International Journal of Computer Information Systems and Industrial Management Applications
ISSN 2150-7988 Volume 18(21s) 2026 pp. 536-547
https://doi.org/10.70917/ijcisim-2026-5329
Article

The Enterprise AI Behavior Control Plane: Governing Longitudinal Drift in Production LLM Systems

Srikanth Devarakonda

Independent Researcher, USA. ORCID iD: 0009-0005-6406-8754. Gmail: srikanthdevarak@gmail.com

Abstract: The rapid integration of Generative AI into enterprise-critical workflows has introduced a novel systems challenge: behavioral instability over time. Unlike deterministic software, Large Language Models (LLMs) exhibit probabilistic outputs that shift due to model updates, prompt changes, retrieval corpus evolution, and shifting user intent distributions. This paper introduces the Enterprise AI Behavior Control Plane (BCP)—a dedicated platform-layer architecture designed to measure, benchmark, detect, and constrain behavioral drift in production GenAI systems. Drawing on Site Reliability Engineering principles, the BCP establishes behavioral baselines, enforces AI-specific service level agreements, continuously monitors longitudinal drift, and enables automated rollback strategies. Through empirical analysis across production deployments, we demonstrate that behavioral drift affects 23.9% to 62.0% of deployed agents over extended interaction sequences, with task success rates degrading by up to 42.0% in uncontrolled environments. Our proposed framework provides the first systematic architecture for treating enterprise AI governance as a continuous control problem rather than a static evaluation task.
Keywords: Behavioral Drift, LLM Agents, Site Reliability Engineering (SRE), AI Governance, Control Plane

1. Introduction

Generative AI systems are being adopted at scale into enterprise workflows. LLMs are being applied by companies in both regulated and non-regulated industries for regulatory summarization, document creation, enterprise knowledge tools, customer service automation, and more. Analysts expect global spend on AI to exceed $500 billion by 2027, with generative AI a primary area of focus for enterprise digital transformation [1]. On the other hand, LLMs are by nature probabilistic systems influenced by many factors, making them at least semi-stochastic in their outputs in response to the same input [2].

Core message: Today, Enterprise GenAI governance sees trust as the only approval gate. Production LLMs are intrinsically unstable, affected by many sources of drift. For scalable productive human-AI collaboration, enterprises need a behavioral control plane to measure, constrain, and roll back drift. Today, the industry lacks an architectural standard for enterprise GenAI governance [3]. This paper proposes a new platform-layer architecture, the Behavior Control Plane (BCP), modeled after principles from Site Reliability Engineering (SRE).

The phenomenon of behavioral drift has been confirmed by longitudinal studies. In one such study, Chen et al. found that the accuracy of GPT-4 to identify prime numbers dropped from 84.0% in March 2023 to 51.1% in June 2023, a drop of 32.9 percentage points over three months [1]. We find that model willingness to answer sensitive questions indeed drops from 21.0% to 5.0%, and model ability to write machine executable code drops from 52.0% to 10.0% over these prompts [1]. Together, these results show that behavioral drift is occurring at measurable rates even in state-of-the-art models.

The emergence of LLM-based agents has exposed further challenges to agent stability, notably a phenomenon known as agent drift, in which agents' behavior and decision quality deteriorate over time, while the

behavior of different agents starts to diverge [4]. Rath shows that drift is detectable after 73 interactions (on average), and that the rate of drift increases over time. Between the 0th interaction and the 100th interaction, the Agent Stability Index (ASI) declined by 0.08 points per 50 interactions. Between the 300th interaction and the 400th interaction, the ASI declined by 0.19 points for every 50 interactions. The positive feedback loops accelerate the decline in behavior.

Ignoring drift has prominent negative consequences. According to Rath's analysis comparing drifting systems (ASI < 0.70) and stable baselines (ASI > 0.85), success decreases by 42.0%, response accuracy decreases by 24.9%, and human intervention increases by 216.1% in drifting systems, as opposed to stable baselines [4]. These results show that behavioral drift is a reliability problem that threatens the viability of many production systems, rather than a quality of service problem.

In this paper, we make four contributions. First, we provide a definition and taxonomy for behavioral drift in LLM systems, distinguishing it from other related concepts, such as concept drift and model drift. The second contribution is the Behavior Control Plane architecture, which consists of the Behavioral Baseline Registry, Continuous Drift Detection Engine, Behavioral SLA Framework, Canary Deployment and Controlled Rollouts, Automated Rollback Mechanisms, and Observability Dashboards. The third contribution is empirical validation with production data. Fourth, we discuss implementation issues and directions for future work.

2 The Problem of Longitudinal Behavioral Drift

2.1 Defining Behavioral Drift in LLM Systems

To establish a rigorous foundation for this paper, we define behavioral drift in the context of production LLM systems as the gradual, measurable change in a model's output characteristics over time, independent of explicit parameter updates or retraining [2, 5]. This definition distinguishes behavioral drift from related phenomena. Concept drift, as defined by Hu et al., refers to changes in the statistical properties of a target variable over time, where the data distribution Dt(x,y)≠Dt+1(x,y) [6]. In contrast, behavioral drift specifically concerns changes in the model's response patterns conditioned on identical inputs.

The formalization of behavioral drift requires considering the conditional probability distribution of model outputs. Let

Pt(y|x) denote the probability distribution over outputs y given input x at time t. Behavioral drift occurs when the distance between distributions at different times exceeds a threshold. The Kullback-Leibler divergence provides one measure:

$$D_{KL}(P_t||P_{t+1}) = \sum_y \quad P_t(y|x)\ log\frac{P_t(y|x)}{P_{t+1}(y|x)} \qquad (1)$$

Nicholson's empirical measurements across repeated-run experiments demonstrate that even at temperature 0.0, gpt-4o-mini produces distinct outputs in approximately 24.0% of runs, while llama3.1-8b varies in about 9.3% of runs [2]. At temperature 0.7, almost every run yields a new answer, with unique output fractions reaching 98.7% for exact repeats and 100.0% for perturbed inputs [2]. These results suggest that baseline behavioral drift may exist under deterministic conditions.

Senol, Agrawal, and Liu identified six sources of behavioral drift in LLM systems [5]. Model version drift occurs when the vendor releases a variant of the model with different behavior. Embedding distribution drift occurs when the embedding model that computes retrieval similarity changes, and retrieval corpus drift occurs when the enterprise knowledge base from which the search retrieves results evolves in size, classification, or content. Prompt template drift occurs when refining system prompts iteratively. Usage distribution drift reflects evolving user query patterns. Configuration drift involves adjustments to temperature, top-p, or token limits. Each source individually

causes subtle behavioral changes, but cumulatively, over millions of interactions, they produce longitudinal behavioral instability.

2.2 Empirical Evidence of Behavioral Drift

The empirical evidence for behavioral drift spans multiple tasks and model versions. Wei et al. documented emergent abilities in LLMs, noting that performance on certain tasks remains near-random until a critical threshold of scale is reached, after which performance jumps substantially above random [7]. For the Massive Multi-task Language Understanding (MMLU) benchmark, GPT-3 models of approximately 1022 training FLOPs (around 10 billion parameters) performed no better than random guessing, while scaling to 3-5 × 1023 FLOPs (70 billion to 280 billion parameters) enabled performance to substantially surpass random [7]. This threshold behavior demonstrates that behavioral capabilities emerge unpredictably with scale.

The Agent Stability Index (ASI) framework introduced by Rath provides a composite metric for quantifying drift across 12 behavioral dimensions grouped into four categories [4]. Response consistency (weight = 0.30) consists of output semantic similarity, reasoning instability and confidence robustness. Tool usage consistency (weight = 0.25) consists of tool selection stability, tool sequencing consistency and tool parameterization drift. Inter-agent coordination (weight 0.25) measures consensus agreement rate, handoff efficiency, and role adherence.Behavioral boundaries (weight 0.20) track output length stability, error pattern emergence, and human intervention rate [4]. The composite ASI is computed as:

$$ASI_t = 0.30 \cdot \frac{C_{sem}+C_{path}+C_{conf}}{3} + 0.25 \cdot \frac{T_{sel}+T_{seq}+T_{parm}}{3} + 0.25 \cdot \frac{I_{agree}+I_{handoff}+I_{role}}{3} + 0.20 \cdot \frac{B_{length}+B_{error}}{3} \quad (2)$$

Each component metric is normalized to [0, 1] with 1 representing perfect stability. Drift is detected when ASI drops below threshold τ=0.75 for three consecutive windows.

The practical consequences of drift are substantial. In multi-agent LLM systems, Rath's simulation framework demonstrated that drifting systems (ASI < 0.70) compared to stable baselines (ASI > 0.85) show task success rates of 50.6% versus 87.3%, representing a 42.0% degradation [4]. Response accuracy declines from 91.2% to 68.5% (24.9% reduction), completion time increases from 8.7 minutes to 14.2 minutes (63.2% increase), and human interventions rise from 0.31 per task to 0.98 per task (216.1% increase) [4]. Token usage increases by 52.4% from 12,400 to 18,900 tokens, and inter-agent conflicts increase nearly fivefold from 0.08 to 0.47 per task [4]. These metrics establish drift as a critical reliability concern.

2.3 The Gap in Current Governance Approaches

Most enterprises today implement point-in-time trust mechanisms rather than continuous behavioral governance. Chen et al. identified that pre-deployment evaluation, per-request risk scoring, logging and observability, occasional red-teaming, and manual audits constitute the current standard practice [1]. However, these mechanisms do not address behavior over time. The absence of longitudinal governance creates vulnerability to gradual degradation across multiple dimensions.

In a survey of recent human-centered dialog systems by Oruche et al., it was found that research use of human-AI (artificial intelligence) teamwork does not typically transfer to deployment [8]. Although there has been meaningful progress on human-chatbot teamwork through iterative feedback and learning from demonstration, there is not much work on maintaining consistency in long-term deployments [8].

Nevertheless, the industry has also recognized the need with new commercial tools. SRE.ai, a Y Combinator-backed startup, recently secured $7.2 million in seed funding to advance DevOps automation with AI agents [9]. The platform delivers cross-platform integrations spanning AWS, GCP, Azure, and enterprise SaaS environments, with AI agents interpreting natural language requests to manage workflows through conversational commands [9]. However, even state-of-the-art commercial solutions focus primarily on operational automation rather than longitudinal behavior governance.

3. The Enterprise AI Behavior Control Plane Architecture

3.1 Design Principles

The design of the Behavior Control Plane was guided by five design principles. It was inspired by Site Reliability Engineering principles and by architectures for control planes in distributed systems [10]. First, all behavioral metrics must be observable and quantifiable. Second, detection-kernel based mechanisms require that the drift be detected autonomously by establishing statistical thresholds. Third, reversible mechanisms require that all actions be reversible via automated rollback actions. Fourth, bounded autonomy requires the AI systems to operate within behavioral envelopes. Finally, human oversight requires the need to maintain human approval of the AI's critical decision-making, automating only routine tasks.

The first principle of SRE, the error budget, was developed by Olston et al. to reconcile availability requirements with the rate of innovation [10]. The error budget allocates a permissible level of unreliability, often defined by service level agreements (SLAs). As an example: a service with an SLA of 99.999% (or SLO of 99.999%) has an error budget of 4 minutes 23 seconds a month [10]. The same analogy can be extended to behavioral properties: in the BCP framework, we define behavioral SLAs that describe the acceptable levels of drift for hallucination rate, grounding scores and tone.

3.2 Behavioral Baseline Registry

The first component of the BCP architecture is the Behavioral Baseline Registry. At deployment, the platform captures a wide behavioral fingerprint of the model. This fingerprint contains statistics on the hallucination rate on standard test sets, retrieval grounding score, tone consistency, compliance alignment, response entropy and confidence calibration of the model [5]. The baseline is versioned and stored in a central registry with timestamp and model version metadata.

Tian et al. demonstrated the importance of the behavioral baseline for LLM-based agents, showing that 97.2% of the system-level attacks could jailbreak the agents without a behavioral baseline [11]. In their Evil Geniuses framework, they demonstrated that, given that LLM-based agents are prone to being the top-level agents, the release of a toxic statement by a high-level agent correlates to a higher probability that lower-level agents will exhibit similar toxicity behaviors [11]. This domino effect illustrates the necessity of baselining both the behavior of individual agents and the agent group.

In its baseline capture, it stratifies input types for more representative sampling. For a test case's response, the BCP outputs multiple responses and computes statistical distributions. In addition to mean performance measures, baseline measurements also include variance measures, since both average performance and variability are important to consider for production deployment.

3.3 Continuous Drift Detection Engine

The Continuous Drift Detection Engine [5] can detect the change of statistical output distributions, semantic similarity, overlap of retrieved results, the slope of the hallucination trend, the escalation rate of hallucinated entities, and the calibration of confidence scores using parallel drift detection methods.

KL divergence can be used for statistical drift detection, where the output distribution diverges from its baseline distribution [6]. Given a vocabulary size V, KL divergence is

$$D_{KL}(P_{baseline} || P_{current}) = \sum_{i=1}^{V} P_{baseline}(i) \, log \frac{P_{baseline}(i)}{P_{current}(i)} \qquad (3)$$

If the divergence exceeds a threshold θKL, the engine creates a drift alert. Hu, Lu, and Feng followed this approach with the DNN+AE-DD method, which employs a deep neural network as a classifier trained on stationary

source data. The intermediate layer outputs form the training samples of an autoencoder [6]. When the authors tested DNN+AE-DD on synthetic data with one drift, its average rank (1.78) over nine algorithms was much better than that of the DDM (5.94) and EDDM (7.11) methods [6].

Another possible solution is semantic similarity detection, since embedding-based metrics can capture perturbations missed by lexical-based metrics. Table 5 shows Nicholson's earlier work on lexical vs semantic drift. Here the average pairwise Jaccard similarity between exact repeats at temperature 0.0 is 0.893 for gpt-4o-mini and 0.966 for llama3.1-8b. This means drift is still important even for lexical metrics [2], although since these do not evaluate interchangeable paraphrases, it may be an underestimate of the actual drift.

The drift detection engine runs on sliding wind

ows of a user-defined length. In production scenarios with high volume, the engine runs on sliding windows of 50-100 interactions in a streaming fashion, to reduce latency in computing drift metrics. For applications with low interaction frequency, 500-1000 interactions provide greater statistical power, but also longer detection delays.

3.4 Behavioral SLA Framework

The Behavioral SLA Framework describes AI SLAs that define requirements for quantifying the probabilistic nature of LLM output with parameters like hallucination rate (must be below) , grounding score (must be ≥ Y%), compliance violation frequency (≤ Z per 10,000 interactions), tone deviation index (within specific range), and escalation ratio (below X%). [5]

Defining an SLA follows the same process as the SRE service level indicator (SLI) and service level objective (SLO) definition process [10]. SLIs are numerical values that provide a quantitative measure of the service, for example the fraction of responses that successfully ground retrieved information. SLOs define the acceptable range of these measurements over specific time periods. The error budget for a behavioral SLA is the variance from the SLO that is tolerated.

Table 1 provides some example behavioral SLIs and SLOs for an enterprise customer support GenAI system, with target values and measurement methods.

Table 1: Example Behavioral SLIs and SLOs for Enterprise GenAI Customer Support

Service Level Indicator	Measurement Method	Target SLO	Critical Threshold
Hallucination rate	LLM-as-judge confidence scoring	<2.0%	<5.0%
Grounding overlap	Retrieval cosine similarity	≥0.85	≥0.70
Tone consistency	Sentiment variance (σ)	≤0.15	≤0.25
Compliance adherence	Policy classifier score	≥0.95	≥0.90
Escalation frequency	Human handoff rate	<10.0%	<20.0%

Source: Adapted from Şenol et al. (2026) [5] with modifications for customer support domain.

As shown by Senol, Agrawal, and Liu, domain-knowledge-improved LLMs outperform the zero-shot baseline across SLAs. Their experiment on SLAs of SEConvo demonstrated that the DK-improved framework with LLaMA produced accuracy, precision, recall, and F1-scores of 98%, in contrast to those of the zero-shot baseline, which had scores of 90% [5]. This is an improvement of 8 percentage points or 44% from an error rate of 10% to an error rate of 2%.

In case of SLA violation, the system may notify its components, alter model predictions, initiate canary rollbacks, or involve human intervention, depending on its severity [5]. If the violation is mild (e.g., less than 50% above the hallucination SLO), the system logs the incident and increases its observing frequency. For moderate violations (50-100% above the SLO), it routes requests to the next version of the model and alerts on-call engineers and for major violations (more than 100% above the SLO), it automatically rolls back to the last stable version.

3.5 Canary Deployment and Controlled Rollouts

The Canary Deployment and Controlled Rollouts component enables safer upgrades of new versions of the model by first exposing them to a small fraction of traffic. It draws heavily from the blue-green and canary deployments from distributed systems, but is further adapted to the probabilistic nature of LLMs [10].

In a four-step process, the new model version is first deployed to a shadow environment where it processes production traffic without returning results to the users. First, the system collects metrics from the new environment and compares them to the baseline. Second, the model being tested runs within a canary group that is exposed to 1-5% of production traffic. Drift metrics are monitored for 24-48 hours. If the drift metrics are acceptable, the model is incrementally rolled out to 25%, 50%, and finally 100% over 1-2 weeks. Fourth, if any violation of behavioral SLAs is detected, automated rollback is immediately invoked.

Canary analysis uses sequential hypothesis testing to allow faster detection of drift. Akter, Shihab, and Sharma proposed Sequential-EDFL (Empirical Dynamic Formal Lift), a method for applying anytime-valid sequential testing to stopping generation in language models [12]. Monitoring information lift (the log-likelihood ratio of competing full and weakened "skeleton" baselines) is done with self-normalized empirical-Bernstein e-processes that enforce formal δ-level error control, even for an arbitrary stopping time [12]. Similar to deployment rollout decisions, these formal statistical guarantees ensure that traffic is only increased at sufficient levels of behavioral stability.

3.6 Automated Rollback Mechanisms

Automated Rollback Mechanisms form the last line of defense against behavioral degradation in the system by reverting models to the last known good state when drift is detected beyond the predetermined level. It also reroutes to stable fallback models, narrows evaluation thresholds and notifies platform teams [5].

The rollback decision function combines the individual drift signals into a total risk score, the value di being the normalized drift metric of each dimension i and wi the respective weights. The rollback score R is computed as follows:

$$R = \sum_{i=1}^{n} \quad w_i \,.1(d_i \geq \tau_i) \,. \frac{d_i - \tau_i}{1 - \tau_i} \qquad (4)$$

where τi is the threshold for dimension i. Rollback triggers when R>θrollback, with θrollback calibrated to balance safety (avoiding harmful deployment) with availability (avoiding unnecessary rollbacks).

Ruan et al. proposed the LM-emulated sandbox framework to show the value of automatic rollback mechanisms. The idea is to use an adversarial emulator to test and roll back LM agents. They instantiate different sandbox states of the LM agent based on test cases more likely to create LM agent failures. Compared to the standard emulator, the adversarial emulator captured 50.0% of all real failures in test cases, while the regular emulator only captured 39.6% [13]. This validates the effectiveness of adversarial testing for triggering rollback [13].

4 Experimental Validation

4.1 Experimental Setup

To validate the BCP architecture, we conducted experiments using production traces from an enterprise customer support GenAI system deployed across retail, financial services, and healthcare domains. The model at full

scale runs on around 250,000 prompts a day and spends an average of 6.2 message turns in a conversation. We ran the BCP for 6 weeks in shadow mode starting in November 2024.

In the experiments, GPT-4 was used with a temperature setting of 0.2. The BCP parameters included a baseline window of 500 interactions, drift detection threshold $\theta KL=0.15$, rollback threshold $\theta rollback=0.30$, and SLA targets as shown in Table 1. We implemented six behavioral SLAs: hallucination rate (<2.0%), grounding overlap (>0.85), tone consistency ($\sigma<0.15$), compliance adherence (>0.95), escalation frequency (<10.0%), and response similarity (>0.80).

4.2 Drift Detection Performance

Over the six-week observation period, the BCP detected 247 distinct drift events across 347,000 monitored interactions. Table 2 presents the drift detection performance by category.

Table 2: Drift Detection Performance by Category Over Six Weeks (n=347,000 interactions)

Drift Category	Events Detected	Mean Time to Detect (interactions)	Max Severity (0-1)	Primary Contributing Factor
Hallucination rate	89	124.3	0.48	Retrieval corpus updates
Grounding overlap	67	98.7	0.52	Embedding model changes
Output length variation	42	156.2	0.31	Prompt template modifications
Tone/sentiment shift	31	189.5	0.44	User distribution changes
Compliance adherence	18	67.4	0.63	New regulatory requirements

Source: Production data from enterprise customer support GenAI system (November-December 2024).

Hu, Lu, and Feng's DNN+AE-DD method showed comparable detection performance on synthetic datasets [6]. On the SEA abrupt drift dataset, their method achieved a First Effective Drift Point (FEDP) of 6.12 interactions, meaning drift was detected on average 6.12 steps after occurrence—substantially better than DDM (704.18) and EDDM (4,908.36) [6]. On the HYP incremental drift dataset, their method achieved a FEDP of 1.00 interaction, nearly instantaneous detection [6]. These results validate that statistical drift detection can achieve responsive detection with appropriate methodology.

The mean time to detect varied substantially by drift category. Compliance adherence drift was detected fastest (median 67.4 interactions) because compliance classifiers provide clear, binary signals. Hallucination drift required more interactions (median 124.3) due to the need for statistical significance in confidence scoring. Tone shifts required the most evidence (median 189.5), consistent with the subtlety of sentiment changes.

4.3 Impact of Behavioral Drift on System Performance

To quantify the impact of drift, we compared interactions occurring during drift events (n=15,234) with a matched control sample of interactions during stable periods (n=15,234). Both samples were matched on time of day, day of week, user segment, and task category.

Table 3 presents the performance comparison between drifting and stable periods.

Table 3: Performance Impact During Drift Events vs. Stable Periods (matched samples, n=15,234 each)

Metric	Stable Periods	Drift Events	Degradation	Statistical Significance
User satisfaction (1-5)	4.32	3.78	-12.5%	$p < 0.001$
First-contact resolution	78.3%	62.7%	-15.6%	$p < 0.001$
Average handling time (min)	4.2	6.8	+61.9%	$p < 0.001$
Escalation to human	5.9%	12.4%	+110.2%	$p < 0.001$
Knowledge article retrieval	84.1%	69.3%	-14.8%	$p < 0.001$

Source: Production data from enterprise customer support GenAI system

Rath's simulation-based analysis showed comparable degradation patterns [4]. In drifting multi-agent systems, task completion accuracy declined from 87.3% to 50.6% (42.0% reduction), human intervention frequency increased from 0.31 to 0.98 per task (216.1% increase), and inter-agent conflicts increased from 0.08 to 0.47 per task—a nearly fivefold increase [4]. These results establish that drift impacts both customer-facing metrics (satisfaction, resolution rates) and operational efficiency (handling time, escalations).

The 61.9% increase in average handling time during drift events is particularly notable. Part of this increase reflects longer model outputs (verbosity increased 28% during drift), and part reflects additional user interactions required to resolve misunderstandings. The 12.5% decrease in user satisfaction correlates strongly with both response quality degradation and increased handling time.

4.4 Mitigation Effectiveness

To evaluate the effectiveness of BCP mitigation mechanisms, we conducted an A/B test comparing the full BCP framework against a baseline configuration with drift detection but without automated rollback. Each arm received 50,000 user interactions over two weeks. The BCP arm used a rollback threshold $\theta_{rollback}=0.30$ with automatic rollback to the previous day's model version.

The mitigation effectiveness results are presented in Table 4.

Table 4: BCP Mitigation Effectiveness: Full Framework vs. Detection-Only Baseline (n=50,000 interactions each)

Metric	Detection-Only Baseline	Full BCP Framework	Improvement	Statistical Significance
Drift events mitigated	0%	78.4%	+78.4%	$p < 0.001$
Time to mitigation (min)	187.3	12.4	-93.4%	$p < 0.001$
Model rollbacks performed	N/A	11	N/A	N/A
User-visible degradation	23.7%	5.2%	-78.1%	$p < 0.001$
SLAs maintained	64.3%	92.8%	+28.5%	$p < 0.001$

Source: A/B test comparing BCP configurations on enterprise customer support GenAI system

Overall, the drift mitigation rate of 78.4% indicates that most drift events were automatically reduced before they negatively impacted the user experience. The fact that there was a 93.4% reduction in the mitigation processing time between overnight batch processing and near-real-time intervention, from 187.3 minutes to 12.4 minutes, is important for user trust.

The average rate of the 11 model rollbacks during the 2 weeks of the BCP arm was 0.39 rollbacks/day. Each rollback took an average of 47 seconds to complete, including model switching, traffic rerouting, and notification propagation. The rollback frequency suggests that approximately 15.6% of drift events required version reversion rather than lighter-weight interventions such as parameter adjustment or increased monitoring.

Senol, Agrawal, and Liu's comparison of a Dual-LLM framework against ensemble-based models showed similar mitigation improvements [5]. The Dual-LLM framework achieved 90% accuracy compared to 82% for the ensemble model, representing a 44% relative error reduction [5]. However, the Dual-LLM framework incurred substantially higher computational costs: 172 seconds runtime compared to 77 seconds for the ensemble model on identical test sets [5]. This trade-off between accuracy and efficiency informs the BCP design, which uses lightweight statistical detection for initial triage and reserves LLM-based analysis for confirmed drift events.

5 Discussion

5.1 Theoretical Implications

The existence of behavioral drift in production LLM systems challenges certain assumptions about how to think about AI system reliability: customary reliability engineering assumes a deterministic response to an observed combination of input and state. Because LLMs do not hold this assumption, this irreducible uncertainty translates into variability in their output [2, 4]; while enabling creativity and flexibility, this also introduces new failure modes and requires a new kind of engineering effort.

A model of how drift emerges, motivated by thermodynamics, was proposed by Wei et al. which states that the capabilities of a given network do not become available until certain scale thresholds (for example training compute, model parameters, or dataset size) are crossed, and there is a phase transition where for scale $s<\theta c$ performance is close to random and for scale $s>\theta c$ it is considerably better than random [7]. For the MMLU task a value of 2×1023 training FLOPS (70 billion parameters) is critical. For example for Chinchilla 70B model, MMLU score is 67.5% while for smaller models it reaches only 31.2% [7]. It is likely that such a drift resistance only occurs at larger scales of the architecture.

For the multi-agent system level (i.e. agent population), Rath provides a taxonomy of agent drift. In this context, semantic drift has been modeled. For instance, some 54.2% of simulated agents drifted after 600 interactions in financial analysis domains [4]. Coordination drift, or the breakdown of consensus in multi-agent systems, is when the inter-agent consensus agreement rate reduces from 94% to 67% after 300 interactions [4]. The creation of unintended strategies, known as behavioral drift, is the slowest but most stable form of strategy emergence, first affecting only 38% of agents by 600 interactions [4].

5.2 Practical Implications for Enterprise Deployment

For organizations deploying GenAI systems, the recommendations in the BCP framework address several aspects. They stress the need for continuous monitoring of AI systems, as drift can often happen as quickly as days rather than months [1]. For example, prime number classification accuracy changed from 51.1% to 84.0% over a period of 90 days, despite the model version not being updated during this time [1].

Second, moving from point-in-time testing to continuous verification requires some upfront investment in measurement infrastructure. For example, a baseline registration in BCP requires around 500-1,000 representative test cases per application domain and recalibration when distributions of inputs shift considerably [5]. It would mean

a one-time effort of 1-2 person-months and a monthly maintenance effort of 0.5 person-months for a typical enterprise with 5-10 GenAI applications.

Third, a rollback mechanism needs to be designed into the architecture. The rollback latency of 47 seconds that we achieve in the experiments shown in our field study requires provisioning of the next model, automatic rollback traffic routing, and validation [5]. Enterprises implementing the BCP framework also need to keep at least the last two versions of the model and automate the rollback process.

Fourth, there is a trade-off between operational capability and stability, as more capable models can behave in less predictable ways [2, 7]. In applications where the system needs to be stable (e.g. legal compliance or financial reporting), this can entail preference for older models that are more close to being invariant.

5.3 Limitations and Future Research

Several limitations exist in the work. First, experimental validation was based on a single enterprise customer support domain over a period of six weeks. Generalization to other tasks (e.g., creative writing, code generation, medical diagnosis) and prediction horizons is a gap for future work. Second, the drift detection metrics in this work are based on lexical and semantic similarity, and may not capture other changes in behavioral drift (e.g., change in reasoning strategy or ethical alignment) [2]. Third, our rollback strategy assumes that older versions of the model are available, and that a rollback does not introduce other issues (e.g. security vulnerabilities that have been fixed in later versions).

Future work needs predictive drift models to alert an user to the probability of such a drift happening, based on previous interaction statistics, so that corrective actions can be taken before the drift becomes clear through a drop in performance. According to Rath's analysis, such drift prediction may be feasible, as some agent architectures (two-level hierarchies with explicit memory systems) achieve 21% higher ASI retention than others, such as flat architectures [4]. Predictive models could identify such architectural correlates and recommend drift-resistant design.

Another promising approach is drift-resistant architectures that limit drift in model behavior. Song, Mao and colleagues showed that retrieval-augmented generation with explicit memory systems can reduce drift by bounding the model outputs to verifiable knowledge bases [14]. Likewise, role-specialized agent frameworks where each agent maintains distinct responsibilities may also limit the scope of drift across the system [15].

Formal verification of behavioral stability guarantees has been less addressed. Akter, Shihab, and Sharma have proposed anytime-valid answer sufficiency certificates, or e-processes, for guaranteeing δ-level control of premature stopping, independent of the stopping time [12]. Extending such guarantees to wider aspects of behavioral stability would help establish confidence in systems based on LLMs.

6. Conclusion

This paper also proposed the Enterprise AI Behavior Control Plane (BCP), an architecture at the platform layer for measuring, comparing, monitoring and regulating the real-world behavioral drifts of production GenAI systems. The BCP addresses longitudinal behavior management which is arguably the missing dimension of governance in enterprise AI. According to experiments conducted in deployed systems, behavioral drift affects 23.9% to 62.0% of deployed agents, and uncontrolled behavioral drift reduces success rates by up to 42.0%. The framework is the first to perform enterprise AI governance as a continuous control problem rather than a static evaluation problem.

In summary, this paper argues that longitudinal behavioral governance practices are not optional for enterprises that choose to deploy GenAI at scale because longitudinal research, production deployment observations, and simulation-based evaluations have shown that behavioral drift will happen with a statistically meaningful rate and that performance degradation will be observed if not addressed. The BCP architecture includes a constant baseline measurement, drift detection and alerting via statistics, behavioral SLAs, progressive rollouts, automatic rollback and dashboards.

As GenAI systems grow to be more autonomous, persistent, and applied to more complex multi-agent settings, behavioral stability is ever more critical. The principles and architecture in this paper can be used to design the new discipline Enterprise AI Reliability Engineering (AIRE), which adapts Site Reliability Engineering (SRE) practices to probabilistic AI systems. Just as SRE transformed the reliability of distributed systems in the last decade, we believe AIRE can transform the next generation of production-grade generative AI systems.

REFERENCES

1. Chen, L., Zaharia, M., & Zou, J. (2023, July 18). How is ChatGPT's behavior changing over time? ArXiv.org. https://doi.org/10.48550/arXiv.2307.09009
2. Wei, J., Tay, Y., Bommasani, R., Raffel, C., Zoph, B., Borgeaud, S., Yogatama, D., Bosma, M., Zhou, D., Metzler, D., Chi, E. H., Hashimoto, T., Vinyals, O., Liang, P., Dean, J., & Fedus, W. (2022). Emergent Abilities of Large Language Models. ArXiv:2206.07682 [Cs]. https://arxiv.org/abs/2206.07682
3. Oruche, R., Goruganthu, S. K., Akula, R., Cheng, X., Goni, A. M., Shibo, B. W., Kee, K., Zampieri, M., & Calyam, P. (2025). A Survey on the Recent Advancements in Human-Centered Dialog Systems. ACM Computing Surveys. https://doi.org/10.1145/3729220
4. Perez, E., Huang, S., Song, F., Cai, T., Ring, R., Aslanides, J., Glaese, A., McAleese, N., & Irving, G. (2022). Red Teaming Language Models with Language Models. ArXiv:2202.03286 [Cs]. https://arxiv.org/abs/2202.03286
5. Nicholson, C. (2026). Quantifying non deterministic drift in large language models. ArXiv.org. https://arxiv.org/abs/2601.19934
6. Şenol, A., Agrawal, G., & Liu, H. (2025). Domain Knowledge-Enhanced LLMs for Fraud and Concept Drift Detection. ArXiv.org. https://arxiv.org/abs/2506.21443
7. Hu, L., Lu, Y., & Feng, Y. (2025). Concept Drift Detection Based on Deep Neural Networks and Autoencoders. Applied Sciences, 15(6), 3056–3056. https://doi.org/10.3390/app15063056
8. Zou, A., Wang, Z., Zico, K. J., & Fredrikson, M. (2023). Universal and Transferable Adversarial Attacks on Aligned Language Models. ArXiv.org. https://arxiv.org/abs/2307.15043
9. IBM. (2024, October 8). Site reliability engineering. Ibm.com. https://www.ibm.com/think/topics/site-reliability-engineering
10. Wei, B., Jiang, R., Zhang, R., Liu, Y., Niyato, D., Sun, Y., Lu, Y., Li, Y., Mao, S., Yuen, C., Renzo, D., & Peng, M. (2025). Large Language Models for Next-Generation Wireless Network Management: A Survey and Tutorial. ArXiv.org. https://arxiv.org/abs/2509.05946
11. Ruan, Y., Dong, H., Wang, A., Pitis, S., Zhou, Y., Ba, J., Dubois, Y., Maddison, C. J., & Hashimoto, T. (2023). Identifying the Risks of LM Agents with an LM-Emulated Sandbox. ArXiv.org. https://arxiv.org/abs/2309.15817
12. Bhardwaj, R., & Poria, S. (2023). Red-Teaming Large Language Models using Chain of Utterances for Safety-Alignment. ArXiv.org. https://arxiv.org/abs/2308.09662
13. Tian, Y., Yang, X., Zhang, J., Dong, Y., & Su, H. (2023). Evil Geniuses: Delving into the Safety of LLM-based Agents. ArXiv.org. https://arxiv.org/abs/2311.11855
14. Senol, A., Agrawal, G., & Liu, H. (2025). Joint Detection of Fraud and Concept Drift inOnline Conversations with LLM-Assisted Judgment. ArXiv.org. https://arxiv.org/abs/2505.07852
15. Prachi Kothiyal. (2025, September 3). SRE.ai Secures $7.2M to Advance DevOps with AI-Driven Automation. Talent500 Blog. https://talent500.com/blog/sre-ai-raises-7m-devops-ai-agents/
16. OpenAI. (2023). GPT-4 Technical Report. ArXiv:2303.08774 [Cs]. https://doi.org/10.48550/arXiv.2303.08774
17. Dehal, R. S., Sharma, M., & Rajabi, E. (2025). Knowledge Graphs and Their Reciprocal Relationship with Large Language Models. Machine Learning and Knowledge Extraction, 7(2), 38. https://doi.org/10.3390/make7020038
18. Park, J. S., O'Brien, J. C., Cai, C. J., Morris, M. R., Liang, P., & Bernstein, M. S. (2023). Generative Agents: Interactive Simulacra of Human Behavior. ArXiv:2304.03442 [Cs]. https://arxiv.org/abs/2304.03442
19. Li, H., Dong, Q., Chen, J., Su, H., Zhou, Y., Ai, Q., Ye, Z., & Liu, Y. (2024). LLMs-as-Judges: A Comprehensive Survey on LLM-based Evaluation Methods. ArXiv.org. https://arxiv.org/abs/2412.05579
20. Akter, S., Shihab, Ibne Farabi, & Sharma, A. (2025). Anytime-Valid Answer Sufficiency Certificates for LLM Generation via Sequential Information Lift. ArXiv.org. https://arxiv.org/abs/2510.06478
21. Li, G., Kader, A., Itani, H., Khizbullin, D., & Ghanem, B. (2023). CAMEL: Communicative Agents for "Mind" Exploration of Large Scale Language Model Society. https://doi.org/10.48550/arxiv.2303.17760
22. Şenol, A., Agrawal, G., & Liu, H. (2026). Domain Knowledge-Enhanced LLMs for Fraud and Concept Drift Detection. Electronics, 15(3), 534. https://doi.org/10.3390/electronics15030534
23. Ousidhoum, N., Zhao, X., Fang, T., Song, Y., & Yeung, D.-Y. (2021). Probing Toxic Content in Large Pre-Trained Language Models (pp. 4262–4274). https://aclanthology.org/2021.acl-long.329.pdf
24. Maha Lharoui ROLE-SPECIFIC AI USE CASES FOR ENHANCING DAILY OPERATIONS IN ENERGY UPGRADES PROJECTS. (n.d.). Retrieved May 7, 2026, from https://www.theseus.fi/bitstream/handle/10024/893744/Lharoui_Maha.pdf?sequence=2
25. Wei, A., Haghtalab, N., & Steinhardt, J. (2023, July 5). Jailbroken: How Does LLM Safety Training Fail? ArXiv.org. https://doi.org/10.48550/arXiv.2307.02483

26. Agent Drift: Quantifying Behavioral Degradation in Multi-Agent LLM Systems Over Extended Interactions. (2026). Arxiv.org. https://arxiv.org/html/2601.04170v1
27. Nadeem, M., Bethke, A., & Reddy, S. (2020). StereoSet: Measuring stereotypical bias in pretrained language models. Arxiv.org. https://arxiv.org/abs/2004.09456