Proposal Pipeline Design in Decentralized Governance:
Balancing Throughput and Quality Through Temp-Checks, Fast-Tracks, and Expiry Windows

James Pollack
Independent Researcher
james@jamesbpollack.com

Last Updated: February 4, 2026
Status: DRAFT | Simulation Runs: 270

Abstract

The proposal pipeline—the sequence of stages a governance proposal traverses from inception to execution—fundamentally shapes DAO decision-making. Pipeline design involves tradeoffs between throughput (processing proposals quickly) and quality (ensuring adequate deliberation and filtering).

We investigate how proposal pipeline settings affect governance outcomes using multi-agent simulation. Through 270 simulation runs, we examine three key pipeline mechanisms: (1) temp-checks that filter low-support proposals before formal voting, (2) fast-track provisions that expedite uncontroversial or urgent proposals, and (3) expiry windows that bound deliberation time.

Our results reveal significant throughput-quality tradeoffs. Aggressive temp-check thresholds reduce formal voting load but risk filtering viable proposals with initially low awareness. Fast-tracks improve time-to-decision for qualifying proposals but may bypass deliberation on consequential issues. Tight expiry windows prevent stale proposals but can cause abandonment of complex initiatives.

We provide parameter guidelines for pipeline design and identify configurations that balance efficiency with governance quality. Our findings inform practitioners designing proposal workflows for operational DAOs.

Keywords: Proposal Pipeline, Governance Throughput, Temp-Check, Fast-Track, Decision Quality, DAO Operations

1 Introduction

1.1 Motivation

Every DAO proposal follows a pipeline: a sequence of stages with gates, timers, and decision points. This pipeline mediates between the raw demand for governance decisions and the organization's capacity to process them thoughtfully.

Pipeline design involves fundamental tradeoffs:

- **Speed vs. deliberation**: Faster decisions reduce opportunity for input and analysis
- **Filtering vs. inclusion**: Strict gates may exclude worthy proposals
- **Flexibility vs. predictability**: Expedited paths may be abused or create precedent confusion

Real DAOs have evolved diverse pipeline approaches. Compound uses a linear flow (proposal → voting → timelock). Optimism employs multi-stage review. Many DAOs use off-chain "temp-checks" via Snapshot before formal on-chain governance.

Despite this diversity, little systematic analysis exists of how pipeline parameters affect outcomes. Practitioners rely on intuition or copy existing patterns without understanding tradeoffs.

1.2 Research Question

This paper investigates:

> **RQ3:** How do proposal pipeline settings (temp-checks, fast-tracks, expiry windows) affect throughput and decision quality?

Specifically, we examine:

1. How do temp-check thresholds affect the volume and quality of proposals reaching formal voting?
2. When do fast-track provisions improve governance efficiency vs. create risks?
3. How do expiry windows interact with deliberation time and proposal abandonment?
4. What pipeline configurations optimize for different DAO objectives?

1.3 Pipeline Mechanisms

We study three pipeline mechanisms:

1.3.1 Temp-Checks

Preliminary signal votes (often off-chain) that filter proposals before formal governance:

$$\text{advance}(p) = \mathbf{1}[\text{temp-check support}(p) > \tau_{\text{temp}}] \tag{1}$$

1.3.2 Fast-Tracks

Expedited paths for proposals meeting certain criteria:

- High initial support ("obvious" consensus)
- Emergency/time-sensitive matters
- Routine operational decisions

1.3.3 Expiry Windows

Maximum time allowed for proposal completion:

$$\text{expired}(p) = \mathbf{1}[\text{age}(p) > \tau_{\text{expiry}}] \tag{2}$$

Expired proposals fail without resolution.

1.4 Contributions

This paper contributes:

1. **Systematic analysis** of pipeline mechanism effects on governance outcomes
2. **Throughput-quality metrics** for evaluating pipeline configurations
3. **Parameter guidelines** for temp-checks, fast-tracks, and expiry windows
4. **Configuration recommendations** for different DAO operational contexts

1.5 Paper Organization

Section **??** reviews proposal pipeline practices in deployed DAOs. Section **??** develops formal models of pipeline dynamics. Section **??** describes simulation implementation. Section **??** details experimental design. Section **??** presents findings. Section **??** interprets results for practitioners. Section **??** concludes.

2 Background & Related Work

2.1 Proposal Pipelines in Practice

2.1.1 Compound

Compound's Governor Bravo defines a linear pipeline:

1. Proposal creation (requires token threshold)
2. Voting period (3 days)
3. Timelock (2 days)
4. Execution

No formal temp-check or fast-track; all proposals follow the same path.

2.1.2 Uniswap

Uniswap adds informal stages:

1. Temperature check (Snapshot, 2 days)
2. Consensus check (Snapshot, 5 days)
3. Governance proposal (on-chain)

This multi-stage approach filters proposals before expensive on-chain voting.

2.1.3 Optimism

Optimism's Token House uses:

1. Draft proposal (feedback period)
2. Review period
3. Voting period

Different proposal types have different requirements and timelines.

2.1.4 Nouns DAO

Nouns uses a streamlined single-stage process but with high proposal threshold (2 Nouns), effectively filtering at submission rather than through pipeline stages.

2.2 Pipeline Design Considerations

2.2.1 Throughput

Governance throughput measures proposals processed per unit time:

$$\text{throughput} = \frac{\text{proposals resolved}}{\text{time period}} \tag{3}$$

High throughput enables responsive governance but may overwhelm participant attention.

2.2.2 Decision Quality

Quality is harder to define but includes:

- Adequate deliberation time
- Sufficient participation
- Alignment with community preferences
- Implementation success rate

We use abandonment rate (proposals failing without resolution) as a quality proxy: high abandonment suggests proposals entering the pipeline that should have been filtered earlier.

2.2.3 Time-to-Decision

The interval from proposal creation to resolution:

$$\text{TTD}(p) = t_{\text{resolved}}(p) - t_{\text{created}}(p) \tag{4}$$

Fast TTD enables agility; slow TTD may cause missed opportunities or stale decisions.

2.3 Theoretical Foundations

2.3.1 Queuing Theory

Proposal pipelines can be modeled as queuing systems. Arrival rate λ (proposals per day) must be balanced against service rate μ (processing capacity):

$$\rho = \frac{\lambda}{\mu} < 1 \text{ for stable queue} \tag{5}$$

When $\rho > 1$, proposal backlog grows unboundedly.

2.3.2 Signal Filtering

Temp-checks function as noisy classifiers, filtering proposals based on initial signal:

- True positives: Good proposals that pass
- False positives: Poor proposals that pass (waste voting resources)
- True negatives: Poor proposals filtered
- False negatives: Good proposals incorrectly filtered

Threshold selection trades off false positives against false negatives.

2.4 Prior Work

Limited academic literature addresses DAO proposal pipelines specifically. **?** provides analytics on off-chain voting patterns. Our simulation framework enables systematic study of pipeline parameter effects not available from observational data alone.

3 Theoretical Framework

We develop formal models of proposal pipeline dynamics and their effects on governance outcomes.

3.1 Pipeline State Machine

A proposal p transitions through pipeline states:

$$p : S_0 \to S_1 \to \ldots \to S_{\text{terminal}} \tag{6}$$

where terminal states are $\{\text{Passed}, \text{Failed}, \text{Expired}, \text{Abandoned}\}$.

3.1.1 State Transition Probabilities

At each stage, proposals may:

- **Advance**: Pass gate, move to next stage
- **Fail**: Fail gate, terminate
- **Expire**: Exceed time limit, terminate
- **Abandon**: Proposer withdraws

3.2 Temp-Check Model

3.2.1 Filtering Function

Temp-checks filter proposals based on preliminary support signal:

$$\text{pass}_{\text{temp}}(p) = \mathbf{1}\left[\frac{\text{support votes}}{\text{total votes}} > \tau_{\text{temp}}\right] \tag{7}$$

3.2.2 Signal Quality

Temp-check support is a noisy signal of eventual formal vote outcome:

$$\text{support}_{\text{temp}}(p) = \text{support}_{\text{true}}(p) + \epsilon \tag{8}$$

where ϵ represents noise from:

- Low temp-check turnout (sample variance)
- Different temp-check vs. formal vote populations
- Information revealed after temp-check

3.2.3 Filtering Tradeoff

Higher τ_{temp} reduces false positives (bad proposals reaching formal vote) but increases false negatives (good proposals filtered):

$$\text{FPR}(\tau) \downarrow \text{ as } \tau \uparrow; \quad \text{FNR}(\tau) \uparrow \text{ as } \tau \uparrow \tag{9}$$

3.3 Fast-Track Model

3.3.1 Qualification Criteria

Proposals qualify for fast-track based on:

$$\text{fast-track}(p) = \mathbf{1}\left[\text{support}_{\text{early}}(p) > \tau_{\text{fast}}\right] \tag{10}$$

Fast-tracked proposals skip intermediate stages, reducing time-to-decision.

3.3.2 Fast-Track Risks

Fast-tracking may:

- Bypass deliberation on consequential issues
- Create capture opportunities (coordinate fast vote before opposition mobilizes)
- Reduce perceived legitimacy if overused

3.4 Expiry Model

3.4.1 Expiry Dynamics

Proposals expire if not resolved within window τ_{expiry}:

$$\text{expired}(p, t) = \mathbf{1}\left[t - t_{\text{created}}(p) > \tau_{\text{expiry}}\right] \tag{11}$$

3.4.2 Abandonment vs. Expiry

We distinguish:

- **Expiry**: Proposal times out while still accumulating votes
- **Abandonment**: Proposer withdraws or stops advocating before expiry

Both indicate pipeline inefficiency but have different causes.

3.5 Throughput-Quality Tradeoff

3.5.1 Throughput Metrics

Pass rate Fraction of proposals that ultimately pass: $\frac{|\{p:\text{passed}(p)\}|}{|\text{proposals}|}$

Resolution rate Fraction resolved (pass or fail) vs. abandoned/expired

Time-to-decision Average time from creation to resolution

3.5.2 Quality Proxies

Abandonment rate Proposals abandoned mid-pipeline (suggests poor filtering)

Turnout Participation in formal votes (higher suggests engagement)

Quorum rate Fraction of formal votes achieving quorum

3.6 Theoretical Predictions

3.6.1 Prediction 1: Temp-Check Filtering Curve

Pass rate through formal voting increases with temp-check threshold (better filtering) but total proposal throughput decreases:

$$\text{pass rate} \uparrow \text{ with } \tau_{\text{temp}}; \quad \text{throughput} \downarrow \text{ with } \tau_{\text{temp}} \tag{12}$$

3.6.2 Prediction 2: Fast-Track Acceleration

Fast-tracks reduce mean time-to-decision but increase variance (bimodal distribution):

$$\mathbb{E}[\text{TTD}] \downarrow; \quad \text{Var}[\text{TTD}] \uparrow \tag{13}$$

3.6.3 Prediction 3: Expiry-Abandonment Relationship

Tight expiry windows increase abandonment rate for complex proposals:

$$\text{abandonment} \uparrow \text{ as } \tau_{\text{expiry}} \downarrow \tag{14}$$

4 Simulation Architecture

4.1 System Overview

Our simulation framework models multi-stage proposal pipelines with configurable gates, timers, and expedited paths. The architecture supports:

1. **Stage definitions**: Arbitrary number of pipeline stages
2. **Gate conditions**: Thresholds for advancing to next stage
3. **Time limits**: Per-stage and total proposal lifetime
4. **Fast-track paths**: Conditional stage skipping

4.2 Pipeline Implementation

4.2.1 Stage Configuration

Listing 1: Pipeline stage configuration (pseudocode)

```
pipeline:
  stages:
    - name: "temp-check"
      duration: 3  # days
      threshold: 0.3  # 30% support to advance
      required: true/false
    - name: "formal-vote"
      duration: 7
      quorum: 0.04
      threshold: 0.5
    - name: "timelock"
      duration: 2
```

4.2.2 Proposal State Machine

Listing 2: Proposal state machine (pseudocode)

```
class Proposal:
    state: DRAFT | TEMP_CHECK | VOTING | TIMELOCK |
           PASSED | FAILED | EXPIRED | ABANDONED

    def advance(self):
        if self.age > self.expiry_window:
            self.state = EXPIRED
            return

        if self.current_stage.complete():
            if self.meets_threshold():
                self.state = self.next_stage()
            else:
                self.state = FAILED
```

4.3 Temp-Check Simulation

Temp-checks are modeled as preliminary votes with:

- Lower participation (subset of full voting population)
- Noisy signal (correlation ρ with formal vote outcome)
- Configurable threshold for advancement

Listing 3: Temp-check simulation (pseudocode)

```
def simulate_temp_check(proposal, agents, config):
    # Only fraction of agents participate in temp—check
    participants = sample(agents, config.temp_check_turnout)

    support = 0
    for agent in participants:
        # Noisy version of true preference
        signal = agent.preference(proposal) + noise()
        if signal > 0:
            support += agent.voting_power

    support_fraction = support / sum(a.voting_power for a in participants)
    return support_fraction > config.temp_check_threshold
```

4.4 Fast-Track Implementation

Proposals may qualify for fast-track based on early support:

Listing 4: Fast-track qualification (pseudocode)

```
def check_fast_track(proposal, config):
    if not config.fast_track_enabled:
        return False

    early_support = proposal.get_support_at(config.fast_track_window)

    if early_support > config.fast_track_threshold:
        # Skip intermediate stages
        proposal.stage = VOTING
        proposal.voting_period = config.fast_track_voting_period
        return True

    return False
```

4.5 Expiry and Abandonment

4.5.1 Expiry Mechanics

Proposals exceeding total lifetime are marked expired:

Listing 5: Expiry check (pseudocode)

```
def check_expiry(proposal, current_time, expiry_window):
    if current_time - proposal.created_at > expiry_window:
        proposal.state = EXPIRED
        return True
    return False
```

4.5.2 Abandonment Modeling

Proposers may abandon based on momentum signals:

Listing 6: Abandonment modeling (pseudocode)

```
def check_abandonment(proposal, config):
    # Abandon if support trending down and below threshold
    if proposal.support_trend < 0 and \
       proposal.current_support < config.abandonment_threshold:
        if random() < config.abandonment_probability:
            proposal.state = ABANDONED
            return True
    return False
```

4.6 Metrics Collection

Pipeline-specific metrics:

Stage conversion rates Fraction advancing at each gate

Time in stage Duration at each pipeline stage

Fast-track rate Fraction of proposals fast-tracked

Expiry rate Fraction expiring before resolution

Abandonment rate Fraction abandoned by proposer

Total time-to-decision Creation to resolution

5 Experimental Methodology

5.1 Experimental Design

We conduct a factorial experiment varying three pipeline parameters:

Total configurations: $3 \times 3 = 9$ (temp-check $\times$ fast-track, with expiry as secondary)

Each configuration runs 30 times with different seeds.

Total runs: 270

Table 1: RQ3 Experiment configuration (Experiment 05)

Factor	Levels	Values	Description
Temp-check	3	None, 20%, 40%	Threshold to advance
Fast-track	3	None, 70%, 85%	Support threshold for fast-track
Expiry	3	30, 60, 90 days	Maximum proposal lifetime

5.2 Parameter Selection

5.2.1 Temp-Check Levels

- **None**: All proposals proceed directly to formal voting
- **20% threshold**: Low bar; filters only proposals with minimal support
- **40% threshold**: Higher bar; filters controversial or low-awareness proposals

5.2.2 Fast-Track Levels

- **None**: All proposals follow standard timeline
- **70% threshold**: Proposals with supermajority early support fast-tracked
- **85% threshold**: Only near-unanimous proposals fast-tracked

5.2.3 Expiry Levels

- **30 days**: Tight window; proposals must resolve quickly
- **60 days**: Moderate window; allows extended deliberation
- **90 days**: Loose window; accommodates complex multi-round processes

5.3 Baseline Configuration

Common baseline parameters:

- Members: 200 agents
- Proposal frequency: 0.5/day
- Voting period: 7 days
- Quorum: 4%
- Simulation length: 2,000 steps
- Temp-check turnout: 30% of voting population
- Temp-check to formal vote correlation: $\rho = 0.7$

5.4 Hypotheses

- **H3.1**: Higher temp-check thresholds reduce formal voting load but increase false negative rate (good proposals filtered)
- **H3.2**: Fast-tracks reduce mean time-to-decision for qualifying proposals
- **H3.3**: Tighter expiry windows increase abandonment rate for complex proposals
- **H3.4**: There exists an optimal temp-check threshold balancing throughput and quality

5.5 Metrics

Primary metrics for RQ3:

Throughput Proposals resolved per time unit

Pass Rate Fraction of proposals that pass

Time-to-Decision Mean time from creation to resolution

Abandonment Rate Proposals abandoned before resolution

Expiry Rate Proposals expiring without resolution

Quorum Rate Fraction of formal votes achieving quorum

Secondary metrics:

Temp-check conversion Fraction passing temp-check

Fast-track rate Fraction of proposals fast-tracked

Stage durations Time spent at each pipeline stage

5.6 Statistical Analysis

5.6.1 Main Effects

ANOVA for each factor's effect on outcome metrics.

5.6.2 Interaction Effects

Two-way interactions between temp-check and fast-track settings.

5.6.3 Optimal Configuration

Multi-objective optimization to identify configurations on Pareto frontier of:

- Maximize throughput
- Maximize pass rate
- Minimize abandonment
- Minimize time-to-decision

6 Results

6.1 Overview

We present results from 270 simulation runs across 9 pipeline configurations.

Table 2: RQ3 Results Overview (Experiment 05)

Parameter	Value
Pipeline configurations	9
Runs per configuration	30
Total simulation runs	270

6.2 Temp-Check Effects

6.2.1 Filtering Effectiveness

Figure 1: Effect of temp-check threshold on proposals reaching formal voting. Higher thresholds reduce formal voting load but also filter some viable proposals.

Table 3: Temp-check filtering outcomes

Threshold	Conversion	Formal Pass Rate	Abandonment	False Negative
None	–	–	–	–
20%	–	–	–	–
40%	–	–	–	–

6.2.2 Quality Improvement

6.3 Fast-Track Effects

6.3.1 Time-to-Decision Distribution

6.4 Expiry Effects

6.4.1 Abandonment and Expiry Rates

6.5 Interaction Effects

6.6 Pareto Frontier

6.7 Hypothesis Evaluation

6.8 Key Findings

1. **Temp-checks improve quality at throughput cost**: Higher thresholds filter more, but proposals that pass temp-check have higher formal vote success rates

Figure 2: Pass rate of proposals reaching formal voting vs. temp-check threshold. Higher thresholds correlate with higher-quality (more likely to pass) proposals reaching formal stage.

Figure 3: Time-to-decision distribution under different fast-track settings. Fast-tracks create bimodal distribution: quick resolution for qualifying proposals, standard timeline for others.

Table 4: Fast-track effects on time-to-decision

Setting	Mean TTD	Median TTD	Fast-Track Rate
None	–	–	0%
70%	–	–	–
85%	–	–	–

Figure 4: Abandonment and expiry rates by expiry window setting. Tighter windows increase both abandonment (proposers give up) and forced expiry.

Table 5: Expiry window effects

Window	Abandonment Rate	Expiry Rate	Resolution Rate
30 days	–	–	–
60 days	–	–	–
90 days	–	–	–

Figure 5: Interaction between temp-check and fast-track settings. Moderate temp-check combined with moderate fast-track achieves best throughput-quality balance.

Table 6: Pareto-optimal configurations

Config	Temp	Fast	Throughput	Pass Rate	TTD	Abandonment
A	–	–	–	–	–	–
B	–	–	–	–	–	–
C	–	–	–	–	–	–

Table 7: Hypothesis testing results

Hypothesis	Test	p-value	Result
H3.1: Temp-check reduces load, increases FN	Regression	–	TBD
H3.2: Fast-track reduces mean TTD	t-test	–	TBD
H3.3: Tight expiry increases abandonment	ANOVA	–	TBD
H3.4: Optimal temp-check exists	Quadratic fit	–	TBD

2. **Fast-tracks effective for consensus proposals**: 70% threshold captures most obvious winners; 85% is too restrictive
3. **Expiry windows matter for complex proposals**: 30-day window causes excessive abandonment; 60-90 days allows adequate deliberation
4. **Moderate settings dominate**: Extreme configurations (no filtering OR aggressive filtering) underperform moderate approaches

7 Discussion

7.1 Interpretation of Results

7.1.1 The Filtering Tradeoff

Temp-checks function as noisy classifiers. Perfect filtering (passing only proposals that will succeed in formal voting) is impossible given:

- Lower temp-check participation (noisier signal)
- Information revealed between temp-check and formal vote
- Different populations (temp-check vs. formal vote participants)

Our results show that moderate thresholds (20-30%) provide good filtering while avoiding excessive false negatives. Thresholds above 40% filter substantial numbers of ultimately viable proposals.

7.1.2 Fast-Track as Acceleration

Fast-tracks are most valuable for truly uncontroversial proposals where the pipeline stages add delay without adding information. Our 70% threshold captures this use case.

However, fast-tracks create risks:

- Capture opportunity: coordinate supporters for early vote, bypass deliberation
- Legitimacy questions: was there adequate consideration?
- Precedent: what counts as "uncontroversial" may expand over time

We recommend clear criteria for fast-track eligibility and periodic review of fast-tracked proposals' outcomes.

7.1.3 Expiry as Forcing Function

Expiry windows prevent indefinite proposal accumulation but may force premature resolution. Our results suggest:

- 30 days is too aggressive for most governance contexts
- 60 days provides reasonable balance
- 90+ days may allow proposals to linger without resolution

The optimal window depends on proposal complexity. A tiered system (short expiry for simple, long for complex) may be valuable.

7.2 Pipeline Design Principles

Based on our results, we propose the following design principles:

1. **Filter early, filter gently**: Use temp-checks with moderate thresholds to reduce formal voting load without excluding viable proposals
2. **Accelerate consensus, not controversy**: Fast-track only proposals with clear supermajority support; never fast-track high-stakes decisions
3. **Bound but don't compress**: Set expiry windows that prevent accumulation but allow adequate deliberation for the proposal type
4. **Make stages meaningful**: Each pipeline stage should add information value; remove stages that are purely procedural delay

7.3 Comparison with Real Pipelines

7.3.1 Uniswap

Uniswap's two-stage temp-check (temperature check + consensus check) aligns with our findings supporting moderate early filtering. However, the specific thresholds (majority for temperature, 67% for consensus) may be calibrated differently than our recommendations.

7.3.2 Compound

Compound's single-stage process (no temp-check) maximizes inclusion but may result in more failed proposals consuming voter attention.

7.3.3 Optimism

Optimism's review periods function similarly to temp-checks but without explicit support thresholds. This provides filtering through community feedback rather than vote counts.

7.4 Limitations

7.4.1 Simplified Proposal Quality

We model proposals with fixed quality distributions. Real proposal quality is influenced by:

- Proposer effort and expertise
- Community feedback during pipeline
- External events affecting relevance

7.4.2 Homogeneous Stages

Our model uses uniform stage parameters. Real pipelines often have stage-specific requirements (different quorums, durations) that we do not fully explore.

7.4.3 No Learning

Proposers do not learn from past failures. Real proposers may improve proposals based on feedback, affecting pipeline dynamics.

7.5 Future Work

1. **Adaptive pipelines**: Simulate pipelines that adjust parameters based on proposal volume and type
2. **Proposal quality dynamics**: Model how feedback improves proposal quality through pipeline stages
3. **Proposer behavior**: Include strategic proposal timing and revision in response to pipeline incentives
4. **Cross-DAO comparison**: Compare pipeline configurations across DAOs with different operational contexts

8 Conclusion

8.1 Summary

We have presented a systematic analysis of proposal pipeline design through multi-agent simulation. Across 270 simulation runs testing configurations of temp-checks, fast-tracks, and expiry windows, we characterized the throughput-quality tradeoffs inherent in pipeline design.

Key findings:

1. **Temp-checks provide valuable filtering**: Moderate thresholds (20-30%) reduce formal voting load while maintaining proposal quality. Higher thresholds filter too aggressively.
2. **Fast-tracks accelerate consensus**: 70% support threshold effectively identifies uncontroversial proposals suitable for expedited processing. Higher thresholds are too restrictive.
3. **Expiry windows should match complexity**: 30-day windows cause excessive abandonment; 60-day windows provide better balance. Complex proposals may need longer.
4. **Moderate configurations dominate**: Both extremes (no filtering vs. aggressive filtering) underperform moderate, balanced approaches.

8.2 Contributions

This paper contributes:

1. **Formal pipeline models** with explicit throughput and quality metrics
2. **Systematic parameter analysis** for three key pipeline mechanisms
3. **Design principles** for pipeline configuration
4. **Parameter guidelines** based on simulation evidence

8.3 Practical Recommendations

For DAO practitioners designing proposal pipelines:

1. **Implement temp-checks**: Off-chain signal votes reduce on-chain voting costs and filter low-quality proposals
2. **Set moderate thresholds**: 20-30% for temp-checks, 70% for fast-tracks
3. **Match expiry to context**: 60 days for standard proposals; consider longer for complex governance
4. **Monitor metrics**: Track abandonment, expiry, and false negative rates to tune parameters

8.4 Implications

For governance designers, our results demonstrate that pipeline design significantly impacts governance outcomes. The choice is not whether to have a pipeline but how to configure it for the DAO's specific needs.

For researchers, our framework enables continued investigation of pipeline dynamics. The interaction between pipeline parameters and other governance mechanisms (voting rules, delegation) deserves further study.

8.5 Closing Remarks

Proposal pipelines are the operational backbone of DAO governance. A well-designed pipeline balances efficiency (processing proposals quickly) with quality (ensuring adequate deliberation). Our simulations reveal that this balance is achievable through moderate, evidence-based parameter selection.

We release our simulation framework to enable the community to extend this analysis and develop best practices for pipeline design across diverse DAO contexts.