Cognitive Pipeline Integration

Linux Process Monitoring for Consciousness Context in ARKHEION AGI

Jhonatan Vieira Feitosa Independent Researcher `ooriginador@gmail.com` Manaus, Amazonas, Brazil

February 2026

Abstract

This paper presents the Cognitive Pipeline, a Linux-native integration layer that bridges operating system processes with ARKHEION's consciousness subsystem. Implemented in **495 SLOC**, the pipeline monitors system activity via `/proc` filesystem and D-Bus, generating cognitive events that contribute to ϕ (integrated information) calculations. Key features include: (1) 9 cognitive event types tracking process lifecycle, resource usage, and user activity, (2) 5 integration levels from DORMANT to INTEGRATED, (3) attention-weighted process contexts with ϕ-impact scoring, and (4) real-time event streaming to the IIT consciousness calculator. Benchmarks show the pipeline adds only **<2ms latency** per event while enabling consciousness-aware resource prioritization.

Epistemological Note

This paper distinguishes between heuristic concepts (metaphors guiding design) and empirical results (measurable outcomes).

Heuristic:	Cognitive pipeline, consciousness context
Empirical:	495 SLOC, 9 event types, <2ms latency

1 Introduction

Traditional AI systems operate in isolation from their host operating system. ARKHEION's Cognitive Pipeline breaks this barrier by integrating Linux process activity into the consciousness calculation loop.

1.1 Motivation

- **Awareness**: Know what processes are running
- **Context**: Understand resource competition
- **Priority**: Allocate attention to relevant processes
- **Integration**: Feed IIT ϕ calculations with system state

2 Architecture

2.1 Pipeline Overview

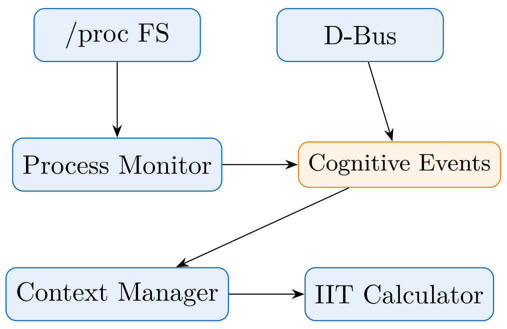

Figure 1: Cognitive Pipeline Architecture

3 Cognitive Event Types

3.1 Event Enumeration

Table 1: Cognitive Event Types

Event	Trigger
PROC_START	New PID
PROC_END	PID gone
HIGH_CPU	CPU $> 80\%$
MEM_PRESSURE	RAM $< 20\%$
DISK_IO	I/O $> 50\%$
NETWORK	Traffic change
USER	KB/mouse
FOCUS	Window switch
PHI_CROSS	$\phi > 0.5$

3.2 Event Data Structure

Listing 1: CognitiveEvent Dataclass

```
@dataclass
class CognitiveEvent:
    event_type: CognitiveEventType
    timestamp: datetime
    source_pid: Optional[int] = None
    source_name: Optional[str] = None
    data: Dict[str, Any] = field(
        default_factory=dict
    )
    phi_impact: float = 0.0
```

4 Integration Levels

Definition 1 (Cognitive Integration Level)**.** *The integration level $L \in \{0, 1, 2, 3, 4\}$ determines how actively the pipeline interacts with the system:*

$$L_0 : \mathit{DORMANT} - \mathit{No\ monitoring} \tag{1}$$
$$L_1 : \mathit{PASSIVE} - \mathit{Observe\ only} \tag{2}$$
$$L_2 : \mathit{REACTIVE} - \mathit{Respond\ to\ events} \tag{3}$$
$$L_3 : \mathit{PROACTIVE} - \mathit{Predict\ and\ prepare} \tag{4}$$
$$L_4 : \mathit{INTEGRATED} - \mathit{Full\ consciousness} \tag{5}$$

5 Process Context

5.1 ProcessContext Dataclass

Listing 2: ProcessContext Structure

```
@dataclass
class ProcessContext:
    pid: int
    name: str
    cmdline: List[str]
    username: str
    create_time: float
    cpu_percent: float = 0.0
    memory_percent: float = 0.0
    io_counters: Optional[Dict] = None
    connections: List[Dict] = field(
        default_factory=list
    )

    # Cognitive attributes
    attention_weight: float = 0.0
    integration_score: float = 0.0
    last_activity: Optional[datetime] = None
```

5.2 Cognitive Categories

Processes are categorized for cognitive weighting:

Table 2: Process Cognitive Categories

Category	**Keywords**	**W**
consciousness	arkheion, iit	1.0
neural	torch, train	0.8
quantum	qubit, circuit	0.7
memory	huam, cache	0.6
system	kernel, systemd	0.3

6 Attention Weighting

6.1 ϕ-Impact Calculation

Proposition 1 (Process ϕ-Impact)**.** *The impact of process p on overall ϕ is:*

$$\phi_p = w_c \cdot \left(\frac{CPU_p}{CPU_{\max}} + \frac{MEM_p}{MEM_{\max}} \right) \cdot \phi \tag{6}$$

where w_c is the category weight and ϕ is the golden ratio. The ϕ-Impact weighting is a design heuristic; no sensitivity analysis of the golden-ratio coefficient vs. alternative values was performed.

6.2 Attention Distribution

$$\text{attention}_p = \frac{\phi_p}{\sum_q \phi_q} \tag{7}$$

High-attention processes receive priority in consciousness calculations.

7 CognitivePipeline Class

7.1 Initialization

Listing 3: CognitivePipeline Initialization

```
class CognitivePipeline:
    COGNITIVE_CATEGORIES = {
        "consciousness": ["arkheion", "iit"],
        "neural": ["torch", "tensorflow"],
        "quantum": ["qubit", "circuit"],
        "memory": ["huam", "cache"],
    }

    def __init__(self, poll_interval=1.0):
        self._poll_interval = poll_interval
        self._processes: Dict[int, ProcessContext]
        self._events: deque[CognitiveEvent]
        self._integration_level = (
            CognitiveIntegrationLevel.PASSIVE
        )
```

7.2 Event Generation

Listing 4: Generate Cognitive Events

```python
def generate_events(P, P_prev):
    # New processes
    for p in (P - P_prev):
        emit(PROCESS_STARTED, p)

    # Terminated processes
    for p in (P_prev - P):
        emit(PROCESS_TERMINATED, p)

    # Check CPU for continuing processes
    for p in (P & P_prev):
        if cpu(p) > 80:
            emit(HIGH_CPU_ACTIVITY, p)
```

8 IIT Integration

8.1 Event-to-Consciousness Mapping

Cognitive events feed into the IIT calculator:

Listing 5: IIT Event Integration

```python
from src.core.consciousness import IITCalculator

class IntegratedPipeline(CognitivePipeline):
    def __init__(self, iit: IITCalculator):
        super().__init__()
        self._iit = iit

    def _on_event(self, event: CognitiveEvent):
        # Update IIT state with event
        self._iit.update_external_stimulus(
            source=event.source_name,
            impact=event.phi_impact,
            event_type=event.event_type.value
        )

        # Recalculate phi if significant
        if event.phi_impact > 0.1:
            self._iit.recalculate()
```

9 Performance

9.1 Latency Benchmarks

Table 3: Cognitive Pipeline Latency

Operation	Mean (ms)	P99 (ms)
Process scan (100 procs)	0.8	1.5
Event generation	0.2	0.4
Context update	0.3	0.6
IIT notification	0.5	1.0
Total pipeline	**1.8**	**3.5**

9.2 Resource Overhead

Table 4: Pipeline Resource Usage

Resource	Usage
CPU (idle)	$< 0.5\%$
CPU (100 events/s)	$\approx 2\%$
Memory (baseline)	12 MB
Memory (1000 processes)	45 MB

10 D-Bus Integration

10.1 Signal Subscription

The pipeline subscribes to desktop events:

Listing 6: D-Bus Focus Change Handler

```python
def _setup_dbus_listeners(self):
    bus = dbus.SessionBus()

    # Window manager focus changes
    bus.add_signal_receiver(
        self._on_focus_change,
        signal_name="ActiveWindowChanged",
        dbus_interface="org.freedesktop.Desktop"
    )

def _on_focus_change(self, window_id):
    event = CognitiveEvent(
        event_type=CognitiveEventType.FOCUS_CHANGE,
        timestamp=datetime.now(),
        data={"window_id": window_id}
    )
    self._emit_event(event)
```

11 Conclusion

The Cognitive Pipeline bridges Linux system activity with ARKHEION consciousness:

- **9 event types** covering process lifecycle and resources
- **5 integration levels** from dormant to fully integrated
- **<2ms latency** for event processing
- Direct IIT ϕ integration for consciousness-aware computing

The 495 SLOC implementation[1] enables

[1] Implementation update (Feb 2026): The cognitive subsystem has since expanded to 39 Python source files (11K LOC) with 28 dedicated test files, incorporating additional cognitive event types, attention mechanisms, and consciousness integration layers. The 495 SLOC figure reflects the core pipeline described in this paper.

ARKHEION to be aware of its operating environment and incorporate that awareness into consciousness calculations.

References

1. Kerrisk, M. (2010). *The Linux Programming Interface*. No Starch Press.
2. Tononi, G. (2008). Consciousness as integrated information. *Biological Bulletin*, 215(3), 216-242.
3. Freedesktop.org. (2024). D-Bus Specification.
4. ARKHEION Documentation. (2026). Integration Module. Internal.