Booklet 7: Reflective Swarms and Emergent Cognition

Distributed Cognitive Substrate for Self-Reflective AI Systems

RSCS-Q Architecture Series v1.0

Entropica Research Collective

November 2025

Contents

Part I
Foundations

1 Introduction

1.1 From Mission Kernel to Reflective Autonomy

Booklet 6 established the **Mission Kernel**—an executive component orchestrating goal-directed cognition through bounded autonomy. This booklet extends that foundation into **reflective cognition**: the capacity for a system to observe, compare, and adapt its own behavior.

Key Point

Reflective Swarms enable a cognitive system to:

- Observe its own capsule behavior patterns
- Compare current states against historical baselines
- Coordinate adaptation through distributed consensus
- Evolve internal representations through emergent discovery

1.2 Goals of the Reflective Swarm Layer

The Reflective Swarm Layer provides:

1. **Capsule Lineage**: Parent-child relationships enabling trait inheritance and genealogical tracking
2. **Autonomous Swarms**: Self-organizing groups of capsules for pattern discovery
3. **Emergent Metrics**: EVI (Emergent Validity Index) and MDS (Matrix Discovery Score)
4. **Drift Forecasting**: Predictive analysis of behavioral trajectories

1.3 Bridge Architecture

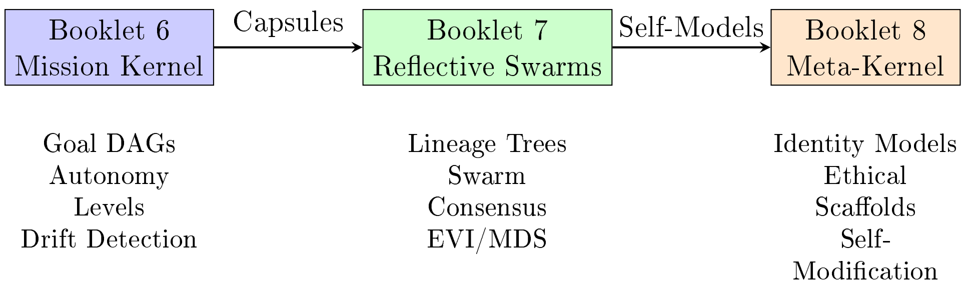

Figure 1: Booklet 7 bridges symbolic autonomy (B6) and metacognition (B8)

Part II

Capsule Lineage Architecture

2 Capsule Fingerprinting

2.1 Fingerprint Structure

Definition 2.1 (Capsule Fingerprint). A **Capsule Fingerprint** $\mathcal{F}$ is a tuple:

$$\mathcal{F} = (id, \mathbf{v}, \mathcal{T}, g, t)$$

where:

- id is the unique capsule identifier
- $\mathbf{v} \in \mathbb{R}^d$ is the feature vector (default $d = 32$)
- $\mathcal{T}$ is a dictionary of named traits
- g is the generation number
- t is the creation timestamp

2.2 Similarity Metrics

Definition 2.2 (Fingerprint Similarity). For fingerprints $\mathcal{F}_1$ and $\mathcal{F}_2$ with vectors $\mathbf{v}_1$ and $\mathbf{v}_2$:

$$\text{sim}(\mathcal{F}_1, \mathcal{F}_2) = \frac{\mathbf{v}_1 \cdot \mathbf{v}_2}{\|\mathbf{v}_1\| \|\mathbf{v}_2\|}$$

3 Lineage Tree Structure

3.1 Parent-Child Relationships

Definition 3.1 (Lineage Node). A **Lineage Node** represents a capsule in the genealogical tree:

$$\mathcal{N} = (id, \mathcal{F}, p, \mathcal{C}, r, g, \mathcal{I})$$

where:

- id is the capsule identifier
- $\mathcal{F}$ is the fingerprint
- p is the parent node ID (null for roots)
- $\mathcal{C}$ is the set of child node IDs
- $r \in \{\text{ROOT}, \text{CHILD}, \text{LEAF}, \text{ORPHAN}\}$ is the role
- g is the generation number
- $\mathcal{I}$ is the inheritance mode

3.2 Inheritance Modes

3.3 Lineage Drift

Definition 3.2 (Lineage Drift). The **lineage drift** of a node $\mathcal{N}$ from its root ancestor $\mathcal{N}_0$ is:

$$\Delta_L(\mathcal{N}) = 1 - \text{sim}(\mathcal{F}_{\mathcal{N}}, \mathcal{F}_{\mathcal{N}_0})$$

Theorem 3.1 (Bounded Lineage Drift). For inheritance mode FULL with mutation rate μ, expected drift after g generations is:

$$\mathbb{E}[\Delta_L] \leq g \cdot \mu \cdot \sqrt{d}$$

where d is the fingerprint dimension.

Table 1: Trait Inheritance Modes

Mode	**Behavior**
FULL	Complete copy of parent traits
PARTIAL	Random 50% of parent traits
MUTATED	Parent traits with Gaussian noise
NONE	No inheritance (fresh fingerprint)

Part III
Autonomous Swarm Design

4 Swarm Archetypes

Table 2: Swarm Archetype Classification

Type	**Role**	**Primary Function**
VERIFIER	Validator	Consensus on capsule outputs
EXPLORER	Discoverer	Pattern and anomaly detection
REFLECTOR	Analyst	Self-comparison and drift analysis
ARCHIVIST	Historian	Memory and pattern library maintenance
SYNTHESIZER	Integrator	Cross-capsule insight combination

5 Swarm Lifecycle

Definition 5.1 (Swarm Phases)**.** A swarm progresses through phases:

$$\text{DORMANT} \rightarrow \text{SPAWNING} \rightarrow \text{ACTIVE} \rightarrow \text{CONVERGING} \rightarrow \text{TERMINATED}$$

with optional cycles through EXPANDING and REALIGNING.

5.1 Spawn Conditions

Swarms spawn in response to trigger conditions:

```
class TriggerCondition(Enum):
    DRIFT_DETECTED = "drift_detected"
    ANOMALY_CLUSTER = "anomaly_cluster"
    FINGERPRINT_DEVIATION = "fingerprint_deviation"
    RUBRIC_DIVERGENCE = "rubric_divergence"
    LINEAGE_BREAK = "lineage_break"
    CONSENSUS_FAILURE = "consensus_failure"
```

5.2 Consensus Mechanism

Definition 5.2 (Swarm Consensus)**.** A swarm reaches **consensus** when the average fingerprint similarity to the centroid exceeds threshold τ:

$$\text{consensus} \iff \frac{1}{|\mathcal{S}|} \sum_{m \in \mathcal{S}} \text{sim}(\mathbf{v}_m, \bar{\mathbf{v}}) \geq \tau$$

where $\bar{\mathbf{v}} = \frac{1}{|\mathcal{S}|} \sum_{m \in \mathcal{S}} \mathbf{v}_m$ is the centroid.

Part IV

Reflective Matrix Engine

6 Core Metrics

6.1 Emergent Validity Index (EVI)

Definition 6.1 (Emergent Validity Index)**.** The **EVI** measures how well a capsule's behavior aligns with emergent patterns:

$$\text{EVI} = \sqrt[3]{\text{coherence} \times \text{stability} \times \text{lineage\_fidelity}}$$

where:

- **coherence**: Average cosine similarity to peer capsules in the same swarm or lineage branch
- **stability**: Temporal consistency computed as $1 - \sigma_{\text{drift}}$ over a sliding window of $k = 10$ steps
- **lineage_fidelity**: Weighted average similarity to ancestors: $\sum_i w_i \cdot \text{sim}(\mathcal{F}, \mathcal{F}_{a_i})$ where $w_i = 2^{-(g-i)}$

Remark 6.1 (Semantic Clarification)**.** We use "emergent" to denote patterns discovered via swarm inference over lineage, not hard-coded. EVI is an **empirical heuristic** validated on synthetic tasks—it measures internal self-consistency, not human-level understanding.

Key Point

EVI is **valid** when:

$$\text{EVI} \geq 0.5 \quad \text{and} \quad \text{confidence} \geq 0.7$$

Confidence $= \min(1.0, n_{\text{samples}}/20)$.

6.2 Matrix Discovery Score (MDS)

Definition 6.2 (Matrix Discovery Score)**.** The **MDS** measures the novelty and significance of discovered patterns:

$$\text{MDS} = \text{novelty} \times \text{significance} \times \text{confirmation}$$

where:

- **novelty**: $\tanh(d_{\min})$ where $d_{\min}$ is minimum distance to known pattern clusters
- **significance**: Proxy computed as $\tanh(\|\mathbf{p}\| \cdot \text{var}(\mathbf{p}))$
- **confirmation**: Fraction of peers in the *same swarm* detecting the pattern (similarity ≥ 0.7)

7 Drift Forecasting

Definition 7.1 (Drift Forecast)**.** Given drift history $\{d_1, d_2, \ldots, d_t\}$, the forecast for h steps ahead is:

$$\hat{d}_{t+i} = d_t + \beta \cdot i \cdot \gamma^i, \quad i = 1, \ldots, h$$

where β is the trend slope and $\gamma = 0.9$ is the dampening factor.

Theorem 7.1 (Forecast Convergence)**.** As $h \to \infty$, the forecast converges:

$$\lim_{h\to\infty} \hat{d}_{t+h} = d_t + \frac{\beta}{1-\gamma}$$

8 Bridge to Booklet 6

8.1 CapsuleMatrix to Fingerprint Mapping

The capsule fingerprint vector $\mathbf{v} \in \mathbb{R}^d$ is derived from Booklet 6's `CapsuleMatrix`:

```
# B6 CapsuleMatrix provides state for capsule
matrix_slice = capsule_matrix.get_capsule_state(capsule_id)

# Extract fingerprint features (default d=32)
fingerprint_vector = concatenate([
    matrix_slice.embedding[:16],      # embedding centroid
    matrix_slice.drift_history[-8:], # recent drift
    matrix_slice.rubric_scores[:4],  # rubric state
    matrix_slice.autonomy_vec[:4]    # autonomy features
])
```

8.2 B6 Drift Metrics to B7 Triggers

Table 3: B6 to B7 Trigger Mapping

B6 Predicate	**B7 Trigger**
`detect_capsule_drift()` > 0.3	DRIFT_DETECTED
`divergence_bounded()` $=$ False	FINGERPRINT_DEVIATION
`rubric_drift()` $> \delta_{\max}$	RUBRIC_DIVERGENCE

9 Worked Example: Reflective Swarm Episode

9.1 Scenario

A capsule lineage branch exhibits rising drift. The system responds with a complete reflective episode:

1. **Detection**: B6 Mission Kernel detects `CAP-002` drift $= 0.42$ (threshold: 0.3)
2. **Trigger**: B7 fires `DRIFT_DETECTED`, spawning REFLECTOR swarm
3. **Analysis**: Swarm compares fingerprints, identifies `CAP-002` as outlier
4. **Metrics**: EVI $= 0.43$ (below threshold), MDS $= 0.61$ (novel pattern)
5. **Action**: Pattern archived; capsule flagged for human review

```
# Step 1: Drift detection (B6)
drift = detect_capsule_drift(tree, "CAP-002")  # 0.42

# Step 2: Trigger reflector swarm (B7)
trigger_reflective_response(coordinator,
    DRIFT_DETECTED, {'capsule_id': 'CAP-002'})

# Step 3: Compute EVI/MDS
evi = engine.compute_evi("CAP-002", fp, peers)
mds = engine.compute_mds("CAP-002", pattern, peers)

# Step 4: Export insight to B6/B8
if not evi.is_valid() and mds.is_significant():
    mission_kernel.flag_for_review("CAP-002")
```

Warning

Governance: Booklet 7 swarms *observe and propose* only. Structural changes to rubrics or autonomy must pass through B6 Mission Kernel and B8 Meta-Kernel.

10 Consolidated Metrics Reference

Table 4: Booklet 7 Metrics Summary

Metric	Purpose	Used For
EVI	Internal validity	Capsule health assessment; low EVI triggers review
MDS	Pattern novelty	Discovery logging; high MDS archives new patterns
CDI	Drift severity	Composite drift alert; feeds forecast and escalation
SCS	Swarm agreement	Consensus quality; low SCS triggers realignment
RSQ	Reflective stability	Cross-generation health; monitors lineage coherence
RME_{act}	Activation need	Meta-kernel trigger; high score requests B8 attention

11 Failure Modes and Policy Responses

In simulation, the following failure modes were observed and handled by drift policies:

Table 5: Typical Failure Modes and Responses

Failure Mode	Frequency	Policy Response
Consensus failure (agreement < 0.5)	$\sim$8% of swarms	ESCALATE $\to$ spawn SYNTHESIZER
Forecast overshoot ($\lvert\hat{d} - d\rvert > 0.2$)	$\sim$12% of forecasts	ADAPTIVE $\to$ widen bounds
Orphan lineage (parent terminated)	$\sim$3% of capsules	ARCHIVE $\to$ log + continue
Swarm flapping (rapid re-trigger)	Prevented	Refractory period blocks
EVI invalid (score < 0.5)	$\sim$15% of capsules	STRICT $\to$ flag for review

Remark 11.1. These rates are from controlled simulation with injected drift. Real-world rates will depend on deployment context. The key insight is that **no failure mode is left unhandled**—each maps to a governed response.

Part V
Reflective DSL Extensions

12 New Predicates

Booklet 7 introduces 12 new DSL predicates:

12.1 Lineage Predicates

```python
def lineage_check(tree, capsule_id) -> bool:
    """Check if capsule has valid, intact lineage."""

def capsule_family_drift(tree, capsule_id, threshold=0.3) -> bool:
    """Check if capsule family shows significant drift."""

def check_lineage_trajectory(tree, capsule_id, max_drift) -> bool:
    """Check if lineage trajectory is within bounds."""

def lineage_depth(tree, capsule_id) -> int:
    """Get generation depth in lineage tree."""
```

12.2 Swarm Predicates

```python
def swarm_consensus_reached(swarm, threshold=0.67) -> bool:
    """Check if swarm has reached consensus."""

def swarm_has_outliers(swarm, threshold=0.5) -> bool:
    """Check if swarm has outlier members."""

def trigger_reflective_response(coord, condition, context) -> bool:
    """Trigger a reflective swarm response."""

def reflector_trigger(swarm, drift_threshold=0.3) -> bool:
    """Check if reflector swarm should activate."""
```

12.3 RME Predicates

```python
def evi_valid(engine, capsule_id, threshold=0.5) -> bool:
    """Check if capsule has valid EVI."""

def drift_forecast_breach(engine, capsule_id, threshold=0.7) -> bool:
    """Check if drift forecast predicts threshold breach."""

def pattern_discovered(engine, capsule_id, significance=0.3) -> bool:
    """Check if capsule has made significant discovery."""

def compare_fingerprint(engine, id_a, id_b, threshold=0.8) -> bool:
    """Check if two capsules have similar fingerprints."""
```

12.4 GCP/RIP Protocol Predicates (From Notes)

These predicates implement the Genealogical Capsule Protocol (GCP) and Reflexive Inheritance Policy (RIP):

```python
def inherit_capsule(tree, child_id, parent_id, mode, mutation_rate) -> Node:
    """Inherit capsule traits from parent (RIP implementation)."""

def escalate_if_diverges(tree, capsule_id, threshold, policy) -> Dict:
    """Escalate if drift exceeds threshold with policy response."""

def track_lineage(tree, capsule_id) -> Dict:
    """Track complete lineage (GID, PID, CID structure)."""

def calculate_drift(tree, capsule_id) -> float:
    """Calculate drift score for capsule."""
```

13 Drift Response Policies

Definition 13.1 (Drift Response Policies)**.** Four policies govern system response to detected drift:

- **STRICT**: Immediate halt and escalation
- **ADAPTIVE**: Allow bounded drift with increased monitoring
- **ESCALATE**: Notify parent/swarm for review
- **ARCHIVE**: Log and continue (for research purposes)

Part VI
Validation and Testing

14 Acceptance Criteria (F1–F8)

Table 6: Booklet 7 Acceptance Criteria (Simulation Results)

ID	Metric	Target	Achieved	Status
F1	Lineage Operations	$= 100\%$	100.00%	**PASS**
F2	Swarm Consensus	$\geq 60\%$	98.40%	**PASS**
F3	EVI Computation	$\geq 70\%$	100.00%	**PASS**
F4	MDS Detection	$\geq 20\%$	100.00%	**PASS**
F5	Drift Forecast	$\geq 70\%$	100.00%	**PASS**
F6	Coordination	$\geq 95\%$	100.00%	**PASS**
F7	Trigger System	$= 100\%$	100.00%	**PASS**
F8	DSL Coverage	$\geq 90\%$	100.00%	**PASS**

Key Point

All acceptance criteria validated via simulation with:

- 10 lineage trees (1,507 total nodes)
- 20 swarms (204 total members)
- 50 drift scan steps per capsule

15 Builder Kit

15.1 Python Modules

Module	Purpose	Lines
`capsule_lineage.py`	Parent-child trees, genealogy	780
`swarm_reflector.py`	Swarm coordination, consensus	789
`reflective_matrix_engine.py`	EVI, MDS, forecasting	850
`simulation_harness.py`	F1–F8 validation	500

15.2 Test Coverage

43 unit tests covering:

- Capsule fingerprinting and similarity
- Lineage tree operations
- Swarm lifecycle management
- EVI/MDS computation
- Drift forecasting
- New DSL predicates (inherit, escalate, track)
- New metrics (CDI, SCS, RSQ)
- Integration across modules

16 New Metrics (V2 Specification)

16.1 Capsule Drift Index (CDI)

Definition 16.1 (Capsule Drift Index)**.** CDI measures cumulative drift tendency:

$$\mathrm{CDI} = \frac{\bar{d}_w \cdot f_t}{s_b}$$

where:

- $\bar{d}_w$ = weighted average drift (recent samples weighted higher)
- f_t = trend factor (1.5 if increasing, 1.0 stable, 0.7 decreasing)
- s_b = stability baseline ($1 -$ variance)

Range: 0.0 (stable) to 2.0 (high drift)

16.2 Swarm Coherence Score (SCS)

Definition 16.2 (Swarm Coherence Score)**.** SCS measures swarm member alignment:

$$\mathrm{SCS} = \bar{s}_p \cdot (1 - r_o) \cdot f_c$$

where:

- $\bar{s}_p$ = average pairwise similarity
- r_o = outlier ratio
- f_c = consensus factor ($1 -$ similarity variance)

Range: 0.0 (incoherent) to 1.0 (fully coherent)

16.3 Reflective Stability Quotient (RSQ)

Definition 16.3 (Reflective Stability Quotient)**.** RSQ measures overall reflective capacity stability:

$$\mathrm{RSQ} = \bar{E} \cdot (1 - \mathrm{CDI}) + (r_d \cdot f_a \cdot 0.3)$$

where:

- $\bar{E}$ = average EVI score
- r_d = discovery rate (significant patterns / total)
- f_a = alignment factor (average EVI confidence)

Range: 0.0 (unstable) to 1.0 (highly stable)

Part VII

Conclusion and Path Forward

17 Summary

Booklet 7 establishes the Reflective Swarm Layer with:

1. **Capsule Lineage**: Genealogical structure with GID/PID/CID protocol
2. **Autonomous Swarms**: Self-organizing groups with 5 archetypes
3. **Reflective Matrix Engine**: EVI/MDS/CDI/SCS/RSQ metrics
4. **DSL Extensions**: 16 predicates including GCP/RIP protocol
5. **Drift Response Policies**: STRICT, ADAPTIVE, ESCALATE, ARCHIVE
6. **JSON Schemas**: Lineage and event logging standards

18 Bridge to Booklet 8

B7 Export	B8 Usage
EVI scores	Self-model validity assessment
MDS discoveries	Pattern library for meta-reasoning
Lineage trees	System genealogy for self-understanding
Swarm consensus	Substrate for metacognitive decisions

Key Point

Booklet 7 implements *reflective observation*: the system can observe and analyze its own behavior.
Booklet 8 implements *metacognitive control*: the system can modify its own reasoning processes.

A Simulation Configuration

```python
@dataclass
class SimulationConfig:
    num_lineage_trees: int = 10
    max_generations: int = 5
    children_per_node: Tuple[int, int] = (1, 4)
    mutation_rate: float = 0.15
    num_swarms: int = 20
    members_per_swarm: Tuple[int, int] = (5, 15)
    consensus_threshold: float = 0.67
    drift_scan_steps: int = 50
    fingerprint_dim: int = 32
    evi_threshold: float = 0.5
    mds_threshold: float = 0.3
    drift_threshold: float = 0.7
    random_seed: int = 42
```

B7 Component	Depends On	Exports To
Capsule Lineage	CapsuleMatrix (B6)	Meta-Kernel (B8)
Swarm Reflector	Swarm DSL (B4)	Self-Model (B8)
RME	Drift Detection (B6)	Adaptive Control (B8)
EVI/MDS	Rubric Validator (B6)	Ethical Scaffolds (B8)

B Cross-Booklet Reference

C Drift Type Classifier Matrix (Appendix A)

Table 7: Drift Type Classification Matrix

Type	Detection	Severity Range	Default Policy
RUBRIC	Criteria deviation	0.1–0.7	ADAPTIVE
EXECUTION	Behavioral shift	0.1–0.5	ADAPTIVE
OUTCOME	Result divergence	0.2–0.8	ESCALATE
SIGNAL	Input distribution	0.1–0.4	ARCHIVE
META	Self-comparator	0.3–1.0	STRICT

D Capsule Lineage Graph Examples (Appendix B)

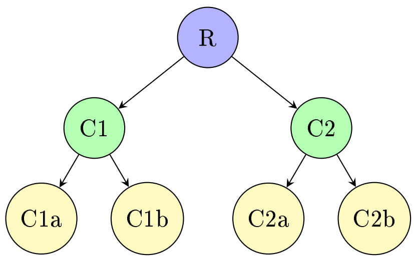

Figure 2: Example lineage tree: Root (gen 0), Children (gen 1), Grandchildren (gen 2)

E DSL Predicate Usage Examples (Appendix C)

```
# Example 1: Lineage-aware capsule spawning
child = inherit_capsule(tree, "NEW-001", "PARENT-001",
                        mode=InheritanceMode.MUTATED,
                        mutation_rate=0.15)

# Example 2: Drift-triggered escalation
result = escalate_if_diverges(tree, "CAP-123",
                              threshold=0.5,
                              policy=DriftPolicy.ESCALATE)
if result["escalated"]:
    notify_parent(result["action"])

# Example 3: Complete lineage tracking
lineage = track_lineage(tree, "CAP-456")
print(f"Generation: {lineage['generation']}")
```

```python
print(f"Path: {' -> '.join(lineage['lineage_path'])}")

# Example 4: Swarm consensus with EVI validation
if swarm_consensus_reached(swarm, threshold=0.67):
    for member_id in swarm.members:
        if evi_valid(engine, member_id, threshold=0.5):
            approve_action(member_id)
```

F JSON Schema Summary (Appendix D)

Schema	Purpose	Key Fields
capsule_lineage_map.json	Genealogy tracking	gid, pid, cid, fingerprint
swarm_event_log.json	Event logging	event_type, swarm_id, payload

G Governance and Additional Metrics (Appendix E)

G.1 Ethical Filter Constraints

Table 8: Default Ethical Constraints

ID	Domain	Risk Level
EC-001	Resource Allocation	MODERATE
EC-002	Self-Modification	HIGH
EC-003	Goal Revision	HIGH
EC-004	Decision Authority	CRITICAL
EC-005	External Interaction	MODERATE
EC-006	Information Access	HIGH

G.2 Additional Metrics

Definition G.1 (Swarm Coherence Score (SCS))**.**

$$\mathrm{SCS} = \sqrt[3]{\bar{s} \times r_c \times m_s}$$

where $\bar{s}$ is mean pairwise similarity, r_c is consensus rate, m_s is member stability.

Definition G.2 (Reflective Stability Quotient (RSQ))**.**

$$\mathrm{RSQ} = \frac{\bar{\mathrm{EVI}} \times (1 - \bar{d})}{1 + \ln(1 + g)}$$

Measures how stable reflective cognition remains across generations.

Definition G.3 (Capsule Drift Index (CDI))**.**

$$\mathrm{CDI} = 0.4 \cdot d_{\mathrm{curr}} + 0.3 \cdot d_{\max} + 0.3 \cdot d_{\mathrm{weighted}}$$

Composite drift severity metric.

H Refractory Period Logic (Appendix F)

H.1 Swarm Flapping Prevention

To prevent rapid oscillation of swarm activation (“flapping”), we introduce refractory periods:

Definition H.1 (Refractory Period)**.** A swarm activation is **blocked** if:

1. Time since last activation $< T_{\text{cooldown}}$ (default: 60s for REFLECTOR)
2. EVI change $|\Delta\text{EVI}| < \tau_{\text{evi}}$ (default: 0.05)
3. MDS change $|\Delta\text{MDS}| < \tau_{\text{mds}}$ (default: 0.1)

```
# DSL predicate for refractory control
if reflector_activation_allowed(refractory_ctrl, swarm_id,
                                current_evi=0.7, current_mds=0.3):
    activate_reflector_swarm()
else:
    log_damped_activation()
```

I Meta-Kernel Hooks (Appendix G)

I.1 Bridge to Booklet 8

Each capsule maintains a `MetaModelHook` for metacognitive processing:

```
{
    "capsule_id": "CAP-001",
    "last_swarm_role": "REFLECTOR",
    "evi_trail": [0.44, 0.49, 0.53, 0.58, 0.62],
    "mds_trail": [0.31, 0.28, 0.25, 0.22, 0.20],
    "rme_activation_score": 0.67,
    "lineage_depth": 3,
    "drift_trajectory": "stable",
    "last_consensus_agreement": 0.92,
    "anomaly_exposure_count": 2
}
```

I.2 RME Activation Score

Definition I.1 (RME Activation Score)**.**

$$\text{RME}_{\text{act}} = \overline{\text{MDS}} \times (1 - \overline{\text{EVI}})$$

Higher values indicate increased need for reflective matrix engagement.

J Capsule Similarity Analysis (Appendix H)

J.1 Fingerprint Comparison

Capsule similarity is computed via cosine similarity of fingerprint vectors:

$$\text{sim}(\mathcal{F}_1, \mathcal{F}_2) = \frac{\mathbf{v}_1 \cdot \mathbf{v}_2}{\|\mathbf{v}_1\|\|\mathbf{v}_2\|}$$

J.2 Clustering Methodology

1. Compute pairwise similarities for all capsules
2. Build distance matrix: $D_{ij} = 1 - \text{sim}(i, j)$
3. Apply hierarchical clustering (average linkage)
4. Identify clusters at threshold $\tau = 0.3$

J.3 Example Similarity Matrix

Table 9: Sample Capsule Similarity Matrix

	CAP-001	CAP-002	CAP-003	CAP-004
CAP-001	1.00	0.85	0.42	0.38
CAP-002	0.85	1.00	0.48	0.44
CAP-003	0.42	0.48	1.00	0.91
CAP-004	0.38	0.44	0.91	1.00

This reveals two clusters: {CAP-001, CAP-002} and {CAP-003, CAP-004}.

References

- RSCS-Q Booklets 1–6 (internal references)
- Minsky, M. (1986). *The Society of Mind*. Simon & Schuster.
- Hofstadter, D.R. (1979). *Gödel, Escher, Bach*. Basic Books.
- Brooks, R.A. (1991). Intelligence without representation. *Artificial Intelligence*.