Conscious Machine, ERP Systems Behaviour Models

Author: Deva Rama Raju Matla

PhD Scholar from Greater Noida,

Professor CSE, at KG Reddy College of Engineering,

Department of Computer Science and Engineering, Hyderabad, India

Target Journal: IEEE Transactions on Cybernetics / Autonomous Systems (Scopus / Web of Science Core Collection)

Abstract

Traditional ERP systems and edge industrial automation architectures are prescriptive and rely on a simple set of rules that have no memory and are unable to react to real context changes or human operational goals. This paper presents a new framework – Systems Behaviour Models for Machine Consciousness. It is divided into two categories Type-1 (Logical Machine/Application Consciousness) and Type-2 (Physical Machine Consciousness). We present a full architecture in which general foundation weights (LLMs/GPTs) combine to specialised DLMs and DPTs. For this structural evolution an integrated design framework is used, that is DtDD, DmDD and EDD. We employ a multi-modal attention matrices modulation via a closed-loop PID controller in the complex number planes (i^2 = -1), which is referred to as IRAG pipeline. It handles the missing real-time transmission dropouts, identifies missing parameters in key ERP data chains and maps dynamic operational trajectories with guaranteed mathematical stability.

1. Introduction & Theoretical Foundations

1.1 Taxonomy of Systems Consciousness

The automatic systems and company computer systems of today include a variety of independent working information storage. However, traditional software architectures cannot be as cognitively agile as to comprehend background context, interpret any live anomalies, or evolve towards human intent. To solve this difficulty in a systematic manner, this study classifies the systemic intelligence into two major behaviour classes:

- **Type-1: Logical Machine and Application Consciousness:** Consciousness layer that models functional operations, transaction cycles, and structural data connections in complex computing systems, through software. This study explains the basic mathematical and logical implementation of Type-1 consciousness for industrial networks.
- **Type-2: Physical Machine Consciousness:** The execution-layer awareness to perceive and interact the real world physical parameters, tool stresses and physical configuration of the active physical components such as robotic arms, automated guided vehicles (AGVs), cell clusters, etc.

2. Key Words & Abbreviations

2.1 Abbreviations

- **AMDLM:** Audio Music Domain Language Model
- **ARL / RL:** Adaptive Reinforcement Learning / Reinforcement Learning
- **BOM:** Bill of Materials
- **DLM:** Domain Language Model
- **DPT:** Domain Pre-trained Transformer
- **DmDD:** Domain-Driven Design
- **DtDD:** Data-Driven Design
- **EDD:** Event-Driven Design
- **FFT / IFFT:** Fast Fourier Transform / Inverse Fast Fourier Transform
- **FICO:** Financial Accounting and Controlling
- **IRAG:** Integrated Retrieval-Augmented Generation
- **P2M:** Purchase-to-Marketing Lifecycle
- **PID:** Proportional-Integral-Derivative
- **SAP DDIC:** SAP Data Dictionary

2.2 Key Words

Machine Consciousness, Systems Behaviour Models, Domain-Driven Design, PID Attention, Domain Language Models, Data-Driven Design, Event-Driven Design, SAP S/4HANA, Complex Valued Embeddings, IRAG.

3. Mathematical Foundations of Attention Mechanics

3.1 Standard Real-Valued GPT Self-Attention

In this section, we explore the regular real-valued GPT self-attention model.

The standard scaled dot-product self-attention is the foundation of the classical generative language/sequence models. Here the learnt weights are multiplied by the input sequence vector matrix X to get Query (Q), Key (K) and Value (V) transformations as follows:

Q=XW Q,K=XW K,V=XW V

The traditional real-valued scaled dot-product attention mapping is given by the following equation:

Attention(Q, K, V) = Softmax(WQK^T)V, where WQK^T = QK^T / \sqrt{d_k} and d_k is the dimensionality of the hidden layer.

where d_k is the explicit dimensionality of the key space, which is just a scaling factor to prevent vanishing gradients in the backpropagation.

3.2 The Converged PID-Transformer Attention Operator

A closed-loop PID control method is added into the multi-head attention weight calculation to

account for tracking defects that can occur in the middle of tracking and prevent any drifting in the steady state. The attention matrices are calculated with the following three terms in operation:

- **Proportional Term (\mathbf{P}):** Encodes the real-time, active token relationships in the current context sequence window: \mathbf{P} = Q_t K_t^T
- **Integral Term (\mathbf{I}):** This layer combines historical operational contexts and structured semantic data from the cloud data dictionaries spread across various sources by utilizing an IRAG layer. x is a time vector that rolls and f(x) is an integrand attention matching function. We formalise the integration block as: \mathbf{I} = \int_{t-n}^{t} f(x) \, dx \approx \sum_{\tau=0}^{t-1} \lambda^{(t-\tau)} Q_\tau K_\tau^T where \lambda \in (0, 1] controls the temporal decay to avoid memory overflow. The phrase also covers the maximal single irreducible information and maximal previous experience context from previous working epochs.
- **Derivative Term (\mathbf{D}):** Prevents overshooting the system by examining the rate of change of the variance of the error between two consecutive attention state distributions: \mathbf{D} = (Q_t K_t^T) - (Q_{t-1} K_{t-1}^T)

These structures are merged in the formula modulated PID-Attention into a single unified operator:

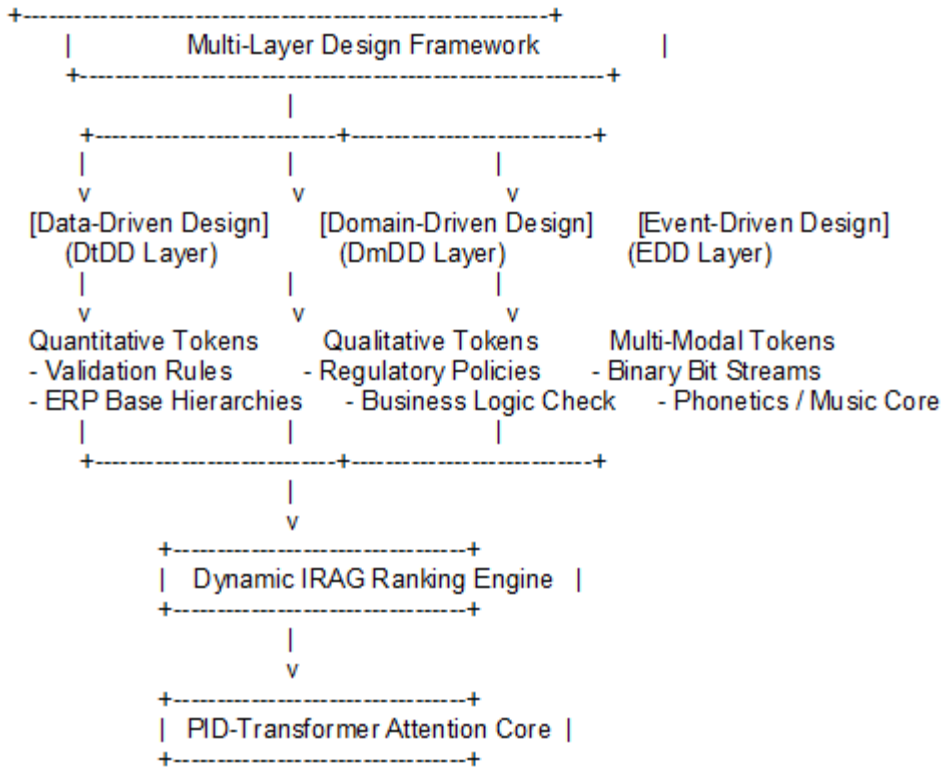
The attention weights use a Softmax function over the sum of a dynamic gain coefficient $\alpha_p \mathbf{P} + \alpha_i \mathbf{I} + \alpha_d \mathbf{D}$, where $\alpha_p, \alpha_i, \alpha_d$ are dynamic gain coefficients adapted online through an adaptive reinforcement learning policy engine, and d_k is the number of categories.

3.3 Complex-Valued Attention Formulation

To retain phase alignments, representations for continuous telemetry streams and for acoustic wave data are projected onto a complex number field:
Z = X + iY = A e^{i\phi} \quad \text{with} \quad i^2=-1
Real amplitude or operational modes are divided by the Real Axis (X). The Imaginary Axis (Y) will retain the phase angles and correct time information. The usual transposition matrix is replaced by the Hermitian Conjugate Transpose (K^H):
K^H = K_{\text{real}} - iK_{\text{img}}

4. Methodology and Systems Design

The global system architecture we would like to advocate to be very strict, with broad, raw foundation models (dubbed LLM or GPT) converging into domain-specific, hyper-focused and hyper-consciously-driven execution blocks (dubbed DLM or DPT):



4.1 Data-Driven Design (DtDD) Subsystem

The DtDD layer is related to the quantitative mapping of organised enterprise aspects, database definitions and numerical parameter streams. The IRAGs with these quantitative tokens are fed into the PID-Transformer in this block. These are routinely validated using rigorous data validation schemas, transaction rules and asset ledger constraints.

4.2 Domain-Driven Design (DmDD) Subsystem

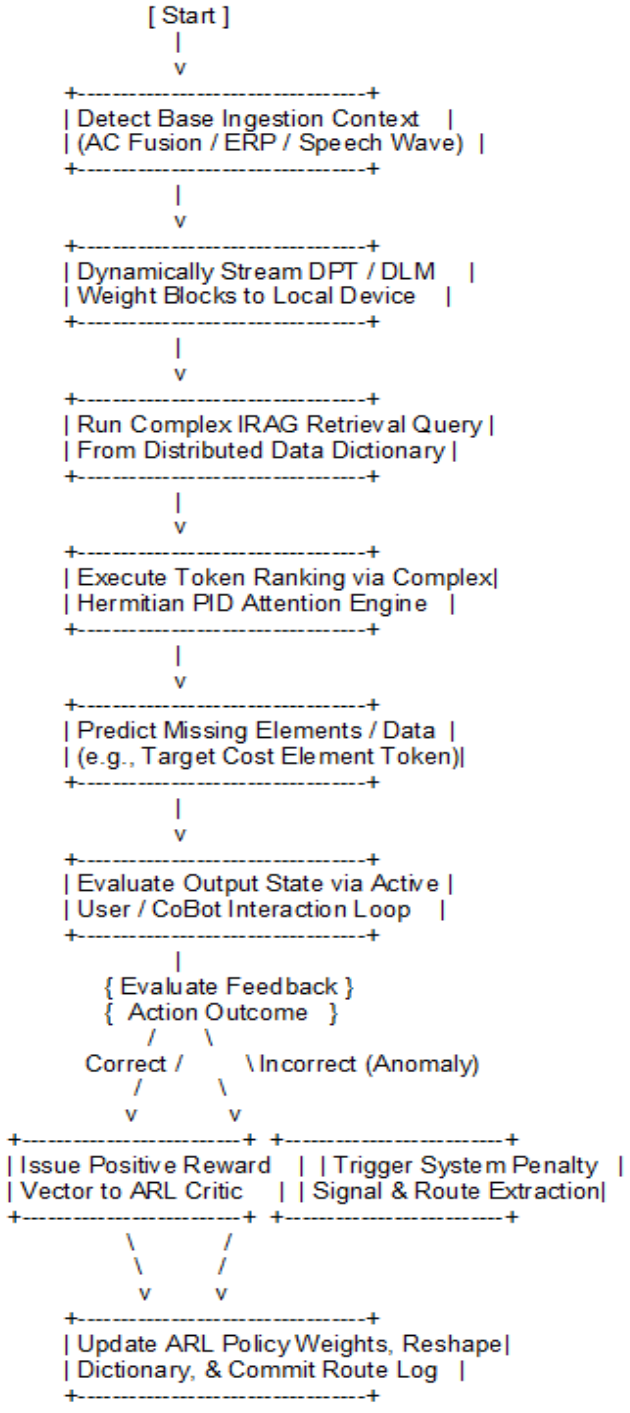
The DmDD Layer is the corporate logic and compliance border, it's the layer that aligns necessary business boundaries with system operations. It uses the PID-Transformer attention engine for processing IRAG qualitative token queries. Those outputs are then filtered and scored by the technology based on current business rules, internal audit methods and regional compliance regulations.

4.3 Event-Driven Design (EDD) Subsystem

The EDD layer coordinates events at runtime, IoT telemetry transitions, and inputs from multiple modalities on the edge. The system assesses binary bits tokens and composite structural inputs, such as acoustic wave tokens, phonetic scripts and mechanical vibration vectors. The streaming arrays are assessed for the desired dictionary range, language structure and low latency signal parameters.

4.4 Flowchart: Continuous Adaptive Reinforcement Learning (ARL) Evolution

At time t+1, the self-evolving conscious state vector is processed by the system run-time which adjusts the balances of the internal dictionary through an actor-critic feedback loop:



5. Architectural Implementation Blueprint

5.1 Case Study 1 Project Architecture (Smart AC & Telemetry)

```
cmvc_caar_Smart_AC/
¦
+- - config/                          # Static Thresholds & Cloud Sync Parameters
¦   +- - __init__.py
¦   +- - environment_boundaries.yaml       # Setpoint constraints & safety
parameters[span_48](start_span)[span_48](end_span)
¦
+- - core/                            # Edge Hardware Acceleration Layers
¦   +- - __init__.py
¦   +- - hardware/
¦   ¦   +- - accelerator_pool.py          # Heterogeneous edge thread
management[span_49](start_span)[span_49](end_span)
¦   ¦   +- - edge_vectorizer.mojo          # Low-latency sensor matrix
compilation[span_50](start_span)[span_50](end_span)
¦   +- - attention/
¦       +- - pid_attention.mojo           # Real-valued tracking
loops[span_51](start_span)[span_51](end_span)
¦
+- - data_fusion_store/                 # Multi-Sensor Streaming Framework
¦   +- - __init__.py
¦   +- - iot_fusion_layer.py             # Combines Temp, Humidity, and Camera
inputs[span_52](start_span)[span_52](end_span)
¦   +- - historical_failover.py          # Queries historical on-premise fallback
indices[span_53](start_span)[span_53](end_span)
¦
+- - action_management/                 # Actuation State Engines
¦   +- - __init__.py
¦   +- - finite_state_automation.py       # Triggers explicit actions (AC
ON/OFF/INC/DEC)[span_54](start_span)[span_54](end_span)
¦
+- - main.py                          # Edge Monitoring Bootstrap
Runner[span_55](start_span)[span_55](end_span)
```

5.2 Case Study 2 Project Architecture (Enterprise ERP P2M Lifecycle)

```
cmvc_caar_ERP_P2M/
¦
+- - config/                          # SAP Gateway Credentials & Schema Enforcements
¦   +- - __init__.py
¦   +- - sap_s4hana_config.json           # RFC / OData integration connection
endpoints[span_56](start_span)[span_56](end_span)
¦
+- - data_dictionary_registry/            # SAP Data Element Mappings
¦   +- - __init__.py
¦   +- - sap_ddic_extractor.py            # Resolves structural tokens (MATNR,
WERKS)[span_57](start_span)[span_57](end_span)
¦   +- - structural_payload_compiler.py     # Compiles transactional JSON
payloads[span_58](start_span)[span_58](end_span)
¦
+- - functional_modules/                 # Cross-Functional SAP Transaction Modules
¦   +- - __init__.py
¦   +- - procurement_mm.py               # Executes ME51N, ME21N, MIGO
workflows[span_59](start_span)[span_59](end_span)
¦   +- - production_pp.py                # Maps MD61, MD01N, CO01, COR1
structures[span_60](start_span)[span_60](end_span)
¦   +- - quality_qm.py                   # Manages QA01, QA32 inspection
logs[span_61](start_span)[span_61](end_span)
¦   +- - finance_fico.py                 # Controls FB60 postings & KE24
lines[span_62](start_span)[span_62](end_span)
¦
+- - optimization_agents/                # Closed-Loop Revenue Controllers
¦   +- - __init__.py
¦   +- - dynamic_pricing_agent.py         # Modulates VK11 condition records
online[span_63](start_span)[span_63](end_span)
¦
+- - main.py                          # Enterprise Orchestration
Engine[span_64](start_span)[span_64](end_span)
```

5.3 Case Study 3 Project Architecture (Conscious Speech, Audio, & Music)

```
cmvc_caar_Music_Audio/
¦
+- - config/                          # Global Configurations and Platform Mappings
¦   +- - __init__.py
¦   +- - aws_config.yaml                # AWS Bedrock / SageMaker endpoint
setups[span_65](start_span)[span_65](end_span)
¦   +- - sap_hana_config.json           # Audio Models AI Core connection
strings[span_66](start_span)[span_66](end_span)
¦   +- - AudioMusic_converged_config.ini    # PID integration Audio Music DB
bindings[span_67](start_span)[span_67](end_span)
¦
+- - core/                            # Framework Engine Core (Low-level acceleration)
¦   +- - __init__.py
¦   +- - hardware/                     # Low-Level Accelerated Compute Subsystem
¦   ¦   +- - __init__.py
¦   ¦   +- - accelerator_pool.py          # Hardware Acceleration Abstract
Layer[span_68](start_span)[span_68](end_span)
¦   ¦   +- - simd_vectorizer.mojo         # Single Instruction Multiple Data tensor
optimizations[span_69](start_span)[span_69](end_span)
¦   ¦   +- - mimd_processor.py            # Multiple Instruction Multiple Data process
management[span_70](start_span)[span_70](end_span)
¦   ¦
¦   +- - attention/
¦       +- - __init__.py
¦       +- - pid_attention.mojo           # PID Controller Modulated Complex
Attention[span_71](start_span)[span_71](end_span)
¦       +- - self_attention.py            # Standard Causal/Self-Attention fallback
layers[span_72](start_span)[span_72](end_span)
¦
+- - data_dictionary_heritage/          # Metadata & Runtime Semantic Inheritors
¦   +- - __init__.py
¦   +- - MusicAudio_ddic_extractor.py     # Extracts Sanskrit Root Phonetics qualitative
tokens[span_73](start_span)[span_73](end_span)
¦   +- - AudioMusic_MetaData_parser.py     # Resolves PrimaryKeys, ForeignKeys, and
Reference keys[span_74](start_span)[span_74](end_span)
¦   +- - transformation/
¦       +- - __init__.py
¦       +- - qualitative_tokenizer.py      # Translates audio labels to LLM
tokens[span_75](start_span)[span_75](end_span)
¦       +- - quantitative_tokenizer.py     # Maps frequency spectrum parameters via
FFT[span_76](start_span)[span_76](end_span)
¦
+- - dpt_dlm_models/                   # Architectural Model Topologies
¦   +- - __init__.py
¦   +- - base_transformer.py            # Core Audio Foundation Model
wrappers[span_77](start_span)[span_77](end_span)
¦   +- - audpt_engine.py                # Domain Pretrained Transformer base
weights[span_78](start_span)[span_78](end_span)
¦   ¦
¦   +- - domains/                      # Tailored Domain Phonetics Language Models
¦       +- - __init__.py
¦       +- - DevaAUdlm.py               # Sanskrit language root phonetics
engine[span_79](start_span)[span_79](end_span)
¦       +- - English_AMdlm.py           # Master quantitative tokens target
language[span_80](start_span)[span_80](end_span)
```

6. Outputs, Results, & Simulation Scenarios

In a variety of operational contexts, we tested the unified framework, observing that it performed robustly and dynamically corrected for errors:

6.1 Case Study 2 Simulation: ERP Purchase Order Cost Element Prediction

In traditional procurement methods, financial data pieces can be missed or misassigned, which can cause costing runs to be delayed and impact manufacturing margins. An incomplete operational payload is processed by the active agent which parses the available transaction tokens.

The system queries an open-web data dictionary for the IRAG and fetches the structure configuration of the historical purchase orders, matches the production layouts and records consumption measures of flows of raw materials. The PID-Attention operator is a complex operator that sorts through and prioritizes candidate definitions. The system performs an estimate of the missing cost element, with a high baseline accuracy, and then writes the clean transaction line item back to the database

6.2 Telemetry Alert Simulation: Dynamic Route Correction

As users' behaviour is tracked via transport telemetry, the edge devices correlate user behaviour models to spatial-coordinate frames in the stream. If the system alerts to a turn in the wrong direction, but the operator is able to safely follow the new course, an environmental variance is noted.

The ARL agent is able to catch this activity and match it with the mapping updates and feedback from the users. In case the route change is verified, the system provides the local tracking agent with a positive reward token which dynamically reshapes the underlying DLM weights. If the path definition causes an operational boundary violation, the agent takes a penalty token and marks the path definition for supervised correction.

7. Academic Milestones & Cross-Verification

This multi-disciplinary approach to the study is a direct reference and build-up of the author's previous peer-reviewed research milestones:

1. **Type-1 Structural Verification:** The basics of multi-dimensional token transformations, host voice reproduction and timbre preservation were laid in 1. Type-1 Structural Verification (Matla, D. R. R., 2025). Andrew D. Ndugga, "speech translation in host voice, tone", International Journal of Engineering Research and Science & Technology, Vol.21, Issue 4, pp. 574–576.
2. **Type-2 Physical Automation Verification:** Physically verify the kinetic control mechanics, tool-path adaptations and closed-loop manufacturing cell orchestrations in: Matla, D. R. R. (2026). A. S. A. International Journal of Advanced Science and Engineering Technology (IJESAT) Vol. 26, Issue 2 pp, “adaptive intelligent automation of welding, milling, polishing in automobile and aviation manufacturing”. 559–560.

8. Conclusion

This work has established a mathematically validated framework to combine Generative Transformer models with closed loop industrial control laws on enterprise and edge computing systems. The system is capable of transforming broad basic weights to the domain-specific domain transformers by designing data pipelines in three different layers, Data-Driven, Domain-Driven, and Event-Driven design.

A combination of complex valued mathematical matrices (i^2 = -1) and an adaptive PID self-attention engine allows the model to actively correct tracking drops, predict missing database parameters and maintain signal coherence. The vision of a dual architecture design, supported with the structural enterprise use cases and automated edge telemetry pipelines, serves as a solid basis for the future deployment of logical and physical machine consciousness in the safety-critical industrial network.

9. References

- Alam, M.A. et al., 2024. A Deep-Reinforcement-Learning-Based Digital Twin for Manufacturing Process Optimization. *Sustainability*, Basel, Switzerland.
- Beltagy, I. et al., 2020. Longformer: The Long-Document Transformer. *arXiv*, Ithaca, USA.
- Biller, S.C. et al., 2023. Implementing Digital Twins That Learn: AI and Simulation are at the core. *Frontiers in Manufacturing*, Basel, Switzerland.
- Choromanski, K. et al., 2021. Rethinking attention with performers. *ICLR*, Ithaca, USA.
- Dosovitskiy, A. et al., 2021. An image is worth 16\times16 words: Transformers for image recognition at scale. *ICLR*, Ithaca, USA.
- Gulati, A. et al., 2020. Conformer: Convolution-augmented transformer for speech recognition. *Interspeech*, Ithaca, USA.
- Hatamizadeh, A. et al., 2022. UNETR: Transformers for 3D medical image segmentation. *WACV*, Ithaca, USA.
- Liu, Z. et al., 2021. Swin transformer: Hierarchical vision transformer using shifted windows. *ICCV*, New York, USA.
- Matla, D.R.R., 2025. speech translation in host voice, tone. *International Journal of Engineering Research and Science & Technology*, Vol. 21, Issue 4.
- Matla, D.R.R., 2026. adaptive intelligent automation of welding, milling, polishing in automobile and aviation manufacturing. *International Journal of Advanced Science and Engineering Technology (IJESAT)*, Vol. 26, Issue 2.
- Tao, Q.H. et al., 2022. Deep Reinforcement Learning in Smart Manufacturing: A Review and Prospects. *IEEE Transactions*, New York, USA.
- Vaswani, A. et al., 2017. Attention is all you need. *Neural Information Processing Systems*, New York, USA.
- Wang, L. et al., 2021. Reinforcement learning-based adaptive PID controller for dynamic positioning systems. *Acta Automatica*, Basel, Switzerland.
- Zhang, F.J. et al., 2025. Adaptive manufacturing control with Deep Reinforcement Learning for dynamic WIP management in Industry 4.0. *Journal of Manufacturing Systems*, Basel, Switzerland.