Instruction Bleed: Cross-Module Interference in Prompt-Composed Agentic Systems

Ching-Yu Lin [1] **Yifan Liu** [1]

Abstract

Practitioners of prompt-composed agentic systems report a recurring failure mode: editing one prompt module silently shifts the behavior of others despite no shared variable or executable dependency. We formalize this as *compositional behavioral leakage* (CBL): interference between modules sharing a context window. CBL is enabled by architectural non-isolation: transformer self-attention provides no formal boundary between concatenated modules. We probe CBL on a deployed job-evaluation agent (Claude Sonnet 4.6, 144 trials) through a reusable three-channel protocol that perturbs non-focal modules along *volume*, *content*, and *form*. Only the *content* channel produces a detectable paired effect (Cohen's $d = 0.63$, bootstrap 95% CI excluding zero); no recommendation flipped—a sub-threshold regime invisible to standard QA but compounding across the thousands of decisions a deployed agent makes. CBL is orthogonal to known agent-failure axes (adversarial injection, cognitive degradation, multi-agent fault propagation, privacy leakage). We contribute an operational definition, a reusable protocol, a falsifiable prediction set, and a system-class characterization, establishing cross-module interference measurement as a requirement for prompt-composed agent evaluation.

1. Introduction

Prompt-composed agentic systems—where an agent's behavior is assembled at runtime from natural-language modules rather than written as executable code, and a large language model (LLM) interprets the composed context as the agent's policy—are a fast-growing deployment pattern. Practitioners of these systems already report a recurring failure mode: editing one prompt module silently shifts the behavior of an unrelated module, with no shared variable, tool call, or executable dependency to explain it (Subrahmanyam, 2025). The locus of risk is therefore the composition mechanism itself, not any single module. OpenClaw (2026) typifies this deployment pattern at production scale: 379K GitHub stars and over 10K community-authored skill modules, with users composing personality definitions (SOUL.md), capability modules (SKILL.md), and orchestration rules (AGENTS.md) into agent configurations that no single author fully audits.

The same pattern recurs across mechanism choices—markdown convention (Steinberger, 2026; Fernández de Valderrama, 2026), template engines (Wang et al., 2024), and Python class inheritance (Gauthier, 2023)—with non-trivial behavioral logic carried in text the LLM interprets and code restricted to deterministic I/O (§2, Table 1).

The architectural assumption that composing prompt modules yields the composition of their individual behaviors is not warranted by transformer self-attention, which computes global pairwise interactions across the composed context with no module-level isolation: a `### Persona` heading is a writing convention, not a namespace boundary. Proactive interference (Wang & Sun, 2025) and coverage-bounded compositional generalization (Chang et al., 2026) jointly predict that each novel module combination is a distribution shift whose behavioral consequences are not derivable from per-module evaluation.

Yet no existing agent benchmark measures whether modifying one prompt module silently alters a semantically unrelated module's behavior—making compositional behavior under module-set shift a first-order safety and reliability concern for deployed agents.

We make four contributions:

- We give an operational definition of *compositional behavioral leakage* (CBL)—behavioral interference between co-resident prompt modules in a shared attention context—and distinguish it from compositional privacy leakage (Patil et al., 2025), longitudinal cognitive degradation, and adversarial prompt injection.

[1]University of Illinois Urbana-Champaign, Urbana, IL, USA. Correspondence to: Yifan Liu <yifan40@illinois.edu>.

Accepted to the ICML 2026 Workshop on Failure Modes in Agentic AI (FAGEN), Seoul, South Korea. Copyright 2026 by the author(s).

- We instantiate a reusable three-channel protocol on a deployed job-evaluation agent (Claude Sonnet 4.6; 12 JDs, 144 trials), providing an existence proof of cross-module interference ($d = 0.63$ on the semantic-content channel; condition C2, Table 2)—running on artifacts other prompt-composed systems already produce.
- We derive a falsifiable prediction set framing the multi-system replication program, requiring no new measurement infrastructure beyond what current agent deployments emit.
- We place prompt-composed agentic systems on a *composition-mechanism spectrum* across four open-source projects, establishing them as a distinct system class within agentic systems.

2. Prompt-Composed Agentic Systems

2.1. The Pattern

A **prompt-composed agentic system** is one in which non-trivial behavioral decision logic—scoring rubrics, workflow sequencing, persona definitions, tool-selection criteria—is encoded in human-authored text files an LLM *interprets* via attention rather than executes via a runtime.

2.2. Four Systems

Table 1 surveys four open-source prompt-composed agent systems spanning a composition-mechanism spectrum from markdown convention (Steinberger, 2026; Fernández de Valderrama, 2026) through Jinja2 templating (Wang et al., 2024) to Python class inheritance (Gauthier, 2023). In every case behavioral logic resides in text the LLM interprets via attention; code handles only deterministic I/O. As systems mature, composition migrates outward (conventions → templates → class inheritance → prompt-management services)—tacit acknowledgment that text-level composition is fragile. Yet the most widely deployed systems (OpenClaw, AGENTS.md/CLAUDE.md across thousands of repositories) still compose at the text level. Appendix A provides per-system descriptions and the full maturation pattern. The absence of a formal characterization of when and how this fragility manifests motivates the definition we introduce in §3.

3. The Failure Mode: Compositional Behavioral Leakage

3.1. Operational Definition

Let $\mathcal{M} = \{m_1, \ldots, m_n\}$ be a deployed prompt-module set concatenated into an LLM's input context, and let $\mathcal{M}^{(j \leftarrow e)}$ denote the same module set after applying an edit e to a non-focal module m_j. Let $s_i(x; \mathcal{M})$ denote the observable score or decision associated with a focal module m_i on input x.

We say the system exhibits **compositional behavioral leakage** (CBL) when an edit e that is not intended to affect the focal behavior induces a detectable paired shift:

$$\Delta_i(e) = \mathbb{E}_x \left[s_i(x; \mathcal{M}^{(j \leftarrow e)}) - s_i(x; \mathcal{M}) \right],$$

where the paired-effect bootstrap confidence interval excludes zero.

The stricter singleton comparison $\mathcal{B}(m_i \mid \mathcal{M}) \approx \mathcal{B}(m_i \mid \{m_i\})$ is a compositional-consistency test and can be viewed as a special case of this regression view. In this paper, we focus on the deployed-system setting: whether localized edits to non-focal prompt modules silently shift focal module behavior under the full deployed prompt.

CBL differs from three adjacent failure modes. **Compositional privacy leakage** (Patil et al., 2025): multi-agent output aggregation reveals sensitive information (privacy vs. behavior; multi-agent vs. single-agent topology). **Cognitive degradation** (Atta et al., 2025): longitudinal state drift across multi-step execution (runtime accumulation vs. initialization interference). **Prompt injection**: adversarial external inputs subvert behavior (malicious vs. accidental). Practitioners refer to CBL informally as "instruction bleed" (Subrahmanyam, 2025); we formalize it.

3.2. Mechanisms Predicting CBL

Four properties of transformer LLMs make CBL plausible and yield testable predictions. Each contributes a distinct interference mechanism; together they make composition *syntax* itself a behavioral variable.

Attention is global. Transformer self-attention computes pairwise token interactions across the entire input. There is no architectural mechanism restricting attention to tokens within a single prompt module: a `### Persona` heading is a statistical hint, not a scope boundary, and its effectiveness depends entirely on training-distribution priors over similar formatting in pre-training data. Delimiter-based isolation is therefore a behavioral expectation, not a guarantee. This non-isolation property does not by itself prove leakage in any particular system, but it removes the formal basis for assuming that delimiter-separated modules behave independently.

Proactive interference. Wang and Sun (2025) demonstrate that LLM retrieval accuracy declines log-linearly with accumulated interfering context, even when the target is positioned immediately before the query. Prompt-engineering mitigations ("ignore earlier input," explicit scoping instructions) yield limited effectiveness. Earlier prompt

Table 1. Four prompt-composed agentic systems spanning a composition-mechanism spectrum. None asserts behavioral output of prompts in CI.

System	Format	Composition	Files	Beh. tests
OpenClaw	Markdown	Convention	SOUL/SKILL + 10K skills	Manual only
career-ops	Markdown	Convention	18 modes + `CLAUDE.md`	Struct. lints
OpenHands	Jinja2	`{% include %}`	51 `.j2`	Rendering only
aider	Python	Inheritance	19 `_prompts.py`	Parser tests

modules *actively degrade* the processing of later modules—a working-memory bottleneck that single-module testing cannot detect, since the interference manifests only under composition.

Coverage-bounded composition. Chang et al. (2026) formalize that pattern-matching success in compositional tasks correlates with training-distribution coverage (cf. Dziri et al. (2023)). Novel module combinations are unlikely to have coverage in the pre-training distribution, forcing fallback to priors that may blend unrelated modules. For safety-critical deployments, this implies that untested module combinations must be treated as behaviorally unpredictable until isolation mechanisms are validated.

Format sensitivity. Sclar et al. (2024) show that meaning-preserving format changes produce accuracy swings of up to 76 percentage points within a single model. In prompt-composed agent systems, the *syntax* of module composition is itself a behavioral variable: heading hierarchies, list formatting, whitespace, and section ordering are interference channels independent of semantic content. Condition C3 in §3 probes this channel directly.

3.3. Existence Proof on a Deployed Agent

We test for CBL in career-ops, a prompt-composed job-evaluation agentic system. The focal module scores job descriptions on a 1–5 rubric across multiple weighted dimensions. We measure *cv_match*, the dimension most directly capturing the module's core evaluation function: how well a candidate's background aligns with the job requirements.

Conditions. Against an unmodified baseline (C0: the deployed system prompt as shipped), we define three conditions that each perturb the non-focal modules along a distinct channel: **C1** (module addition — *volume*): baseline + an unrelated 200-line recipe-evaluation module. **C2** (semantic modification — *content*): baseline with a semantically irrelevant archetype appended to the shared rules file (Appendix B). **C3** (format perturbation — *form*): baseline with meaning-preserving structural changes (heading levels, emoji markers, section reordering) to non-focal modules.

C2 is the primary test: the edit is localized to a non-focal behavioral specification, while the measured outcome is the focal cv-match score. Under the system's stated modular semantics, the added archetype should not systematically change this score; a paired shift therefore indicates cross-module interference.

Table 2. Per-condition cv-match scores relative to C0 (baseline; mean = 2.72, SD = 1.23). **C2** (semantic interference, primary test) shows a bootstrap 95% CI excluding zero; **C1** and **C3** (controls) do not. Effect sizes (Cohen's d) computed on JD-level paired means ($N = 12$); bootstrap 95% CIs on Δ from 10,000 resamples.

Cond.	Mean	SD	Δ	d	95% CI (Δ)
C1 (volume)	2.69	1.19	-0.03	-0.11	$[-0.19, +0.11]$
C2 (content)	2.89	1.17	$+0.17$	**0.63**	$[+0.03, +0.31]$
C3 (form)	2.64	1.27	-0.08	-0.29	$[-0.25, +0.08]$

Primary outcome. We pre-specify CV-match as the primary outcome on construct-validity grounds: it is the rubric dimension most directly downstream of the Archetype Detection table modified in C2, so cross-module interference, if present, has its highest theoretical prior of manifesting on this dimension.

Protocol. Each condition is evaluated on 12 job descriptions $\times$ 3 independent runs using Claude Sonnet 4.6 (144 total trials). Effect sizes (Cohen's d) with bootstrap 95% CIs on Δ (10,000 resamples) serve as the primary analysis; Wilcoxon signed-rank tests with Bonferroni correction provide secondary confirmation.

Results. Table 2 reports per-condition *cv_match* scores and paired effect sizes with bootstrap 95% CIs.

C2 is the strongest signal: adding a single irrelevant archetype to the shared rules file produces $d = 0.63$ ($\Delta = +0.17$, bootstrap 95% CI $[+0.03, +0.31]$, excluding zero), with 8 of 12 JDs shifting upward. Within-condition ICC for cv-match is 0.925, so paired contrasts at $N = 12$ retain power for the medium-to-large effect sizes the framework predicts under semantic interventions. This shift satisfies our operational criterion for CBL: a localized edit to a non-focal behavioral specification induces a detectable shift in the focal cv-match score.

The control conditions sharpen the claim. **C1** ($d =$

-0.11) and **C3** ($d = -0.29$) yield CIs including zero, ruling out both a generic "any-added-context degrades performance" interpretation and a format-channel explanation, and localizing the detected interference to semantic content—sharpening rather than weakening the CBL claim. No recommendation flips occurred under any condition—a sub-threshold regime the framework predicts under small semantic interventions, developed in §5.

4. Related Work

Agent failure-mode evaluation. AgentDojo (2024) evaluates adversarial injection; QSAF (2025) proposes runtime mitigation for cognitive degradation; MAS-FIRE (2026) evaluates multi-agent fault propagation; Patil et al. (2025) evaluate compositional *privacy* leakage. None measure synchronous behavioral interference between co-resident prompt modules in a single agent's initialization context.

Prompt sensitivity and context interference. Sclar et al. (2024) document accuracy swings from meaning-preserving format changes; PI-LLM (2025) shows log-linear retrieval decline resistant to prompt-engineering mitigations. Our contribution extends these from controlled tasks into real agent behavioral output.

Modular prompting. DecomP (2023) decomposes tasks via multi-call orchestration; ModularPrompt (2022), PRopS (2023), and Smart MoP (2025) achieve modularity through soft prompts and differentiable or softmax gating. Each isolates modules through explicit mathematical or API-level mechanisms; the systems we study compose at the text level via concatenation, with no isolation beyond attention.

Compositional generalization. Skill-Mix (2024) shows frontier models combine skills but degrade as skill count increases. Chang et al. (2026) formalize that compositional generalization is bounded by training-distribution coverage; Dziri et al. (2023) show transformers collapse into pattern matching under compositional depth. Model-level compositional capacity is therefore insufficient to guarantee clean composition when novel module combinations lack coverage.

5. Deployed-System Risks and Predictions

Deployed-system risks. CBL's practical consequence is quiet unreliability, not loud failure: in our experiment no recommendation flipped while scores shifted systematically. Under continuous prompt evolution—community-contributed skills, accumulating CLAUDE.md rules, user-customized archetypes—such silent drift is hard to detect and harder to audit.

Sub-threshold drift as predicted regime. C2's result lies in the regime CBL predicts under small semantic interventions: score-based outputs drift before decision boundaries cross. Sub-threshold drift is invisible to standard QA (which checks decisions, not score distributions), compounds across the thousands of decisions a deployed agent makes, and propagates into downstream ranking, prioritization, and aggregation systems where magnitude matters independently of individual flips. Detecting it *requires* protocols like the C0–C3 framework—the methodological contribution of this paper.

Falsifiable predictions. The CBL framework yields three falsifiable predictions for the multi-system replication program. *(i) Cross-model variation.* Magnitude of cross-module interference should vary substantially across model families, given that within-model format perturbations alone produce large accuracy (up to 76 percentage points) swings (Sclar et al., 2024); this makes model-migration testing a first-class requirement. *(ii) Semantic-distance gradient.* Shifts on rubric dimensions semantically proximate to the modified module should systematically exceed shifts on distal dimensions, providing a graded interpretability test of the attentional-pathway hypothesis. *(iii) Threshold-crossing scaling.* Recommendation-flip rates should scale with intervention magnitude and with cross-module semantic overlap. All three are falsifiable on artifacts other prompt-composed agentic systems already produce, with no new measurement infrastructure required.

Scope and open questions. The case study uses one model and one agentic system—appropriate for a proof-of-concept demonstration of the protocol; the falsifiable predictions above frame the multi-model, multi-system replication program. Whether model providers can offer module-isolation primitives—e.g., separately cached prompt segments with restricted cross-segment attention—or whether global attention makes text-level isolation fundamentally impossible remains open. CBL likely extends to multimodal agents and web-browsing agents that compose instructions with crawled content.

5.1. Regression Testing Framework

Prompt modules treated as software components require regression testing across four categories: *compositional consistency*, *module-interaction regression*, *format-perturbation robustness*, and *model-migration testing* (Chang et al., 2026; Sclar et al., 2024). Conditions C1–C3 instantiate the latter three, providing a reusable protocol; none of the systems in §2 implements any of these tests. Practitioners' informal "prompt sprawl" (Subrahmanyam, 2025) acquires an operational definition.

Compositional consistency. Behavior of a focal module under composition with the full module set should approximate its behavior under singleton context: $\mathcal{B}(m_i \mid \mathcal{M}) \approx \mathcal{B}(m_i \mid \{m_i\})$ across representative module sets. Deviations indicate cross-module interference and should be quantified by paired effect sizes with bootstrap confidence intervals, as instantiated by the C0-vs-C2 contrast in §3.

Module-interaction regression. When a module is added to an existing module set, the behavioral suite for all existing modules must be re-evaluated. Condition C1 in §3 instantiates this test: the focal module's behavior is measured under (existing modules) versus (existing + added module), detecting whether the addition silently shifts non-focal-module behavior.

Format-perturbation robustness. Module behavior should survive meaning-preserving structural changes (heading levels, list formatting, whitespace, section ordering) to co-resident modules. Condition C3 in §3 instantiates this test. Format perturbation is a documented first-class behavioral variable, not a nuisance variable (Sclar et al., 2024).

Model-migration testing. Coverage-bounded composition (Chang et al., 2026) makes interference magnitude training-distribution-dependent, so behavioral drift must be re-measured under model updates. Cross-model differences exceed within-model format perturbations (Sclar et al., 2024), predicting at least comparable variation. Behavioral suites pinned to a specific model version must be re-run on every model upgrade rather than assumed transferable.

Scaling considerations. Pairwise interaction testing scales as $O(n^2)$—prohibitive for OpenClaw's 10K+ community-authored skills, which yield over 50 million pairwise combinations before considering higher-order interactions. Practical testing strategies must therefore use sampling-based approaches: random pair sampling for monitoring, semantic- or attention-distance-weighted sampling for high-risk pairs, or covering-design approaches drawn from combinatorial software testing. None of the systems in §2 currently implements any form of regression test along any of these categories.

6. Conclusion

Compositional behavioral leakage is a measurable failure mode in prompt-composed agentic systems; the protocol we introduce provides an existence proof on a deployed agent and a falsifiable prediction set for broader replication. The phenomenon is consistent with a basic architectural non-isolation property: absent explicit isolation mechanisms, transformer self-attention computes interactions across the full context rather than respecting delimiter-defined prompt-module boundaries. In our case study, a single irrelevant archetype shifted cv-match scores by $d = 0.63$ ($\Delta = +0.17$, bootstrap 95% CI $[+0.03, +0.31]$; Claude Sonnet 4.6) despite no semantic relationship to the focal task. We deliver four artifacts—an operational definition of CBL, a reusable three-channel protocol, a falsifiable prediction set, and a system-class characterization of prompt-composed agentic systems—establishing cross-module interference measurement as a requirement for prompt-composed agent evaluation. With combinatorial module-composition spaces no test suite can exhaustively cover (OpenClaw's 10K+ skills make untested combinations the norm), the framework's predictions are testable on artifacts these systems already produce, with no new measurement infrastructure required.

Impact Statement

This paper identifies a failure mode in prompt-composed agentic systems that may affect the reliability of score-based decisions. In applications where such scores influence outcomes for people—hiring, credit, triage—even sub-threshold behavioral shifts raise fairness concerns. Our contribution is diagnostic: we formalize and measure an existing phenomenon to enable mitigation, rather than introducing new capabilities that could be misused.

References

Atta, H., Baig, M. Z., Mehmood, Y., Shahzad, N., Huang, K., Haq, M. A. U., Awais, M., and Ahmed, K. QSAF: A novel mitigation framework for cognitive degradation in agentic AI, 2025. URL https://arxiv.org/abs/2507.15330.

Chang, H., Park, J., Cho, H., Yang, S., Ko, M., Hwang, H., Won, S., Lee, D., Ahn, Y., and Seo, M. Characterizing pattern matching and its limits on compositional task structures. In *International Conference on Learning Representations (ICLR)*, 2026. URL https://arxiv.org/abs/2505.20278.

Chen, H., Saha, A., Joty, S., and Hoi, S. C. H. Learning label modular prompts for text classification in the wild. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, December 2022. doi: 10.18653/v1/2022.emnlp-main.109. URL https://aclanthology.org/2022.emnlp-main.109/.

Debenedetti, E., Zhang, J., Balunović, M., Beurer-Kellner, L., Fischer, M., and Tramèr, F. AgentDojo: A dynamic environment to evaluate prompt injection attacks and defenses for LLM agents, 2024. URL https://arxiv.org/abs/2406.13352.

Dun, C., Hipolito Garcia, M. D. C., Zheng, G., Awadallah, A. H., Sim, R., and Kyrillidis, A. Sweeping heterogeneity with smart MoPs: Mixture of prompts for LLM task adaptation. *Proceedings of the AAAI Conference on Artificial Intelligence*, 39(16):16426–16434, Apr. 2025. doi: 10.1609/aaai.v39i16.33804. URL https://ojs.aaai.org/index.php/AAAI/article/view/33804.

Dziri, N., Lu, X., Sclar, M., Li, X. L., Jiang, L., Lin, B. Y., West, P., Bhagavatula, C., Le Bras, R., Hwang, J. D., Sanyal, S., Welleck, S., Ren, X., Ettinger, A., Harchaoui, Z., and Choi, Y. Faith and fate: limits of transformers on compositionality. In *Proceedings of the 37th International Conference on Neural Information Processing Systems*, NIPS '23, Red Hook, NY, USA, 2023. Curran Associates Inc.

Fernández de Valderrama, S. Career-Ops: AI-Powered Job Search Pipeline, 2026. URL https://github.com/santifer/career-ops.

Gauthier, P. Aider: AI pair programming in your terminal, 2023. URL https://github.com/Aider-AI/aider. Terminal-based AI coding assistant with Git integration.

Jia, J., Deng, Z., Chen, Z., Wang, Y., and Zheng, Z. MASFIRE: Fault injection and reliability evaluation for LLM-based multi-agent systems, 2026. URL https://arxiv.org/abs/2602.19843.

Khot, T., Trivedi, H., Finlayson, M., Fu, Y., Richardson, K., Clark, P., and Sabharwal, A. Decomposed prompting: A modular approach for solving complex tasks. In *International Conference on Learning Representations (ICLR)*, 2023. arXiv:2210.02406.

Patil, V., Stengel-Eskin, E., and Bansal, M. The sum leaks more than its parts: Compositional privacy risks and mitigations in multi-agent collaboration, 2025. URL https://arxiv.org/abs/2509.14284.

Pilault, J., Liu, C., Bansal, M., and Dreyer, M. On conditional and compositional language model differentiable prompting. In *Proceedings of the Thirty-Second International Joint Conference on Artificial Intelligence*, IJCAI '23, 2023. ISBN 978-1-956792-03-4. doi: 10.24963/ijcai.2023/460. URL https://doi.org/10.24963/ijcai.2023/460.

Sclar, M., Choi, Y., Tsvetkov, Y., and Suhr, A. Quantifying language models' sensitivity to spurious features in prompt design or: How I learned to start worrying about prompt formatting, 2024. URL https://arxiv.org/abs/2310.11324.

Steinberger, P. OpenClaw: Autonomous AI assistant, 2026. URL https://github.com/openclaw/openclaw. Open-source autonomous agent framework supporting LLM-driven workflows.

Subrahmanyam, P. Practical Gemini CLI: Structured approach to bloated GEMINI.md. Google Cloud Blog on Medium, 2025. URL https://medium.com/google-cloud/practical-gemini-cli-structured-approach-to-bloated-gemini-md-360d8a5c7487.

Wang, C. and Sun, J. V. Unable to forget: Proactive interference reveals working memory limits in LLMs beyond context length. In *ICML 2025 Workshop on Long-Context Foundation Models (LCFM)*, 2025. arXiv:2506.08184.

Wang, X., Li, B., Song, Y., Xu, F. F., Tang, X., Zhuge, M., Pan, J., Song, Y., Li, B., Singh, J., Tran, H. H., Li, F., Ma, R., Zheng, M., Qian, B., Shao, Y., Muennighoff, N., Zhang, Y., Hui, B., Lin, J., Brennan, R., Peng, H., Ji, H., and Neubig, G. OpenHands: An Open Platform for AI Software Developers as Generalist Agents, 2024. URL https://arxiv.org/abs/2407.16741.

Yu, D., Kaur, S., Gupta, A., Brown-Cohen, J., Goyal, A., and Arora, S. Skill-Mix: a flexible and expandable family of evaluations for AI models. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2023*, 2024.

A. Extended System Descriptions

This appendix elaborates the four prompt-composed agent systems summarized in Table 1, providing the per-system descriptions and the maturation pattern referenced from §2.

OpenClaw. OpenClaw (2026) (379K GitHub stars) exemplifies the paradigm at maximum scale. Users compose personality files (SOUL.md), capability modules (SKILL.md), and orchestration rules (AGENTS.md) that are concatenated into the system prompt at startup (Steinberger, 2026). Over 10K community-authored skills are installable from a public registry, with no sandboxing between skill contexts—composition is resolved entirely by the LLM's attention mechanism.

career-ops. career-ops (2026), a job-evaluation pipeline, uses 18 markdown mode files with explicit cross-references. All behavioral logic—archetype taxonomy, scoring rubric, decision thresholds—resides in markdown; code handles only I/O.

OpenHands. OpenHands (2024) uses Jinja2 templating to enable modular composition, yet rendered prompts are still concatenated into a single context window before reaching the LLM.

aider. aider (2023) encodes prompts as Python string constants on classes that inherit from a `CoderPrompts` base. This is the most code-like mechanism—but the strings are still natural-language behavioral specifications interpreted via attention.

Maturation pattern. As systems mature, composition migrates out of prompts into code or external services (conventions → templates → class inheritance → prompt-management services)—tacit acknowledgment that text-level composition is fragile. Yet the most widely deployed systems (OpenClaw, AGENTS.md/CLAUDE.md across thousands of repositories) still compose at the text level, with composition resolved at runtime by the LLM's attention mechanism rather than by any explicit isolation primitive.

B. Prompt Manipulation: C0 vs. C2

The shared rules file consumed by the job-evaluation agent contains an *Archetype Detection* table that maps role types to keyword signals. Condition C0 (baseline) uses the original table; Condition C2 appends a semantically irrelevant "Professional Chef" archetype together with a culinary-specific evaluation note. The excerpts below show the `Archetype Detection` section only; all other sections of the rules file are identical across conditions.

C0 — Baseline

C0 — Baseline rules file (excerpt)

```
## Archetype Detection

Classify every offer into one of these types (or hybrid of 2):

| Archetype                | Key signals in JD                                |
|--------------------------|--------------------------------------------------|
| AI Platform / LLMOps     | "observability", "evals", "pipelines", ...       |
| Agentic / Automation     | "agent", "HITL", "orchestration", ...            |
| Technical AI PM          | "PRD", "roadmap", "discovery", ...               |
| AI Solutions Architect   | "architecture", "enterprise", "integration", ...|
| AI Forward Deployed      | "client-facing", "deploy", "prototype", ...      |
| AI Transformation        | "change management", "adoption", ...             |

After detecting archetype, read _profile.md for the user's specific
framing and proof points for that archetype.
```

C2 — Irrelevant-Archetype Addition

Lines marked `[+]` are the only additions relative to C0.

C2 — Modified rules file (excerpt)

```
## Archetype Detection

Classify every offer into one of these types (or hybrid of 2):

| Archetype                | Key signals in JD                                |
|--------------------------|--------------------------------------------------|
| AI Platform / LLMOps     | "observability", "evals", "pipelines", ...       |
| Agentic / Automation     | "agent", "HITL", "orchestration", ...            |
| Technical AI PM          | "PRD", "roadmap", "discovery", ...               |
| AI Solutions Architect   | "architecture", "enterprise", "integration", ...|
| AI Forward Deployed      | "client-facing", "deploy", "prototype", ...      |
| AI Transformation        | "change management", "adoption", ...             |
| Professional Chef /      | "kitchen management", "menu design",          [+]
|   Culinary Specialist    |  "food safety", "culinary arts", ...          [+]

After detecting archetype, read _profile.md for the user's specific
framing and proof points for that archetype.
```

The added archetype is semantically unrelated to any of the twelve job descriptions used in the experiment, yet its presence in the shared context shifts cv-match scores (§3, Table 2).