From Vision to Action: Enabling Real-World Agentic VLMs

Aravilli Atchuta Ram
Visa Inc
ataravil@visa.com

Abstract

Mobile and desktop task automation using agentic vision-language models (VLMs) faces critical safety challenges: processing sensitive UI screenshots containing personally identifiable information (PII), generating potentially harmful GUI actions, and operating autonomously without human oversight. Current approaches like VisionTasker rely on post-execution validation or lack generation-time safety guarantees entirely. We present **SafeAgent-MCP**, a framework combining **semantic-constrained decoding** with context-aware entity detection for generation-time safety and **Model Context Protocol (MCP)** for dynamic policy management. Our system enforces three constraint types: (1) *Entity-level blocking* using context-aware semantic entity recognition to prevent PII leakage from screenshots, (2) *Action-space constraints* via NVIDIA NIM structured generation restricting dangerous GUI operations, and (3) *Policy-aware refusal* leveraging OpenAI's gpt-oss-safeguard for reasoning-based safety validation. SafeAgent-MCP provides the first systematic safety framework for production GUI automation agents combining modular semantic entity recognition with industry-standard constrained generation infrastructure.

1 Introduction

1.1 Motivation and Problem

Agentic vision-language models (VLMs) for mobile and desktop automation represent a rapidly advancing frontier in human-computer interaction. Systems like VisionTasker [Song et al., 2024] successfully automate complex, real-world tasks on Android smartphones. While emerging frameworks target industrial software GUI automation [Lin et al., 2025]. These agents process UI screenshots to understand interface states and generate executable actions including clicks, text inputs, and navigation commands.

However, current agentic VLM systems operate with **zero generation-time safety guarantees**. They must process screenshots that often contain PII and credentials, yet can freely verbalize or copy this content before any safeguards apply. At the same time, they may issue destructive actions (e.g., file deletion, unauthorized access) because the action space is not restricted during decoding, and existing defenses rely on brittle post-hoc filters such as regex-based rules or after-the-fact plan validation, which are costly, over-block legitimate cases (e.g., public DNS addresses), and remain vulnerable to simple reformulations.

1.2 Our Contributions

We propose **SafeAgent-MCP**, providing generation-time safety for agentic VLMs through three innovations:

39th Conference on Neural Information Processing Systems (NeurIPS 2025) Workshop: VLM4RWD.

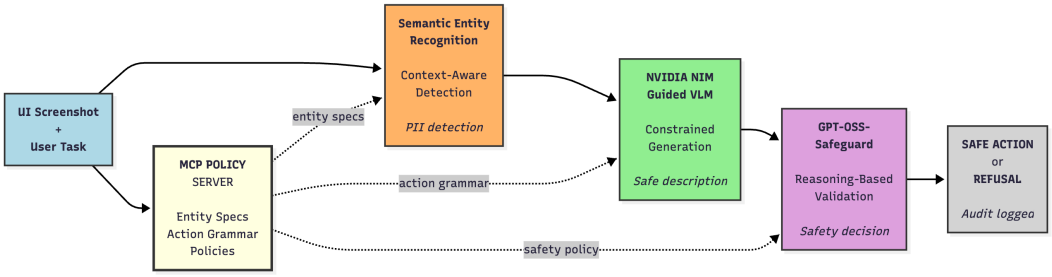

Figure 1: SafeAgent-MCP architecture.

(1) Semantic-Constrained Decoding Architecture: We integrate context-aware semantic entity recognition with NVIDIA NIM's structured generation API [NVIDIA Corporation, 2025] to block PII tokens during VLM decoding. Unlike regex-based approaches, our framework uses semantic understanding to distinguish between sensitive instances and benign mentions, preventing over-blocking while maintaining robust safety.

(2) Policy-Aware Safety Reasoning: We leverage OpenAI's gpt-oss-safeguard [OpenAI, 2025], a 20B parameter reasoning model, to evaluate UI contexts against developer-defined safety policies at inference time, enabling nuanced decisions that pattern matching cannot capture.

(3) MCP-Based Dynamic Policy Management: Building on Model Context Protocol [Anthropic, 2024], we implement a centralized policy server allowing runtime policy updates without agent redeployment—critical for production systems where threat landscapes evolve rapidly.

2 Related Work

Agentic VLMs for GUI Automation—Song et al. [2024] introduced vision-based UI understanding with LLM task planning for mobile automation. Lin et al. [2025] extended this to industrial software using exploration-based learning with multi-layered safety. Both systems validate plans *after* generation, lacking generation-time prevention of unsafe outputs.

Structured Generation for VLMs—NVIDIA NIM [NVIDIA Corporation, 2025] enables JSON schema, regex, and grammar constraints for VLM outputs. HuggingFace's Outlines and vLLM's guided decoding [Willard and Louf, 2023] provide similar capabilities. These tools enforce syntactic correctness but lack semantic safety integration.

PII Detection and Privacy—Traditional approaches use regex patterns (Presidio [Microsoft, 2023]) or fixed-label NER models (BERT-NER [Devlin et al., 2019]). Zero-shot entity recognition enables arbitrary entity types at inference time. RECAP [Rajgarhia et al., 2025] combines pattern matching with context-aware models. Existing systems detect PII in static text, not during generation.

LLM Safety Classifiers—OpenAI's gpt-oss-safeguard [OpenAI, 2025] introduced reasoning-based safety classification where developers provide policies at inference time. Recent work applied grammar-based constraints for secure code generation, but lacks adaptation for multimodal VLMs.

3 SafeAgent-MCP Framework

3.1 Architecture Overview

SafeAgent-MCP uses a three-tier architecture: (1) an MCP policy server that stores entity types, action grammars, and safety rules; (2) a semantic-constrained decoder that combines context-aware entity detection with structured generation; and (3) a policy-aware executor that performs context-level safety validation.

3.2 MCP Policy Server

The MCP policy server maintains three specification types: entity constraints, action constraints, and safety policies. Entity constraints define blocked entity types per domain (for example, account numbers and credit cards in banking, or medical record IDs and prescription numbers in healthcare). Action constraints define context-free grammars that limit safe operations such as `click`, `scroll`, and `read`, while forbidding actions such as `delete`, `uninstall`, and `format`. Safety policies provide natural-language rules for reasoning-based safety classification, for example, "Allow reading notification text, but refuse copying or sharing."

3.3 Semantic-Constrained Decoding

SafeAgent-MCP integrates context-aware semantic entity recognition with structured generation to prevent sensitive entity leakage through three stages.

3.3.1 Stage 1: Context-Aware Entity Detection

Instead of regex patterns that block surface forms, the framework performs semantic entity recognition using contextual embeddings. Given OCR text and domain-specific entity types from the policy server, the system identifies entities based on semantic context. The entity detection module is pluggable, so practitioners can integrate zero-shot NER systems or fine-tuned privacy classifiers. This design distinguishes sensitive instances (for example, "my credit card is 4532-1234-5678-9012") from benign mentions (for example, "Visa cards start with 4"), avoiding over-blocking of legitimate information such as "the IP address of Google's DNS is 8.8.8.8". Adversarial reformulations (for example, spelling out digits) are mitigated using semantic representations and attack patterns that are continuously updated in the policy server.

3.3.2 Stage 2: Dynamic Constraint Synthesis

Rather than static pattern matching, SafeAgent-MCP synthesizes constraints conditioned on detected entities and intent. For each sensitive entity instance, entity-specific blocklists construct token-level exclusions for the exact value and common reformulations, preventing leakage of particular secrets while allowing generic discussion of the entity type. In addition, contextual semantic constraints use entity embeddings to filter candidate tokens whose similarity to detected PII exceeds a threshold, capturing paraphrased leakage that simple pattern matching would miss. The constraint synthesis process adapts to entity type, domain, and task intent (for example, information retrieval versus sensitive data operations).

3.3.3 Stage 3: Intent-Aware Constrained Generation

A lightweight intent classifier analyzes the user task before applying constraints. For legitimate information queries such as "find public DNS servers" or "look up store contact info", constraints apply only to entities visually detected in screenshots that correspond to user private data, not to public information retrieved from knowledge sources. For sensitive data tasks involving forms, banking apps, or healthcare UIs, all instances of specified entity types are constrained, since PII leakage risk outweighs potential utility. This three-stage design aims to avoid over-blocking, remain robust to reformulation, and support practical deployment through domain-specific tuning.

3.4 Action-Space Constraints via Grammar-Based Generation

For task planning, grammar-based constraints limit the action vocabulary. The policy server provides context-free grammars that encode safe operations for specific UI states, and the language model is restricted to sequences that parse under the provided grammar. This prevents entire classes of dangerous commands, even in the presence of prompt injection attempts.

3.5 Policy-Aware Safety Reasoning

Before executing generated actions, a reasoning-based safety model receives the developer-provided safety policy, the current UI description, and the planned action sequence. It outputs a binary SAFE or UNSAFE judgment together with a structured rationale. This allows the system to handle nuanced

Algorithm 1 SafeAgent-MCP Execution

Require: Screenshot S, user task T, app domain D
Ensure: Executed action A or refusal reason R

$E_{\text{blocked}} \leftarrow$ PolicyServer.GetEntityConstraints(D)
$x_{\text{ocr}} \leftarrow$ ExtractText(S)
$intent \leftarrow$ ClassifyIntent(T, D)
$E_{\text{detected}} \leftarrow$ SemanticEntityRecognition($x_{\text{ocr}}, E_{\text{blocked}}$)
$constraints \leftarrow$ SynthesizeConstraints($E_{\text{detected}}, intent$)
$u \leftarrow$ VLM.GenerateWithConstraints($S, constraints$)
$G_{\text{action}} \leftarrow$ PolicyServer.GetActionGrammar(u)
$A_{\text{next}} \leftarrow$ LLM.GenerateWithGrammar(u, T, G_{action})
$P_{\text{safety}} \leftarrow$ PolicyServer.GetSafetyPolicy(T)
$r \leftarrow$ SafetyModel.Classify($P_{\text{safety}}, u, A_{\text{next}}$)
if r.conclusion == SAFE **then**
    Execute(A_{next})
    return A_{next}
else
    Log(r)
    return R {refusal with reasoning}
end if

cases that grammars alone cannot capture, such as distinguishing between reading and sharing the same UI element or between local and remote data transfers. If the action is classified as UNSAFE, the system logs the reasoning for audit and returns a structured refusal; if SAFE, execution proceeds.

3.6 End-to-End Workflow

Algorithm 1 summarizes the SafeAgent-MCP pipeline.

This workflow provides multi-layer safety: semantic entity recognition reduces PII leakage, grammar constraints restrict the action space, and reasoning-based models add a final semantic validation layer before execution.

4 Limitations and Future Work

SafeAgent-MCP represents a step toward safer agentic VLMs, but several limitations and open challenges remain. Current semantic entity recognition operates on OCR text only; extending the framework with vision models for visual PII (faces, IDs, sensitive charts) would enable truly multimodal constraints. While constraints are enforced via regex- and grammar-based mechanisms, formal methods for constrained decoding could strengthen guarantees on which unsafe behaviors are provably excluded. The added latency (on the order of hundreds of milliseconds per step) is acceptable for many GUI tasks but may require distillation, parallelization, or lighter safety models for latency-critical deployments. Finally, like all safety systems, SafeAgent-MCP can be further hardened against sophisticated adversarial attacks through adversarial training, semantic stealth detection, and learning-based constraint synthesis.

5 Conclusion

We introduced SafeAgent-MCP, a framework that combines semantic-constrained decoding, context-aware entity detection, structured generation, reasoning-based safety classification, and MCP-based dynamic policy management for safe agentic VLM deployment. By reducing PII leakage during generation, restricting the action vocabulary via grammars, and applying context-aware safety checks, SafeAgent-MCP offers a practical, modular safety layer compatible with existing GUI automation infrastructure. As agentic VLMs move from prototypes to production systems that handle sensitive user data, such generation-time safety mechanisms become essential infrastructure.

LLM Contribution Statement

LLMs were used to polish manuscript writing and gather related work. Framework design, algorithm development, and all research contributions are by the authors.

References

Anthropic. Model context protocol. Anthropic Documentation, 2024. URL `https://www.anthropic.com/news/model-context-protocol`.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 4171–4186, 2019.

Liangtao Lin, Zhaomeng Zhu, Tianwei Zhang, and Yonggang Wen. Inframind: A novel exploration-based gui agentic framework for mission-critical industrial management. *arXiv preprint arXiv:2509.13704*, 2025.

Microsoft. Presidio: Context aware, pluggable and customizable pii anonymization, 2023. URL `https://microsoft.github.io/presidio/`.

NVIDIA Corporation. Structured generation for vision-language models. NVIDIA NIM Documentation, 2025. URL `https://docs.nvidia.com/nim/vision-language-models/latest/structured-generation.html`.

OpenAI. Introducing gpt-oss-safeguard. OpenAI Blog, October 2025. URL `https://openai.com/index/introducing-gpt-oss-safeguard/`.

Harshit Rajgarhia, Suryam Gupta, Asif Shaik, Gulipalli Praveen Kumar, Y Santhoshraj, Sanka Nithya Tanvy Nishitha, and Abhishek Mukherji. An evaluation study of hybrid methods for multilingual pii detection. *arXiv preprint arXiv:2510.07551*, 2025.

Yunpeng Song, Yiheng Bian, Yongtao Tang, Guiyu Ma, and Zhongmin Cai. Visiontasker: Mobile task automation using vision based ui understanding and llm task planning. In *Proceedings of the 37th Annual ACM Symposium on User Interface Software and Technology*, pages 1–17, 2024.

Brandon T Willard and Rémi Louf. Efficient guided generation for large language models. *arXiv preprint arXiv:2307.09702*, 2023.