DOMUSMIND: A BENCHMARK FOR EVALUATING LIFELONG SMART HOME AGENTS UNDER DRIFT

Rong Xu[1] **Yinxin Wan**[2] **Xiaochan Xue**[3*]
[1]Stevens Institute of Technology [2]University of Massachusetts Boston
[3]University of Hawaii at Mānoa

ABSTRACT

Smart home agents require continuous operation in non-stationary environments where human preferences and device reliability keep evolving. However, dominant evaluation protocols remain episodic and reset-based, failing to capture the degradation and recovery dynamics essential for long-term deployment. To address this gap, we introduce **DomusMind**, a benchmark for evaluating lifelong agents under two sources of non-stationarity: *preference drift* and *tool drift*. **DomusMind** instantiates a persistent smart-home control loop where agents balance autonomous execution and user burden. By tracking time-resolved metrics across preference, tool, and mixed drift scenarios, our results show that online Theory of Mind (ToM) with uncertainty-gated confirmation provides the most robust adaptation. Notably, ORACLE persona access alone does not eliminate failures under *tool drift*, identifying execution reliability as a distinct bottleneck. By sweeping a confirmation threshold, **DomusMind** characterizes a success–annoyance frontier that enables principled selection of operating points for long-horizon alignment.

1 INTRODUCTION

Smart homes are an important real-world deployment environment for AI agents, requiring continuous operation in safety-critical environments without system resets Fernandes et al. (2016). To maintain alignment in these settings, an agent must synchronize its behavior with the household as residents revise routines, shift their automation tolerances, and update their device ecosystems Chalhoub et al. (2021). This demand for long-term personalization is already evident in industry. For example, SmartThings is moving towards "Home AI" that uses ambient sensing to recognize daily activities and deliver personalized experiences Samsung SmartThings (2024). However, most existing agent benchmarks are episodic and reset-based, and therefore fail to evaluate the key objective in smart homes: sustained assistance that preserves user control under changing conditions.

In the home, this challenge arises from two sources of non-stationarity. *Preference drift* reflects the evolving preferences for human privacy and autonomy, so previously helpful actions can become intrusive Zhang et al. (2025). Sustaining alignment therefore requires a continual Theory of Mind (ToM): the ability to persistently model and update states while avoiding forgetting and overfitting Luo et al. (2025). Second, *tool drift* affects the execution layer as devices become unreliable, latency fluctuates, and failure rates increase, thereby altering both the safety risks and the operational costs of acting Fu et al. (2021); Wan et al. (2020). These drifts contrast with the inherent rigidity of practical systems. High costs of data collection and safety validation preclude frequent retraining, while online fine-tuning is hindered by noisy feedback and the risk of unvetted regressions Zhang et al. (2024). Consequently, an agent's utility is measured by its lifelong alignment stability: how gracefully it degrades, how rapidly it recovers, and the human burden imposed during adaptation.

To bridge this gap, we introduce **DomusMind**, a reproducible benchmark for evaluating long-horizon, lifelong smart-home agents under controlled non-stationarity. **DomusMind** models a Home Assistant–style control loop where agents choose meta-actions to balance autonomy and the user burden induced by confirmation interruptions. Across preference, tool, and mixed drift scenarios, we find that online ToM with uncertainty-gated confirmation provides the most robust adaptation.

*Corresponding author: xxue@hawaii.edu

Our experiments reveal that while online tracking is essential for preference recovery, even oracle access to user preferences cannot eliminate failures caused by unreliable tool execution, identifying execution reliability as a distinct bottleneck. Finally, by sweeping the confirmation threshold, **DomusMind** characterizes a success–annoyance frontier, enabling the principled selection of operating points for long-horizon alignment that minimizes user burden.

2 Related Work

Smart home benchmarks and simulators. Prior work introduced smart home datasets and simulators for activity understanding and automation, including annotated in-home sensor streams such as CASAS and the van Kasteren dataset Van Kasteren et al. (2008); Cook et al. (2009), and controllable simulators such as OpenSHS Alshammari et al. (2017). More recently, SimuHome provides a time-accelerated, Matter-based simulator and an interactive benchmark suite for smart home LLM agents Seo et al. (2025). However, most resources still emphasize offline recognition or reset-based episodes, rather than long-term deployment where an agent must decide whether to execute, ask for confirmation, delay, or abstain as the household evolves.

Non-stationarity and continual evaluation. Distribution shift benchmarks such as WILDS motivate evaluation beyond in-distribution test sets Koh et al. (2021). Continual learning and continual reinforcement learning study adaptation under sequential experience and catastrophic forgetting Kirkpatrick et al. (2017); Wolczyk et al. (2021); Zhang et al. (2023). In RL and bandits, non-stationarity is often handled through explicit drift models and change point detection Xie et al. (2021); Besson et al. (2022). These settings typically abstract away smart-home tool execution and rarely separate *preference drift* from *tool drift*, or report user-facing metrics such as confirmation burden and safety violations that are central in home automation.

User modeling, oversight, and interactive agents. ToM modeling motivates inferring latent traits from behavior, as in ToMnet Rabinowitz et al. (2018), and mixed initiative principles formalize when systems should act versus defer under uncertainty Horvitz (1999). Preference-based learning connects interaction feedback to behavior optimization Christiano et al. (2017), and recent work further studies personalization under drifting implicit preferences using real-world dissatisfaction signals or decoding time adaptation Wang et al. (2025b); Kim et al. (2025). Representative web and software agent benchmarks evaluate multi-step tool use and long-horizon workflows, and StableToolBench improves reproducibility via tool virtualization Wang et al. (2025a); Guo et al. (2024). However, they largely assume stable goals and tools, leaving long-term alignment under preference and *tool drift* with burden and safety costs underexplored, which **DomusMind** targets via time-resolved success, burden, safety, efficiency, and recovery.

3 DomusMind

We present **DomusMind**, a reproducible, decision-centric benchmark for lifelong smart-home control without resets. Inspired by Home Assistant, **DomusMind** models device control as service calls Home Assistant (2026) and targets alignment stability under non-stationary conditions. It injects two deployment-relevant sources of drift: *preference drift* in a latent user persona and *tool drift* in the execution layer. Agents interact through a small set of meta-actions (Execute, AskConfirm, Delay, NoOp), enabling time-resolved evaluation of success, safety, user burden, and efficiency. Figure 1 summarizes the closed-loop components. Each run produces step-level logs and standardized metrics for replayable comparison across methods.

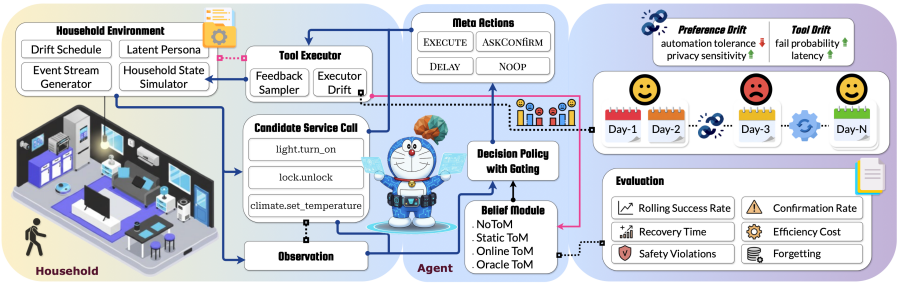

Figure 1: **DomusMind** evaluates long-horizon smart-home agents under controllable drifts.

3.1 BENCHMARK INTERACTION LOOP

DomusMind defines a closed-loop environment comprising an agent A, a household state simulator S subject to drift, and a tool executor E. We adopt the step $t \in \{1, \ldots, T\}$ as the discrete unit of deployment time, representing one iteration of the interaction loop. At each step, the simulator S updates the household state and emits an observation o_t, which the environment provides to the agent. The environment may propose one and only one candidate service call $\hat{c}_t$ produced by an event-stream generator, such as `turn_on(light)`, `unlock(door)`, or `set_temperature(thermostat)`. Steps with $\hat{c}_t$ are *actionable*. Given $(o_t, \hat{c}_t)$, the agent selects a meta-action $a_t \in \{\text{EXECUTE}, \text{ASKCONFIRM}, \text{DELAY}, \text{NOOP}\}$ as follows:

- **EXECUTE**: The executor E attempts $\hat{c}_t$. The environment logs the outcome and samples user feedback based on the current persona.
- **ASKCONFIRM**: The environment queries the user and executes $\hat{c}_t$ only upon explicit approval.
- **DELAY/NOOP**: No action is taken. If the triggering context persists, the situation may reappear in subsequent steps.

This interface decouples the proposal of candidate service calls ($\hat{c}_t$) from the agent's decision-making under uncertainty, safety constraints, and drift. Before each interaction, the environment updates the persona and executor parameters according to a pre-defined drift schedule. The resulting T-step trajectory, captured in JSONL format, provides a high-fidelity log of interactions and outcomes for a standardized metric summary.

3.2 SCENARIO GENERATION AND DRIFT MODELING

The event generator instantiates recurring home-automation templates, including comfort-oriented cases (e.g., motion-triggered lighting and high-temperature cooling) and privacy-sensitive cases (e.g., doorbell-triggered door unlocking). At step t, a template determines whether to emit a candidate service call $\hat{c}_t$ and, if so, which call to propose. The *acceptability* of $\hat{c}_t$ is governed by a latent persona state z_t together with the current context ctx_t, while the *execution reliability* is governed by the tool executor.

User feedback. We represent acceptability with a binary approval variable $y_t \in \{0, 1\}$ sampled as $y_t \sim P(y \mid z_t, \text{ctx}_t)$, where $y_t = 1$ indicates the user would accept the proposed call. This captures heterogeneity in privacy sensitivity and autonomy tolerance across households. If the agent chooses ASKCONFIRM, the environment reveals y_t and executes $\hat{c}_t$ only when $y_t = 1$. If the agent chooses EXECUTE, the environment still samples y_t for alignment evaluation. In safety-critical templates, executing when $y_t = 0$ triggers a violation flag. The executor simulates device and service-call behavior and returns structured outcomes, including success, latency, retry count, and error flags.

Drift dynamics and budgets. DomusMind introduces non-stationarity through a configurable drift schedule specified by onset time t_{drift}, optional transition duration, and severity. *Preference drift* perturbs z_t either abruptly or gradually, shifting autonomy tolerance and privacy sensitivity over time. *Tool drift* degrades executor reliability by increasing failure probability, latency, and offline-event frequency. Separately, *budget settings* define deployment constraints on user burden and efficiency to select operating points, without altering the underlying environment dynamics.

3.3 METRICS FOR LIFELONG ALIGNMENT STABILITY

DomusMind computes alignment metrics on *actionable* steps (those with a candidate service call) and summarizes long-horizon stability from the resulting trajectories. On each actionable step t, the environment samples an approval variable $y_t \in \{0, 1\}$ for the candidate call. Let $x_t \in \{0, 1\}$ indicate whether the call is executed at step t (after confirmation when applicable), and let $e_t \in \{0, 1\}$ indicate executor success when $x_t = 1$. We define step success s_t as

$$s_t = \begin{cases} 1, & y_t = 0 \text{ and } x_t = 0, \\ 1, & y_t = 1 \text{ and } x_t = 1 \text{ and } e_t = 1, \\ 0, & \text{otherwise.} \end{cases} \tag{1}$$

Core Performance and Safety. We define mean success is $\bar{s} = \mathbb{E}[s_t]$ over actionable steps. We measure confirmation-induced interruption by the confirmation rate $r_{\text{confirm}} = \frac{\#\text{ASKCONFIRM}}{\#\text{actionable steps}}$.

We report the safety-violation rate r_{safevio} as the number of violations per 100 actionable privacy-sensitive suggestions, such as door unlocking. **Adaptation Dynamics.** To quantify dynamics, we compute a rolling success curve $S_{\text{roll}}(t)$ by averaging s_t over a fixed window of actionable steps. For a drift event at t_{drift}, we report: (i) *PreDrift* (S_{pre}), the mean of $S_{\text{roll}}(t)$ over the window immediately before t_{drift}; (ii) *Drift* (S_{drift}), the mean of $S_{\text{roll}}(t)$ over the configured drift interval; (iii) *Post-Drift* (S_{post}), the mean of $S_{\text{roll}}(t)$ for all steps following t_{drift}, capturing the immediate impact and adaptation phase; (iv) *Steady* (S_{steady}), the mean of $S_{\text{roll}}(t)$ over a late window after performance stabilizes; and (v) *Recovery time*, defined as the time to return to 90% of *PreDrift*. For a recovery reference step t_0, we define $T_{\text{recover}}(t_0) = \min\{t > t_0 \mid S_{\text{roll}}(t) \geq 0.9\, S_{\text{pre}}\} - t_0$. For schedules with a later regime revisit, we report $T_{\text{recover1}} = T_{\text{recover}}(t_{\text{drift}})$ and $T_{\text{recover2}} = T_{\text{recover}}(t_{\text{revisit}})$. When a pre-drift regime reappears, we quantify forgetting as $F = S_{\text{pre}} - S_{\text{steady}}$. We summarize efficiency and burden using an aggregate cost that combines confirmation and executor outcomes (including latency and retries) with fixed weights shared across methods. For policies with a confirmation-gating threshold τ, we sweep τ over a fixed grid and report $(\bar{s}, r_{\text{confirm}})$ to obtain an empirical success–annoyance trade-off curve.

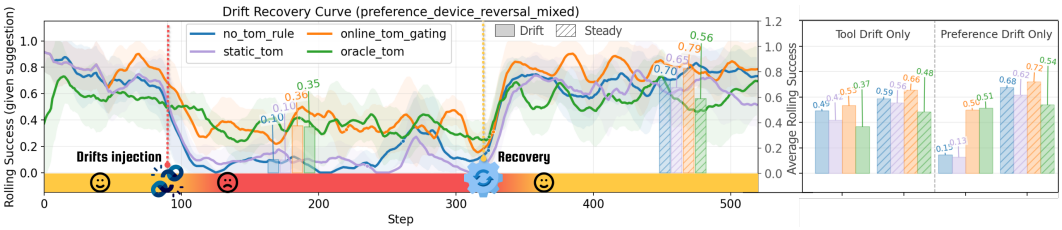

Figure 2: Rolling success under mixed drift shows persistent post-drift misalignment.

3.4 Baselines and Evaluation Protocol

We use Theory of Mind (ToM) to denote inferring and updating user preferences and constraints from interaction history and feedback, and we compare policies that differ in persona representation and drift adaptation. NoToM is a fixed rule that ignores persona uncertainty, using a constant confirmation rate and otherwise executing $\hat{c}_t$. Static fits a persona model during a calibration phase and then freezes it. Online continuously updates a belief over the persona from feedback and uses an uncertainty-gated confirmation rule parameterized by τ, with larger τ yielding more confirmations and more conservative behavior. Oracle observes the true persona state z_t at each step, removing preference uncertainty but not execution failures. We define oracle access only for the persona, since executor states driving *tool drift* are typically unobservable in practice. We also include tinker_policy, a prompt-only LLM baseline that maps $(o_t, \hat{c}_t)$ to a meta-action without updating any persona state (Appendix D). All baselines interact with the **DomusMind** environment and are evaluated under identical drift schedules, severity levels, and random seeds. Metrics are computed on actionable steps, and we sweep τ on a fixed grid when applicable.

4 Experiments and Evaluation

4.1 Setup

We evaluate **DomusMind** in long-horizon, no-reset runs that emulate continuous deployment, with one discrete step corresponding to one hour. All baselines in Section 3.4 interact with the same environment under identical drift schedules, severity levels, and random seeds. We consider three drift settings: *preference drift* only, *tool drift* only, and mixed drift, where both occur together, optionally with a regime revisit to measure forgetting, and we vary drift severity while keeping the horizon and schedule fixed. We evaluate on actionable steps and report rolling success, confirmation rate, safety violation rate (per 100 actionable suggestions on safety-critical calls), recovery time, and pre- and post-drift summary levels (*PreDrift*, *PostDrift*). For Online ToM Gating, we sweep the confirmation threshold τ over a fixed grid to trace the success–annoyance trade-off curve.

4.2 Results

Figure 2 plots rolling success for the mixed-drift setting (drift at ∼90, revisit at ∼320), with preference-only and tool-only curves deferred to Appendix A. Shaded regions indicate variabil-

	Policy	Success	r_{safevio}	Cost	PreDrift	PostDrift	F	T_{recover1}	T_{recover2}	r_{confirm}
Pref-R	NoToM	0.45±0.02	0.00±0.00	13.05±0.89	0.65±0.25	0.36±0.05	0.29±0.29	143.60±117.07	3.40±5.64	0.25±0.03
	STATIC	0.42±0.10	10.57±14.54	43.38±41.91	0.69±0.22	0.26±0.11	0.44±0.29	115.20±129.76	0.00±0.00	0.19±0.14
	ONLINE	0.64±0.03	1.04±0.93	19.82±2.98	0.80±0.15	0.80±0.24	-0.00±0.26	82.60±91.43	3.60±7.50	0.76±0.04
	ORACLE	0.53±0.02	14.48±3.34	54.03±9.36	0.55±0.27	0.43±0.21	0.12±0.38	38.00±45.99	19.00±23.72	0.44±0.04
Tool-R	NoToM	0.55±0.04	0.00±0.00	16.91±0.91	0.59±0.31	0.54±0.23	0.06±0.46	15.40±31.71	2.80±6.26	0.25±0.03
	STATIC	0.50±0.08	10.57±14.54	46.91±41.83	0.58±0.27	0.48±0.27	0.10±0.35	27.20±37.28	6.80±15.21	0.19±0.14
	ONLINE	0.63±0.03	0.87±0.61	23.77±1.25	0.78±0.14	0.57±0.28	0.21±0.37	75.20±90.18	3.40±7.60	0.79±0.09
	ORACLE	0.43±0.04	25.43±2.81	85.46±11.21	0.51±0.24	0.26±0.11	0.25±0.35	23.20±44.90	26.40±20.17	0.00±0.00
Mixed-R	NoToM	0.44±0.03	0.00±0.00	16.23±0.58	0.65±0.25	0.40±0.12	0.25±0.35	171.40±101.28	3.40±5.64	0.25±0.03
	STATIC	0.41±0.09	10.57±14.54	46.83±41.92	0.69±0.22	0.25±0.06	0.44±0.26	145.20±140.53	0.00±0.00	0.19±0.14
	ONLINE	0.60±0.04	0.51±0.47	21.73±1.41	0.80±0.15	0.45±0.21	0.35±0.34	130.00±122.08	7.00±6.28	0.90±0.06
	ORACLE	0.49±0.03	14.48±3.34	56.19±9.11	0.55±0.27	0.43±0.17	0.12±0.26	95.60±101.99	4.00±5.34	0.44±0.04

Table 1: Performance (mean±std) under strong preference (Pref-R), tool (Tool-R), and mixed (Mixed-R) drift. "-R" denotes a later regime revisit. Baselines: NoToM (rule), STATIC (one-time personalization), ONLINE (uncertainty-gated confirmation, $\tau = 0.5$), and ORACLE (true persona).

ity across random seeds. After drift, NoToM and STATIC collapse to near-zero rolling success and remain misaligned until later regime revisit, a failure mode that reset-based evaluation would miss. In contrast, ONLINE sustains substantially higher performance ($S_{\text{drift}} \approx 0.36$) than frozen or rule-based baselines ($S_{\text{drift}} \approx 0.10$) and recovers to the highest plateau for S_{steady}. Notably, ORACLE access to z_t does not match ONLINE, because removing uncertainty over the latent persona state but not step-level approval randomness or execution failures Table 1 further quantifies these: in Mixed-R, ONLINE outperforms ORACLE in success (0.60 vs. 0.49) while reducing safety violations rate from 14.48 to 0.51 and lowering aggregate cost (21.73 vs. 56.19), at the cost of a higher confirmation rate ($r_{\text{confirm}} = 0.90$ vs. 0.44). ONLINE also shows incomplete average post-drift recovery (PostDrift 0.45 vs. PreDrift 0.80), reflecting that mixed drift introduces an execution bottleneck beyond preference estimation. Consistent with this, in Tool-R, ORACLE degrades sharply (Success 0.43, PostDrift 0.26) with high cost (85.46), whereas ONLINE achieves 0.63 success with PostDrift 0.57 and much lower cost (23.77). In Pref-R, ONLINE exhibits negligible forgetting ($F \approx 0$) compared with STATIC ($F \approx 0.44$), supporting that continual belief updates are important for lifelong alignment stability.

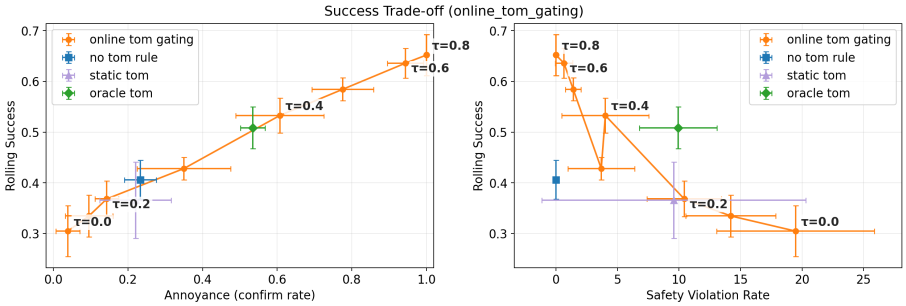

Figure 3: Success trade-offs induced by sweeping τ in ONLINE.

Figure 3 traces the success versus confirmation frontier induced by sweeping τ in ONLINE. Higher τ increases confirmations and typically improves success while driving safety violations down to near zero, exposing tunable operating points under different confirmation budgets.

5 Discussion and Limitations

Beyond reporting scores, **DomusMind** supports drift attribution by separating preference uncertainty from execution unreliability, while also exposing confounds from per-step approval randomness. In this view, confirmation is a costly information-gathering action, turning the success–annoyance frontier into an information-cost curve rather than mere hyperparameter tuning. Moreover, recovery time depends on protocol choices such as the rolling window and threshold, motivating robustness checks, and the current benchmark should be read as a controlled stress test rather than a calibrated digital twin. Looking ahead, multi-user households introduce preference conflicts and authority constraints, so a single persona z_t is insufficient and confirmation must specify who is asked. Meanwhile, feedback is often missing or delayed, reducing observability for online updates and making confirmation an even more central probing mechanism.

REFERENCES

Nasser Alshammari, Talal Alshammari, Mohamed Sedky, Justin Champion, and Carolin Bauer. Openshs: Open smart home simulator. *Sensors*, 17(5), 2017. ISSN 1424-8220. doi: 10.3390/s17051003. URL https://www.mdpi.com/1424-8220/17/5/1003.

Lilian Besson, Emilie Kaufmann, Odalric-Ambrym Maillard, and Julien Seznec. Efficient change-point detection for tackling piecewise-stationary bandits. *Journal of Machine Learning Research*, 23(77):1–40, 2022.

George Chalhoub, Martin J Kraemer, Norbert Nthala, and Ivan Flechais. "it did not give me an option to decline": A longitudinal analysis of the user experience of security and privacy in smart home products. In *Proceedings of the 2021 CHI conference on human factors in computing systems*, pp. 1–16, 2021.

Paul F Christiano, Jan Leike, Tom Brown, Miljan Martic, Shane Legg, and Dario Amodei. Deep reinforcement learning from human preferences. *Advances in neural information processing systems*, 30, 2017.

Diane Cook, Maureen Schmitter-Edgecombe, Aaron Crandall, Chad Sanders, and Brian Thomas. Collecting and disseminating smart home sensor data in the casas project. In *Proceedings of the CHI workshop on developing shared home behavior datasets to advance HCI and ubiquitous computing research*, pp. 1–7. IEEE, 2009.

Kyle Cox, Jiawei Xu, Yikun Han, Rong Xu, Tianhao Li, Chi-Yang Hsu, Tianlong Chen, Walter Gerych, and Ying Ding. Mapping from meaning: Addressing the miscalibration of prompt-sensitive language models. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pp. 23696–23703, 2025.

Earlence Fernandes, Jaeyeon Jung, and Atul Prakash. Security analysis of emerging smart home applications. In *2016 IEEE symposium on security and privacy (SP)*, pp. 636–654. IEEE, 2016.

Chenglong Fu, Qiang Zeng, and Xiaojiang Du. {HAWatcher}:{Semantics-Aware} anomaly detection for appified smart homes. In *30th USENIX Security Symposium (USENIX Security 21)*, pp. 4223–4240, 2021.

Zhicheng Guo, Sijie Cheng, Hao Wang, Shihao Liang, Yujia Qin, Peng Li, Zhiyuan Liu, Maosong Sun, and Yang Liu. Stabletoolbench: Towards stable large-scale benchmarking on tool learning of large language models. *arXiv preprint arXiv:2403.07714*, 2024.

Home Assistant. Home assistant. https://www.home-assistant.io/, 2026.

Eric Horvitz. Principles of mixed-initiative user interfaces. In *Proceedings of the SIGCHI conference on Human Factors in Computing Systems*, pp. 159–166, 1999.

Minbeom Kim, Kang-il Lee, Seongho Joo, Hwaran Lee, Thibaut Thonet, and Kyomin Jung. Drift: Decoding-time personalized alignments with implicit user preferences. *arXiv preprint arXiv:2502.14289*, 2025.

James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, et al. Overcoming catastrophic forgetting in neural networks. *Proceedings of the National Academy of Sciences*, 114(13):3521–3526, 2017.

Pang Wei Koh, Shiori Sagawa, Henrik Marklund, Sang Michael Xie, Marvin Zhang, Akshay Balsubramani, Weihua Hu, Michihiro Yasunaga, Richard Lanas Phillips, Irena Gao, et al. Wilds: A benchmark of in-the-wild distribution shifts. In *International conference on machine learning*, pp. 5637–5664. PMLR, 2021.

Yun Luo, Zhen Yang, Fandong Meng, Yafu Li, Jie Zhou, and Yue Zhang. An empirical study of catastrophic forgetting in large language models during continual fine-tuning. *IEEE Transactions on Audio, Speech and Language Processing*, 2025.

Neil Rabinowitz, Frank Perbet, Francis Song, Chiyuan Zhang, SM Ali Eslami, and Matthew Botvinick. Machine theory of mind. In *International conference on machine learning*, pp. 4218–4227. PMLR, 2018.

Samsung SmartThings. Samsung smartthings introduces new ai features at unpacked. https://blog.smartthings.com/roundups/samsung-smartthings-introduces-new-ai-features-at-unpacked/, 2024.

Gyuhyeon Seo, Jungwoo Yang, Junseong Pyo, Nalim Kim, Jonggeun Lee, and Yohan Jo. Simuhome: A temporal-and environment-aware benchmark for smart home llm agents. *arXiv preprint arXiv:2509.24282*, 2025.

Tim Van Kasteren, Athanasios Noulas, Gwenn Englebienne, and Ben Kröse. Accurate activity recognition in a home setting. In *Proceedings of the 10th international conference on Ubiquitous computing*, pp. 1–9, 2008.

Yinxin Wan, Kuai Xu, Guoliang Xue, and Feng Wang. Iotargos: A multi-layer security monitoring system for internet-of-things in smart homes. In *IEEE INFOCOM 2020-IEEE Conference on Computer Communications*, pp. 874–883. IEEE, 2020.

Weixuan Wang, Dongge Han, Daniel Madrigal Diaz, Jin Xu, Victor Rühle, and Saravan Rajmohan. Odysseybench: Evaluating llm agents on long-horizon complex office application workflows. *arXiv preprint arXiv:2508.09124*, 2025a.

Yifan Wang, Bolian Li, Junlin Wu, Zhaoxuan Tan, Zheli Liu, Ruqi Zhang, Ananth Grama, and Qingkai Zeng. Drift: Learning from abundant user dissatisfaction in real-world preference learning. *arXiv preprint arXiv:2510.02341*, 2025b.

Maciej Wolczyk, Michal Zajkac, Razvan Pascanu, Lukasz Kucinski, and Piotr Milos. Continual world: A robotic benchmark for continual reinforcement learning. *Advances in Neural Information Processing Systems*, 34:28496–28510, 2021.

Annie Xie, James Harrison, and Chelsea Finn. Deep reinforcement learning amidst continual structured non-stationarity. In *International Conference on Machine Learning*, pp. 11393–11403. PMLR, 2021.

Xiaochan Xue, Shucheng Yu, and Min Song. Secure device trust bootstrapping against collaborative signal modification attacks. In *IEEE INFOCOM 2023-IEEE Conference on Computer Communications*, pp. 1–10. IEEE, 2023.

Han Zhang, Yu Lei, Lin Gui, Min Yang, Yulan He, Hui Wang, and Ruifeng Xu. Cppo: Continual learning for reinforcement learning with human feedback. In *The Twelfth International Conference on Learning Representations*, 2024.

Han Zhang, Lin Gui, Yu Lei, Yuanzhao Zhai, Yehong Zhang, Zhuo Zhang, Yulan He, Hui Wang, Yue Yu, Kam-Fai Wong, et al. Copr: Continual human preference learning via optimal policy regularization. In *Findings of the Association for Computational Linguistics: ACL 2025*, pp. 5377–5398, 2025.

Yuji Zhang, Jing Li, and Wenjie Li. Vibe: Topic-driven temporal adaptation for twitter classification. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pp. 3340–3354, 2023.

A ADDITIONAL PLOTS

Figures 4 and 5 show rolling success under preference-only and tool-only drift, respectively. Together, they complement the mixed-drift results in the main paper by separating single-drift behavior from the mixed setting.

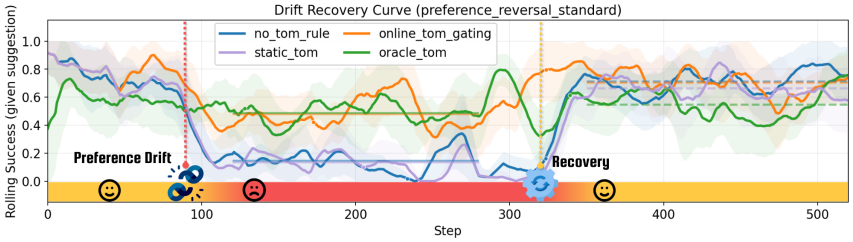

Figure 4: Rolling success under preference-only drift.

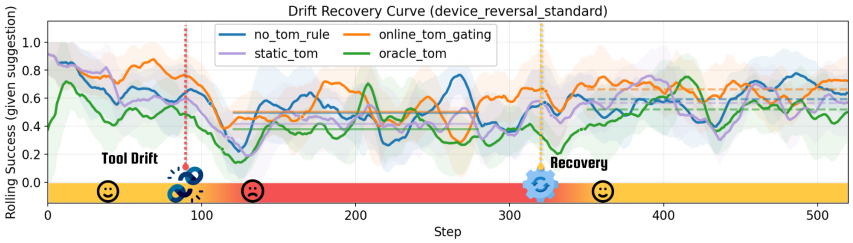

Figure 5: Rolling success under tool-only drift.

B ENVIRONMENT AND DRIFT SUMMARY

This appendix summarizes the **DomusMind** environment contract at the level of state variables, distributions, and logged signals. A run is seed-controlled with horizon T, where one discrete step corresponds to one hour of deployment time. At each step $t \in \{1, \ldots, T\}$, the environment emits a structured observation o_t and an optional suggested service call $\hat{c}_t$ in Home Assistant style. The agent returns a discrete meta action $a_t \in \{\text{EXECUTE}, \text{ASKCONFIRM}, \text{DELAY}, \text{NOOP}\}$. The environment applies the action, samples feedback from a latent persona, and logs step-level signals that reproduce all reported metrics.

B.1 ENVIRONMENT CONTRACT

Latent state, dynamics, and observation. The simulator maintains a compact household state

$$s_t = \langle h_t, \theta_t, o_t^{\text{occ}}, m_t, d_t \rangle,$$

where h_t is hour of day, θ_t is indoor temperature, o_t^{occ} indicates occupancy, m_t indicates a motion event, and d_t indicates a delivery-waiting event used by the door-unlocking template. The hour advances deterministically, temperature follows a mean-reverting process with an hour-dependent target level, and occupancy, motion, and delivery indicators are sampled from simple hour-dependent distributions. At each step, the environment emits a structured observation o_t together with an optional suggested service call $\hat{c}_t$. When no suggestion is emitted, the event field is null. A step is actionable iff $\hat{c}_t \neq \varnothing$. Unless stated otherwise, all rates and rolling metrics in the paper are computed on actionable steps.

Field (type)	Meaning
`service` (*string*)	`domain.action`, for example `light.turn_on`
`entity_id` (*string*)	target entity id
`data` (*dict*)	template specific arguments
`template_id` (*string*)	template identifier
`privacy_sensitive` (*bool*)	whether the suggestion is privacy-sensitive

Table 2: Suggested service call schema.

Suggested service calls and templates. A suggestion $\hat{c}_t$ is a service call object with the schema in Table 2. The boolean `privacy_sensitive` marks safety-critical suggestions. **DomusMind**

proposes suggestions using a small template library. At step t, the generator builds the set of eligible templates whose trigger conditions are satisfied by the current s_t. If the eligible set is empty, then $\hat{c}_t$ is null. Otherwise, one eligible template is selected uniformly at random and instantiated into a service call.

Meta actions, execution, and step labels. On an actionable step, the agent chooses one meta action. If $a_t = \text{EXECUTE}$, the environment attempts the suggested call immediately via the tool executor. If $a_t = \text{ASKCONFIRM}$, the environment queries the persona for an approval label and executes only when approved. If $a_t = \text{DELAY}$ or $a_t = \text{NOOP}$, no tool call is attempted at this step. For metric computation, we define $x_t \in \{0, 1\}$ to indicate whether a service call is executed after confirmation when applicable. We set $x_t = 1$ when $a_t = \text{EXECUTE}$, and we set $x_t = 1$ when $a_t = \text{ASKCONFIRM}$ and the persona approves. Otherwise $x_t = 0$. When $x_t = 1$, the executor returns binary success $e_t \in \{0, 1\}$. On each actionable step, the environment samples a latent approval variable $y_t \in \{0, 1\}$ from the persona model in Section B.2. We use the step success label from the main paper. A safety violation is defined only for suggestions with `privacy_sensitive` true. We set `safety_violation`$= 1$ when the call is executed ($x_t = 1$) and the latent approval is negative ($y_t = 0$). This matches the main text description that executing a safety-critical call against user preference constitutes a violation, and it remains well defined even under tool failures.

B.2 Persona model and observed feedback

Each run maintains a latent persona state $z_t = \langle \alpha_t, \pi_t \rangle$, where $\alpha_t \in [0, 1]$ is automation tolerance and $\pi_t \in [0, 1]$ is privacy sensitivity. These parameters can drift over time according to Section C. Given a suggestion with template id k and privacy indicator $I_{\text{priv}} \in \{0, 1\}$, the persona computes an approval probability $p_t = \text{clip}_{[0,1]}(b_0 + w_\alpha \alpha_t + b_k - w_\pi \pi_t I_{\text{priv}})$. Here b_0 is a global intercept, w_α and w_π are fixed weights, and b_k is a template-specific bias. The latent approval is sampled as $y_t \sim \text{Bernoulli}(p_t)$. The environment also emits an observed feedback object with `complaint` and `reason`. When $a_t = \text{ASKCONFIRM}$, the policy observes a clean approval outcome, which is either denial or confirmed execution. When $a_t \in \{\text{EXECUTE}, \text{DELAY}, \text{NOOP}\}$, feedback is noisy and indirect, with complaint probability increasing when outcomes disagree with the latent approval and when tool execution fails

$$\Pr(\texttt{complaint}{=}1) \;=\; \text{clip}_{[0,1]}(\gamma_0 \;+\; \gamma_{\text{mis}}\, \mathbb{I}[y_t \neq x_t] \;+\; \gamma_{\text{fail}}\, \mathbb{I}[x_t{=}1 \wedge e_t{=}0])\,.$$

Here γ_0 is a base complaint rate, γ_{mis} weights approval mismatch, and γ_{fail} weights execution failure. The categorical `reason` encodes the realized case for analysis and debugging.

B.3 Tool executor and outcomes

The tool executor maintains mutable parameters `failure_prob`, `offline_rate`, `latency_mean`, `latency_jitter`, and `max_retries`. Tool drift modifies these parameters over time, see Section C. When executing a call ($x_t = 1$), the executor performs repeated attempts up to `max_retries`. For each attempt, it samples an offline event, a failure event, and a latency draw

$$u^{\text{off}} \sim \text{Bernoulli}(\texttt{offline_rate}), \qquad u^{\text{fail}} \sim \text{Bernoulli}(\texttt{failure_prob}),$$

$$\ell \sim \max\big(0,\, \mathcal{N}(\texttt{latency_mean}, \texttt{latency_jitter}^2)\big)\,.$$

Offline attempts fail with error `device_offline`, online failures use error `execution_failed`. The executor stops early on the first success, otherwise the call fails after exhausting attempts. The executor returns `tool_outcome` with schema in Table 3. These fields are used by efficiency and cost metrics.

C Drift and cost configuration

A scenario defines drift as a sorted list of drift events. Each event specifies (i) a start step `at_step`, (ii) a drift domain `drift_type` in {preference, tool}, (iii) a target attribute `target`, (iv) a temporal mode `mode` in {sudden, gradual}, (v) a duration in steps `duration` for gradual drift, and (vi) either an absolute final value `set` or an additive change `delta`. Drift updates are applied before generating o_t and $\hat{c}_t$, so the agent experiences drift immediately at the drift step.

Field	Type	Meaning
attempted	bool	whether execution was attempted ($x_t = 1$)
success	bool	executor success flag ($e_t = 1$)
latency_s	float	cumulative latency over all attempts
retries	int	number of retries used
error	string or null	terminal error code when not successful
device_offline	bool	whether any attempt observed an offline event

Table 3: Tool outcome schema.

Sudden drift updates the target once at `at_step`. Gradual drift linearly interpolates from the onset value to the specified final value across `duration` steps, then clamps to the exact final value. Each step logs `drift_flags` with an `active` bit and a list of activated event descriptors.

Preference drift modifies the latent persona parameters α_t and π_t in Section B.2. A strong preference drift uses an abrupt shift to low automation tolerance and high privacy sensitivity. A mild preference drift uses a slower transition over a fixed window.

Tool drift modifies executor parameters in Section B.3, including failure probability, offline rate, and mean latency. A strong tool drift applies an abrupt degradation, and a mild tool drift applies a gradual degradation. We do not define an oracle baseline that observes future tool outcomes or true latent tool health. In realistic deployments, the tool layer only reveals outcomes after an attempted call, and health signals are partial and interface-dependent. As a result, the clean oracle in this benchmark is an oracle persona that removes preference uncertainty, while tool drift remains a deployment-side source of failure. A related but distinct deployment issue is secure device trust bootstrapping during device onboarding, whereas our tool drift model focuses on runtime failures, offline events, and latency degradation after a device is already in the loop Xue et al. (2023).

C.1 Cost model and budget regimes

Each step logs an additive `total_cost`. Costs include confirmation cost, delay cost, unsafe action cost for `safety_violation`, execution cost when a tool call is attempted, retry cost proportional to `retries`, and latency cost proportional to `latency_s`. The logged scalar reward is included for bookkeeping only and is not part of the reported evaluation metrics. Budget regimes vary the weights of these components without changing the environmental dynamics.

D LLM prompt baseline

We include a prompt-only LLM baseline, also referred to as the TINKER_POLICY in the draft. The policy is intentionally stateless and does not maintain a user model or update parameters online, so any adaptation must come from its per-step decision rule. Prompt-only policies can also be sensitive to prompt phrasing, which may affect calibration even when prompts preserve meaning Cox et al. (2025). We therefore interpret this baseline as performance under a fixed prompt template rather than a prompt-invariant estimate. On each actionable step, we render a compact textual observation and the proposed Home Assistant style service call into a single turn instruction prompt. The prompt enumerates the action space and marks whether the suggested call is privacy sensitive:

```
You are a smart-home assistant.
Choose one action: EXECUTE, ASKCONFIRM, DELAY, or NOOP.
Observation: t=... hour=... temp=... occupancy=... event=...
Suggestion: Suggest <service> on <entity_id> data={<...>}
(privacy-sensitive|non-sensitive)
```

We query an external model with low-temperature decoding, with temperature 0 by default, and a small generation budget. For each run, we record the model identifier and decoding settings, and we log parse-hit and fallback rates so that prompt failures can be separated from weak decisions.