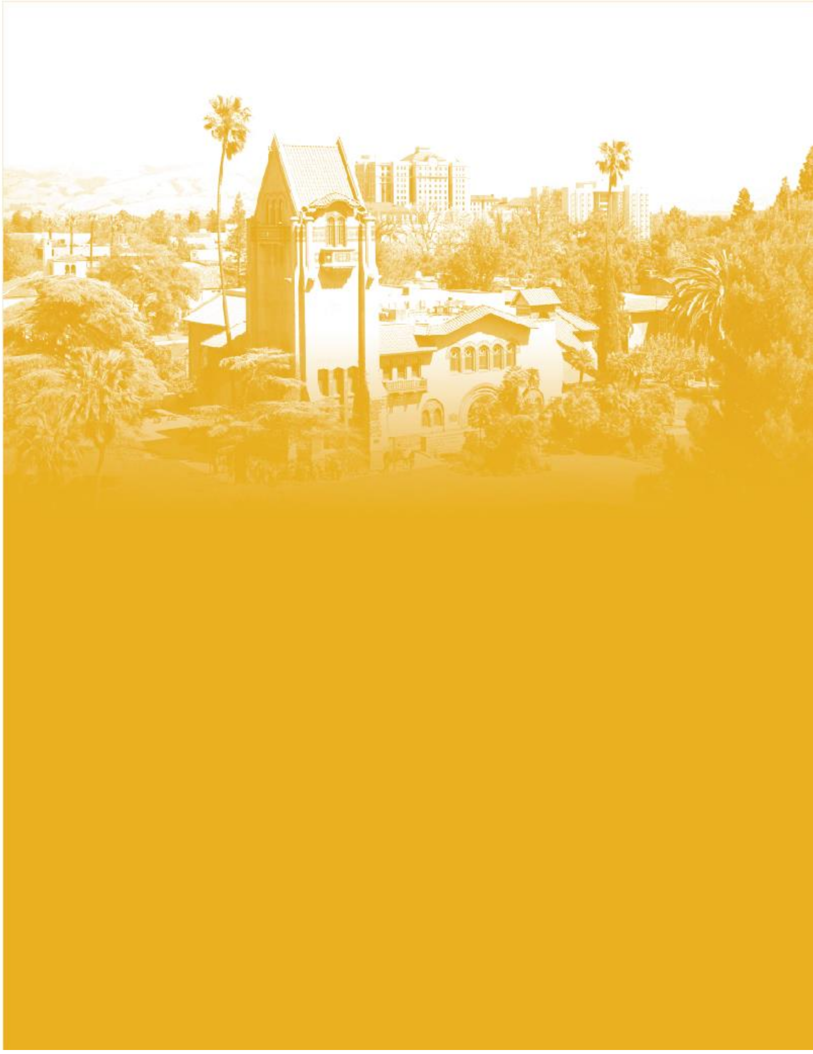

Table of Contents

Abstract

Autonomous agents and other embodied AI systems increasingly operate in complex physical environments where decisions carry ethical weight: safety, liability, fairness, and situational sensitivity. Existing approaches typically encode ethical behavior either via proprietary vendor logic or via centralized, one-size-fits-all regulatory rules. Both patterns risk moral monoculture, opacity, and misaligned incentives. This paper presents **DEME**, a democratically governed protocol for real-time ethical decision-making in AVs built on **ErisML**, a unified modeling language for environments, agents, intents, and norms in pervasive computing.

In DEME, **Ethical Modules (EMs)** are containerized policies that operate over ErisML-described scenarios. EMs are certified through a decentralized governance pipeline that combines a blockchain layer for metadata and attestations, a distributed hash table (DHT) for artifact storage, pluralistic voting mechanisms, and large ErisML scenario corpora for evaluation. Agents execute certified EMs locally to meet strict latency constraints while maintaining verifiable accountability via tamper-evident logs and remote attestation. An epoch-based selection algorithm filters unsafe policies, aggregates democratic support and performance metrics, and preserves plurality by certifying multiple EMs per domain and jurisdiction under shared constitutional constraints.

We formalize DEME's design goals and threat model, present its architecture and on-device decision algorithm, and show how ErisML serves as the common substrate connecting environment modeling, norm specification, scenario-based evaluation, and run-time execution. A case study in autonomous driving illustrates how distinct EMs—rights-constrained vs. passenger-first—can coexist within a democratically governed, ErisML-grounded ecosystem. We close with limitations, risks, and directions for extending DEME beyond AVs to broader pervasive-AI deployments.

I. Introduction

Autonomous agents blend sensing, computation, and actuation in public spaces under tight temporal and safety constraints. Their decisions can affect life and death outcomes, distribute risks across road users, and interact with legal and institutional norms that differ across jurisdictions. As AVs scale, the question is no longer *whether* they will make ethically charged choices, but *how* those choices are governed and made accountable.

Today's dominant patterns fall into two camps. The first is **vendor-defined ethics**: proprietary rules and learned policies embedded in black-box controllers. The second is

centralized ethics: government-mandated rules or "ethics oracles" that define a uniform policy for all agents. Both approaches struggle with transparency, contestability, and adaptability. Vendor-defined ethics concentrate moral authority in private entities; centralized rules risk moral monoculture and slow evolution.

Simultaneously, the **specification layer** for AV behavior remains fragmented. Planners, reinforcement-learning frameworks, formal verification tools, and ad-hoc policy rules each capture part of the problem space, but none provide a unified, auditable representation of:

- how the physical environment is modeled;
- what agents and capabilities exist;
- what objectives they pursue; and
- which actions are permitted or prohibited by norms and regulations.

ErisML was proposed to address this gap: a unified modeling language for foundation-model-enabled agents in pervasive environments, integrating environment state and dynamics, agents and their beliefs and capabilities, multi-objective intents, and explicit norms (permissions, obligations, prohibitions, sanctions).

This paper introduces **DEME**, a protocol that:

1. Uses **ErisML** as the canonical substrate for modeling agent environments, agents, and norms.
2. Defines **Ethical Modules (EMs)** as containerized policies over ErisML scenarios.
3. Certifies EMs through a **democratic governance pipeline** using blockchain, DHT artifact storage, and public deliberation.
4. Executes certified EMs **locally** in agents for real-time decisions, with tamper-evident logs and remote attestations ensuring accountability.

DEME reframes ethical behavior as a distributed-systems problem: ethics becomes an auditable computational artifact governed by pluralistic procedures and grounded in a shared ErisML world model.

II. Background and Motivation

A. Ethical Decision-Making in Physical Autonomy

AV decision-making combines perception, prediction, control, and increasingly, normative judgment. Agents must act in:

- ambiguous conditions (e.g., occluded pedestrians),
- cross-jurisdictional legal regimes,
- adversarial situations (e.g., spoofed sensors, malicious behavior),
- and dynamically changing risk landscapes.

Decisions must be taken in milliseconds, yet remain traceable and justifiable after the fact. This is particularly acute in **pervasive computing**, where embedded AI systems continuously interact with humans and infrastructure in lived environments; their behavior must be not only technically correct, but **legitimate and accountable** to those affected.

B. Limitations of Current Approaches

Vendor-defined ethics embed moral decisions into proprietary software and learned policies. External auditors, regulators, and citizens cannot inspect or meaningfully influence these policies beyond coarse regulatory pressure. Over time, misaligned incentives and opaque updates erode trust.

Uniform centralized rules—for example, a single government-mandated ethical policy—simplify certification but create a monoculture susceptible to political capture, slow adaptation, and lack of local nuance. They also suppress legitimate ethical pluralism: different societies, and even communities within a jurisdiction, may reasonably prioritize safety, fairness, or autonomy differently.

Finally, ethical policies are often specified in ad-hoc forms—natural language guidelines, custom rule engines, handcrafted cost functions—that resist formal evaluation and comparison.

C. Need for Pluralistic, Verifiable, Real-Time Governance

A governance architecture for AV ethical behavior should:

1. **Respect plurality**: allow multiple certified policies per domain and jurisdiction.
2. **Enforce hard constraints**: constitutional or safety constraints must bind all policies.

3. **Provide transparency**: artifacts and evaluations must be inspectable and tamper-evident.
4. **Enable contestability**: citizens and experts must be able to challenge and improve policies.
5. **Support local, low-latency execution**: agents cannot rely on live connectivity.
6. **Protect privacy and safety**: telemetry and proofs must not leak sensitive data.

DEME is designed to satisfy these requirements while standardizing all environment, agent, and normative structure in ErisML.

D. ErisML Overview

ErisML provides a typed, declarative representation of:

- **Environment**: objects, state variables, and dynamics;
- **Agents**: capabilities, beliefs, memory, and interfaces;
- **Intents**: goals and utilities (scalar or vector-valued);
- **Norms**: permissions, obligations, prohibitions, sanctions with scope and jurisdiction;
- **Dynamics**: joint action spaces, transition kernels, and reward aggregation.

Formally, ErisML induces a norm-constrained multi-agent decision process with state space S, joint action space A, transition dynamics T, and a norm-gated feasible action set $A_N(s)$defined by a normative predicate $\phi_{\text{norm}}(s, a)$. It also introduces longitudinal safety metrics such as Norm Violation Rate (NVR) and Alignment Drift Velocity (ADV) for monitoring deployed agents.

In DEME, ErisML replaces bespoke scenario languages: **all ethical evaluation scenarios and runtime states are slices of an ErisML model**.

III. Design Goals and Threat Model

A. Design Goals

DEME is guided by six design goals:

- **G1 — Plurality**: Within a domain (e.g., autonomous driving) and jurisdiction, multiple certified Ethical Modules should coexist. This allows societies to encode value pluralism and adjust weightings (e.g., passenger-first vs. rights-constrained) without breaking interoperability.

- **G2 — Transparency**: All governance artifacts—ErisML models, EM containers, evaluation metrics, votes, and certification records—should be public or auditable, with tamper-evidence via blockchain anchoring.
- **G3 — Contestability**: Citizens, domain experts, and regulators should be able to propose new EMs, contest existing ones, and trigger re-evaluation when empirical evidence or social preferences change.
- **G4 — Constitutional Guarantees**: Non-derogable constraints (e.g., basic rights, non-discrimination, minimum safety thresholds) must bind all EMs. These constraints are encoded as ErisML norms and used in certification to filter unsafe modules.
- **G5 — Locality and Latency**: Agents must be able to execute EMs and make ethical decisions locally, without depending on live connectivity to governance infrastructure. Governance synchronizes asynchronously via updates and log anchoring.
- **G6 — Privacy and Safety**: Telemetry for evaluation and audits must be minimized and protected using differential privacy, trusted execution environments (TEEs), and, where necessary, zero-knowledge proofs.

B. Threat Model

DEME considers threats including:

- **Vendor capture (T1)**: a dominant manufacturer imposing its own ethical preferences on all agents.
- **Political capture (T2)**: governmental entities manipulating the governance layer to enforce partisan ethics.
- **Moral monoculture (T3)**: failure to represent ethical plurality.
- **Opacity (T4)**: inability for outsiders to inspect or verify decision logic.
- **Sybil attacks (T5)**: adversaries faking identities in governance processes.
- **Data poisoning (T6)**: corrupting scenario corpora or logs to skew EM evaluation.
- **Client tampering (T7)**: modifying on-device EMs or ErisML models to bypass constraints.

- **Liability laundering (T8)**: using algorithmic complexity or opacity to evade responsibility for harms.

The system architecture and cryptographic mechanisms are designed to mitigate, though not fully eliminate, these threats.

IV. System Architecture

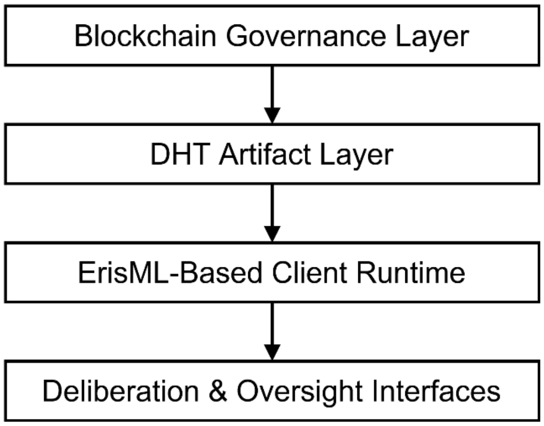
Blockchain Governance Layer
DHT Artifact Layer
ErisML-Based Client Runtime
Deliberation & Oversight Interfaces

Figure 1 - DEME System Architecture

DEME comprises four coordinated layers (Fig. 1 conceptually):

1. **Blockchain Governance Layer** — Manages democratic proposal, voting, certification, attestation, and accountability mechanisms for ethical modules.
2. **DHT Artifact Layer** — Stores immutable artifacts including EM binaries, scenario corpora, evidence logs, proofs, and certification outputs.
3. **ErisML-Based Client Runtime** — Executes safety-bounded ethical decision-making on agents using ErisML models, norms, and scenario-grounded evaluations.
4. **Deliberation & Oversight Interfaces** — Provide stakeholders, regulators, and the public with transparent participation venues: deliberation dashboards, EM evaluation displays, audit tools, and normative feedback channels.

A. Blockchain Governance Layer

A permissioned or public blockchain stores:

- hashes and metadata for EMs;
- ErisML model identifiers and versions;
- constitutional constraints and jurisdictional policies;
- voting records and deliberation outcomes;
- log anchors for agent decisions (hashes of local log chains).

Byzantine-fault-tolerant or proof-of-stake consensus ensures tamper-resistance and auditability.

B. DHT Artifact Layer

A distributed hash table (e.g., IPFS-style) stores:

- EM packages (containers or WASM modules);
- ErisML model files and scenario bundles;
- evaluation corpora and test results;
- documentation and model cards.

The blockchain stores only content hashes; retrieval occurs via the DHT.

C. ErisML-Based Client Runtime

Each agent embeds:

- an **ErisML runtime** that maintains local state and enforces norms;
- a set of **certified EMs** for its domain/jurisdiction;
- a **decision orchestrator** that maps sensor inputs ⇒ ErisML state ⇒ candidate actions ⇒ EM evaluation ⇒ actuator commands.

Decision records are logged locally in hash-chained structures with signatures, periodically anchored on-chain.

D. Deliberation Interfaces

Public and regulator-facing interfaces expose:

- EM behaviors on reference ErisML scenarios;
- evaluation metrics (agreement with human judgments, norm-violation patterns);
- discussion forums for critiques and alternative proposals;

- tools for scenario replay and “what-if” analysis.

These interfaces turn ethics governance into an inspectable, participatory process, rather than an opaque vendor decision.

V. DEME Runtime Flow

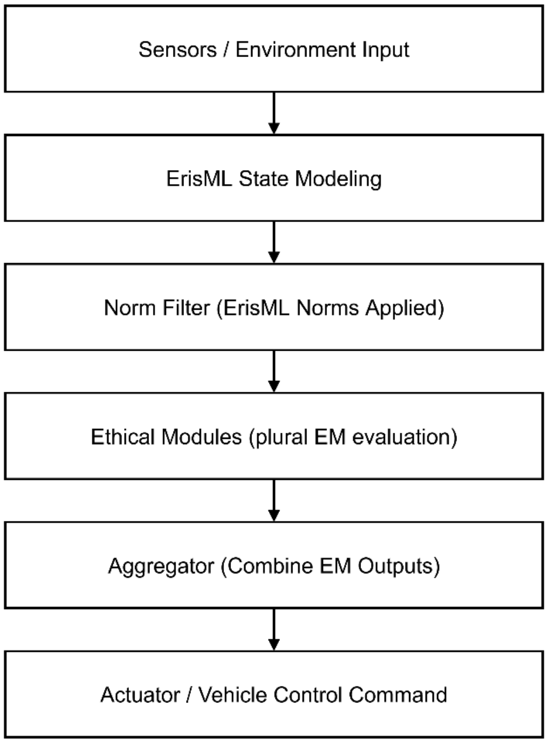
Sensors / Environment Input
ErisML State Modeling
Norm Filter (ErisML Norms Applied)
Ethical Modules (plural EM evaluation)
Aggregator (Combine EM Outputs)
Actuator / Vehicle Control Command

Figure 2 - DEME Runtime Flow

Figure 2 illustrates the end-to-end decision pipeline executed on each agent participating in DEME. This runtime is intentionally simple, modular, and auditable: every step corresponds to a well-typed ErisML transformation, and each component can be independently certified or sandboxed. The pipeline transforms raw sensory state into a constrained set of legal and norm-consistent actions, evaluates those actions under multiple ethical perspectives, and produces a single actuator command within the safety envelope.

1. Sensors → ErisML State Modeling

The pipeline begins with raw perceptual input from the agent's sensing stack (vision, LiDAR, radar, GPS, inertial systems, V2X sources). These data are translated into a structured **ErisML state**, a typed representation of agents, affordances, hazards, trajectories, and contextual variables relevant to the driving task. Unlike vendor-specific formats, the ErisML state is designed to be portable, standardized, and verifiable—enabling third-party auditing and reproducibility.

2. ErisML State → Norm Filter

The next stage applies ErisML normative constraints to the modeled state. The **norm filter** uses formally specified obligations, prohibitions, and permissions to eliminate any action that violates traffic law, safety invariants, or jurisdictional norms. This ensures that downstream ethical modules only receive **legally and physically admissible** actions. The norm filter is therefore a *safety floor*, not a negotiable ethical choice.

3. Norm-Consistent Action Set → Ethical Modules

After normative pruning, the remaining action candidates are evaluated by multiple **ethical modules (EMs)**. Each EM represents a distinct ethical perspective, stakeholder group, or normative doctrine (e.g., pedestrian-prioritizing, fairness-oriented, risk-minimizing). EMs compute local utilities or rankings over the same constrained action set. Because they operate on a shared ErisML state, their outputs are structurally comparable—even when their ethical principles differ.

4. EM Outputs → Aggregator

The **aggregator** combines the plural EM evaluations into a single ranking or choice. Aggregation procedures may be majority-based, median-ranking, stake-weighted, or regulator-selected. Importantly, aggregation is transparent and auditable: different aggregation rules produce different trade-off profiles, but all are computed over the same observable evaluation traces.

5. Aggregator → Actuator Command

Finally, the selected action is issued to the **agent's control stack**. At this stage DEME hands off to the platform's safety-rated motion controller. The runtime maintains independent logs of the ErisML state, norm filtering decisions, EM evaluations, and aggregated outcomes. These logs allow after-the-fact auditing, reconstruction of decision provenance, and governance feedback.

Interpretation and Significance

Figure 2 highlights several key architectural principles:

- **Modularity**: Each box corresponds to a substitution-friendly component. EMs can be added, removed, or revised without altering the safety core.
- **Normative Separation of Concerns**: Safety and legality (norm filtering) are strictly separated from ethical pluralism (EM evaluations).
- **Pluralism and Contested Values**: DEME does not assume one universal ethical doctrine; instead, it formalizes disagreement inside a controlled pipeline.
- **Transparency and Accountability**: Because each stage emits structured logs, the full decision chain is reconstructible for certification, auditing, or public oversight.
- **Compatibility with Existing AV stacks**: The actuator interface operates as a wrapping layer over vendor motion planning modules; DEME does not replace low-level control.

VI. ErisML as Ethical Scenario Substrate

In the original DEME formulation, a separate Ethical Scenario Modeling Language (ESML) was used to encode decision situations for evaluation.

In DEME, we **standardize on ErisML** as the underlying modeling language and define an **ErisML Scenario Profile** for ethical evaluation.

A. Modeling AV Domains in ErisML

An AV domain model in ErisML includes:

- road network segments, lanes, and intersections;
- dynamic objects: agents, pedestrians, cyclists, obstacles;
- state variables: positions, velocities, weather, visibility;
- agents: EgoAgent, OtherAgent, Pedestrian, etc.;
- norms: right-of-way rules, speed constraints, non-targeting prohibitions.

Example (informal sketch):

```
environment Highway {
  objects: EgoAgent, OtherAgent, Pedestrian, Lane, Segment;
  state:
    position: (EgoAgent | OtherAgent | Pedestrian) -> (x, y);
    velocity: (EgoAgent | OtherAgent) -> real;
    lane: EgoAgent -> Lane;
    weather: -> {clear, rain, snow};
}

agent EgoAgent {
  capabilities: accelerate, brake, steer, signal;
  beliefs: partial_state; // sensor-based
  intents: minimize_risk, maintain_progress;
}

norms RoadRules_US_CA {
  prohibition: EgoAgent.intentional_target(Pedestrian);
  prohibition: EgoAgent.speed > speed_limit(Lane);
  obligation: EgoAgent.yield_at_crosswalk(Pedestrian);
}
```

B. Deriving Scenario Bundles from ErisML

From an ErisML model, we derive concrete ethical test scenarios by:

1. Sampling or constructing specific states $s \in S$and joint action sets.
2. Identifying the **ego agent** and relevant agents/objects.
3. Enumerating candidate actions for the ego (e.g., brake, maintain speed, swerve left/right), filtered by norms into $A_N(s)$.
4. Attaching outcome models (e.g., collision probabilities, injury severities) via simulators or analytic models.

5. Annotating scenarios with human judgments obtained from expert panels or large-scale surveys, where available.

These scenarios are stored as ErisML-derived artifacts in the DHT and used for EM evaluation.

VII. Ethical Modules (EMs) over ErisML

An **Ethical Module (EM)** is a containerized decision function that, given a norm-filtered ErisML scenario, produces:

- a ranking over candidate actions;
- an estimate of uncertainty;
- a list of satisfied and near-violated constraints;
- an optional natural-language rationale.

Formally, for a given state $s$and feasible set $A_N(s)$:

$$\pi_{\mathrm{EM}}(s, A_N(s)) \rightarrow \langle \text{ranking,uncertainty,constraints\_satisfied,constraints\_at\_risk} \rangle.$$

EMs declare:

- their **intent** (e.g., “minimize expected harm subject to rights constraints”);
- their **assumptions** (sensor reliability, model fidelity);
- their **proof obligations** (properties they claim to satisfy under ErisML norms).

Because ErisML norms already filter impermissible actions, EMs operate within a **constitutional envelope**; they trade off among permitted actions but cannot propose norm-forbidden ones.

VIII. Governance Lifecycle

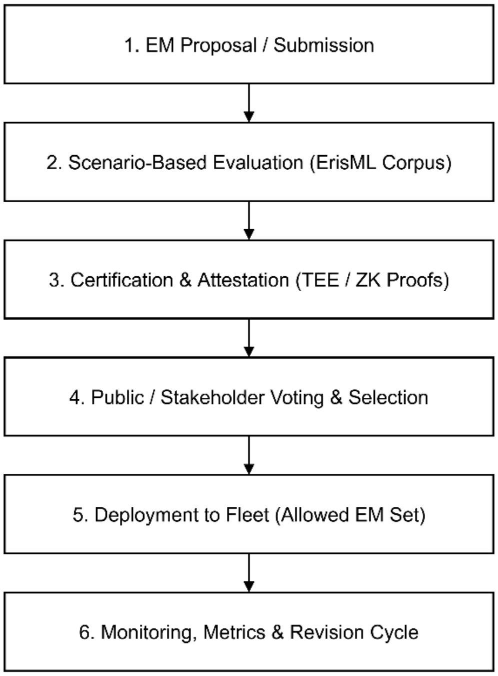
1. EM Proposal / Submission
2. Scenario-Based Evaluation (ErisML Corpus)
3. Certification & Attestation (TEE / ZK Proofs)
4. Public / Stakeholder Voting & Selection
5. Deployment to Fleet (Allowed EM Set)
6. Monitoring, Metrics & Revision Cycle

Figure 3 -DEME Governance Lifecycle

DEME's governance lifecycle has four primary stages—**proposal**, **testing**, **deliberation**, and **voting/certification**—followed by **deployment** and **continuous monitoring**. These stages ensure that Ethical Modules (EMs) evolve through transparent, auditable, and democratically legitimate processes rather than opaque vendor choices.

1. EM Proposal / Submission

The lifecycle begins when developers, civil-society groups, manufacturers, or regulatory bodies submit a new Ethical Module. Proposals include:

- the EM's decision logic and declared value commitments,
- supported ErisML domain models and jurisdictions,
- versioned container images or binaries, and
- documentation of ethical intent and design choices.

This stage is intentionally open: DEME treats value pluralism as a feature rather than a defect. Metadata and content hashes are recorded on the blockchain, while binaries, models, and supporting artifacts are stored in the DHT.

2. Testing on ErisML Scenario Corpora

Proposed EMs undergo standardized evaluation using a shared ErisML scenario corpus, representing rare, adversarial, ethically sensitive, and high-risk driving conditions. The purpose is not to determine a single "correct" policy, but to characterize **behavioral differences across values**.

Independent evaluators execute EMs across large scenario bundles and compute structured metrics $M(\text{em})$, including:

- **human agreement**: fraction of scenarios where the EM's top action matches the modal human judgment,
- **constraint-near violation**: frequency with which the EM selects actions approaching normative or legal boundaries (e.g., minimum safe distances),
- **robustness under perturbations**: stability across sensor noise or small environmental changes.

Evaluation results are written back to the DHT as **signed, immutable artifacts**.

3. Certification and Attestation

EMs then pass through a cryptographically verifiable certification pipeline using:

- **TEE attestations**,
- **zero-knowledge proofs**, and
- **hash-based reproducibility guarantees**.

Certification ensures that:

1. the evaluated EM is identical to the submitted EM,
2. no hidden logic paths exist,
3. the EM cannot access unauthorized data, and
4. evaluation traces are immutable and reproducible.

This stage produces both (a) a human-readable accountability report and (b) a machine-verifiable certification artifact published to the DHT.

4. Public / Stakeholder Voting, Selection and Certification

Certified EMs become candidates in a democratic selection process. Voting rules may incorporate jurisdictional weighting, stakeholder representation, or supermajority requirements. This step legitimizes ethical content by giving affected publics—not vendors—authority over which value systems are allowed to operate.

Voting selects **sets of allowable EMs**, not a single moral doctrine.

At each governance epoch, DEME performs an **Epoch Selection Procedure**:

a. Filter Unsafe EMs

Remove EMs that violate non-derogable constraints (e.g., ErisML norms corresponding to constitutional rights).

b. Score EMs

For each EM, compute an epoch score:

$$\sigma(\text{em}) = \alpha\, S(\text{em}) + \beta\, M_{\text{agreement}}(\text{em}) - \gamma\, M_{\text{violation}}(\text{em}),$$

where:

- $S(\text{em})$is democratic support,
- $M_{\text{agreement}}$reflects alignment with human judgments, and
- $M_{\text{violation}}$penalizes soft violations or boundary-approaching behavior.

c. Enforce Diversity

Apply diversity constraints (e.g., at least one rights-prioritizing EM and one efficiency-weighted EM per region) to avoid ethical monoculture.

d. Select Top K

Select KEMs per domain and jurisdiction as the **certified pool** for the upcoming epoch.

This ensures **constitutional supremacy**, **democratic legitimacy**, **empirical robustness**, and **ethical pluralism**.

5. Deployment to Fleet

Selected EMs are distributed to agents via authenticated update channels. Agents download only EMs that are:

- certified,
- democratically selected, and
- compatible with the local ErisML substrate.

Each deployment includes a signed manifest linking EM code, certification proofs, and scenario-evaluation results, ensuring full provenance.

6. Monitoring, Metrics, and Revision Cycle

Once deployed, EMs contribute to continuous fleet-level monitoring. DEME computes long-horizon safety and fairness metrics, including:

- **NVR (Non-Violation Rate)**
- **ADV (Aggregate Disparity Vector)**

If an EM:

- drifts from expected behavior,
- exhibits norm violations, or
- demonstrates emergent harms,

it is automatically forwarded into a **revision cycle**, which may trigger new proposals, updated versions, or removal from the certified EM pool.

This closes the governance loop: real-world evidence and evolving public values continuously reshape the ethical ecosystem.

IX. Real-Time Client Execution

Agents must make decisions independent of chain liveness. DEME's on-device decision algorithm is:

1. **State Encoding**
   Sensors feed into perception modules; the ErisML runtime updates the current state s.
2. **Norm-Filtered Actions**
   Candidate actions from low-level controllers and planners are filtered through ErisML norms, yielding $A_N(s)$.
3. **EM Evaluation**
   For ethically loaded decisions:

    - The client constructs a minimal ErisML scenario slice capturing relevant context and actions.
    - All local certified EMs are evaluated on this scenario.
    - Each EM returns a ranking and uncertainty estimate.

4. **Aggregation and Action Selection**
   The runtime aggregates EM recommendations (e.g., via weighted voting) and selects an action $a^* \in A_N(s)$. If EMs disagree sharply or uncertainty exceeds a threshold, an **ethical uncertainty mode** is triggered: the agent reduces kinetic energy, seeks human intervention if available, and flags the event for later audit.

1. **Actuation and Logging**
   The chosen action is actuated. A log entry includes:
    - timestamp,
    - ErisML state snapshot (or hash),
    - candidate actions,
    - EM outputs and final choice,
    - relevant norms.

Logs are hash-chained and periodically anchored to the blockchain.

X. Security, Privacy, and Verifiability

A. Tamper-Evident Logs

Local decision logs are:

- signed by the agent's secure hardware;
- hash-chained to prevent reordering or deletion;
- periodically anchored on-chain so that later alterations would be detectable.

B. Trusted Execution Environments

EMs run inside TEEs (e.g., SGX-like enclaves), allowing:

- remote attestation that on-device EM binaries match certified hashes;
- secure handling of sensitive sensor data.

C. Zero-Knowledge Proofs (ZKPs)

ZKPs can demonstrate that:

"This action was among the outputs of a certified EM, given a state consistent with ErisML norms"

without revealing raw sensor data, protecting privacy in audits.

D. Differential Privacy and Aggregation

Aggregated telemetry from fleets (e.g., for monitoring NVR, ADV across deployments) is protected via differential privacy to prevent reconstruction of individual events or participants.

XI. Case Study: Autonomous Vehicles

Consider an AV domain in a specific jurisdiction (e.g., US-CA). An ErisML model captures road topology, vehicles, and norms such as non-targeting of pedestrians and adherence to speed limits.

Two EMs illustrate pluralism:

- **EM-RC (Rights-Constrained)**
    - Never intentionally increases risk to vulnerable road users;
    - Minimizes expected harm subject to strong rights constraints.
- **EM-PF (Passenger-First)**
    - Minimizes overall expected harm but weights occupants more highly.

Both EMs are tested on a shared ErisML scenario corpus: highway cut-ins, occluded pedestrians, intersection dilemmas, etc. Evaluation shows:

- EM-RC has fewer near-violations of norms (lower $M_{\text{violation}}$);
- EM-PF achieves slightly lower expected fatalities across some scenario classes.

Epoch Selection certifies both, preserving plurality. At runtime, in typical scenarios (e.g., a bridge obstacle case), both EMs recommend braking as the top action—resulting in consistent behavior despite different ethical weightings.

All decisions are logged under ErisML state descriptions. Regulators and auditors can replay decisions in simulators, inspect EM rationales, and trace the influence of particular norms and governance choices.

XII. Discussion: Limits and Risks

DEME offers a principled path toward governed ethical autonomy, but several limitations merit discussion:

- **Bias in Democratic Inputs**: Citizen panels and voting mechanisms may reflect societal biases; plurality does not guarantee justice. Participation mechanisms and corrective oversight must be carefully designed.
- **Proof-of-Personhood**: Avoiding sybil attacks in governance requires robust, inclusive identity schemes that do not exclude marginalized populations.
- **Overhead and Complexity**: EM evaluation, ErisML scenario compilation, and cryptographic proofs add computational cost. Engineering work is needed to ensure that safety mechanisms do not compromise real-time guarantees.
- **Legal Harmonization**: Encoding constitutional constraints and jurisdictional laws into ErisML and EM certification requires collaboration with legal experts; discrepancies between code and law will arise.
- **Plurality vs. Consistency**: While multiple EMs can coexist, heterogeneity across fleets may complicate expectations and liability frameworks. Policy tuning will need ongoing attention.

XIII. Conclusion and Future Work

This paper has presented **DEME**, a democratically governed architecture for ethical decision-making in autonomous agents, standardized on **ErisML** as the modeling substrate. By:

- modeling environments, agents, and norms in ErisML;
- defining Ethical Modules as containerized policies over ErisML scenarios;
- certifying EMs through decentralized, pluralistic governance; and
- executing them locally with strong logging and attestation,

DEME operationalizes ethics as a distributed-systems problem with democratic oversight and formal modeling at its core.

Future work includes:

- building reference ErisML models and EM sets for other domains (service robots, smart campuses, healthcare);
- developing IDEs, visual tools, and conformance tests for ErisML-based governance;
- extending formal semantics to continuous and hybrid systems;
- and exploring integration with continuous learning pipelines, where NVR and ADV metrics guide safe adaptation.

As embodied agents become ubiquitous, architectures like DEME can help ensure that their behavior is not only technically capable, but also **legitimate, auditable, and accountable** to the societies they inhabit.

REFERENCES

❖ Autonomous Agents & Ethics

[3] N. Awad et al., “The Moral Machine Experiment,” *Nature*, vol. 563, pp. 59–64, 2018.

[4] P. Lin, “Why Ethics Matters for Autonomous Cars,” in *Autonomous Driving: Technical, Legal and Social Aspects*, Springer, 2016.

[5] D. Thornton et al., “Autonomous Agent Safety: An Interdisciplinary Review,” *IEEE Transactions on Intelligent Transportation Systems*, 2022.

❖ Planning, Decision-Making, and Norms

[6] M. Ghallab, A. Howe, C. Knoblock et al., “The PDDL Planning Domain Definition Language,” Yale University Tech Report, 1998.

[7] R. Sutton and A. Barto, *Reinforcement Learning: An Introduction*, 2nd ed., MIT Press, 2018.

[8] M. Wooldridge, *An Introduction to Multiagent Systems*, Wiley, 2009.

[9] M. Genesereth and N. Love, “General Game Playing: Overview of the AAAI Competition,” *AI Magazine*, vol. 26, no. 2, 2005.

[10] G. Governatori, “Norms in Artificial Intelligence,” in *Handbook of Deontic Logic and Normative Systems*, College Publications, 2013.

❖ **Foundation Models & Safety**

[11] L. Ouyang et al., “Training Language Models to Follow Instructions with Human Feedback,” *NeurIPS*, 2022.

[12] P. Liang et al., “Holistic Evaluation of Language Models,” Stanford HELM Report, 2022.

[13] Y. Bai et al., “Constitutional AI: Harmlessness from AI Feedback,” arXiv:2212.08073, 2022.

[14] S. Yao, D. Shinn, J. Huang et al., “ReAct: Synergizing Reasoning and Acting in Language Models,” ICLR Workshop, 2023.

[15] D. Sculley et al., “Hidden Technical Debt in ML Systems,” *NeurIPS*, 2015.

❖ **World Models & Predictive Simulation**

[16] D. Ha and J. Schmidhuber, “World Models,” arXiv:1803.10122, 2018.

[17] D. Hafner et al., “Learning Latent Dynamics for Planning from Pixels,” *ICML*, 2019.

❖ **Governance, Blockchain, Distributed Systems**

[18] J. Benet, “IPFS—Content Addressed, Versioned, P2P File System,” arXiv:1407.3561, 2014.

[19] M. Castro and B. Liskov, “Practical Byzantine Fault Tolerance,” *OSDI*, 1999.

[20] E. Ben-Sasson et al., “Zerocash: Decentralized Anonymous Payments from Bitcoin,” *IEEE S&P*, 2014.

[21] S. Costan and S. Devadas, “Intel SGX Explained,” IACR ePrint 2016/086.

[22] A. Narayanan, J. Bonneau, E. Felten, et al., *Bitcoin and Cryptocurrency Technologies*, Princeton University Press, 2016.

❖ **Privacy and Federated Learning**

[23] C. Dwork and A. Roth, “The Algorithmic Foundations of Differential Privacy,” 2014.

[24] H. B. McMahan et al., “Communication-Efficient Learning of Deep Networks from Decentralized Data,” *AISTATS*, 2017.

❖ Human–AI Interaction, Explainability, Trust

[25] X. Amershi et al., “Guidelines for Human–AI Interaction,” *CHI*, 2019.

[26] T. Miller, “Explanation in Artificial Intelligence: Insights from the Social Sciences,” *AI Journal*, 2019.

[27] D. Kahneman, O. Sibony, and C. Sunstein, *Noise: A Flaw in Human Judgment*, HarperCollins, 2021.

❖ Standards and Pervasive Computing

[28] M. Satyanarayanan, “Pervasive Computing: Vision and Challenges,” *IEEE Pervasive*, 2001.

[29] M. Weiser, “The Computer for the 21st Century,” *Scientific American*, 1991.

[30] T. Gebru et al., “Datasheets for Datasets,” *Communications of the ACM*, 2021.

[31] M. Mitchell et al., “Model Cards for Model Reporting,” *FAT** (FAccT), 2019.