From Vibe to Verifiable Spec-Driven Development: A Demo of Intent and Realization Engineering

Keheliya Gallaba[1], Zhiyu Fan[1], Jiahuei (Justina) Lin[1], Filipe R. Cogo[1], Benjamin Rombaut[1], Dayi Lin[1], Ahmed E. Hassan[2]

[1]Centre for Software Excellence, Huawei Canada
[2]Queen's University, Canada
keheliya.gallaba@huawei.com, zhiyu.fan@huawei.com, justina.lin@huawei.com, filipe.roseiro.cogo1@huawei.com, ben.rombaut@huawei.com, dayi.lin@huawei.com, ahmed@cs.queensu.ca

Abstract

The proliferation of foundation models has given rise to "vibe coding," a paradigm where end-users create software by describing desired outcomes in natural language. While platforms that enable this approach are democratizing software development, they often produce simplistic, fragile applications that lack the formal guarantees required for scalable, reliable systems. The core software engineering challenges of translating ambiguous user intent into formal specifications and managing the lifecycle of AI-generated code remain largely unaddressed. This demo presents a system built on the Software Engineering for Software Makers (SE4SM) framework, which introduces a disciplined, two-pillar approach to AI-assisted development. The first pillar, **Intent Engineering**, employs a collaborative human-AI dialogue to transform a user's high-level concept into a set of clear, unambiguous, and verifiable specifications while considering both explicitly stated needs and implicitly inferred requirements. The second, **Realization Engineering**, uses a spec-driven, multi-agent system to autonomously build, test, and verify the software. This methodology preserves the accessibility of vibe coding while embedding professional engineering practices, producing more trustworthy, evolvable systems.

Introduction

The advent of powerful foundation models is catalyzing a paradigm shift in software creation, moving towards a more democratized and accessible process often referred to as "vibe coding" (Karpathy 2025). This movement, powered by a new generation of AI-native tools, allows end-user software developers (Ko et al. 2011), or "software makers," to build applications by describing their goals in natural language, abstracting away the complexities of traditional programming. While this approach significantly lowers the barrier to software creation and accelerates prototyping, it concurrently introduces critical challenges in ensuring the reliability, maintainability, and trustworthiness of the resulting applications (Sabouri et al. 2025).

The core problem lies in the inherent ambiguity of natural language and the unstructured nature of the development process. Current "vibe coding" platforms often translate a user's initial, often vague, prompt directly into code, bypassing the crucial software engineering phases of requirements elicitation, specification, and verification. This can result in software that is functionally correct for a narrow set of assumed conditions but is brittle, insecure, and lacks the formal guarantees necessary for scalable, production-grade systems. Furthermore, the resulting codebases are often monolithic and difficult to evolve, as the underlying design rationale and user intent are not formally captured. These issues mirror long-standing challenges in requirements engineering, now amplified by the direct-to-code nature of AI-driven generation.

Our demo presents the *Software Engineering for Software Makers (SE4SM)* framework, a disciplined, two-pillar approach designed to bridge the gap between the intuitive, "vibe-driven" creation process and the rigor of professional software engineering. We address the challenges of ambiguity and lack of formalism through **Intent Engineering**, a process where a collaborative AI system engages the user in a structured dialogue to transform a high-level idea into a set of clear, verifiable specifications. To address the challenge of unconstrained and unreliable code generation, we introduce **Realization Engineering**, a spec-driven, multi-agent architecture that autonomously builds, tests, and verifies the software against the user-validated specifications.

We demonstrate a system that operationalizes the SE4SM framework. We present a methodology that retains the accessibility of "vibe coding" while embedding the essential engineering practices of specification, verification, and automated testing. This approach paves the way for a new class of development tools that can empower non-technical users to create not just simple applications, but robust, evolvable, and trustworthy software systems.

Intent Engineering: Eliciting Verifiable Specifications

A primary obstacle in user-driven software generation is resolving the inherent ambiguity of natural language. Requirements elicitation, a known bottleneck in software engineering, is amplified when the user is non-technical and the developer is an AI. Our Intent Engineering phase addresses this challenge not through simple conversational Q&A, but through a structured, interactive process grounded in principles from Theory of Mind (ToM) (Wang et al. 2021), designed to build a robust, shared understanding between the

user and the AI system.

Instead of passively translating user statements, our system actively models the user's mental state, which includes their beliefs, desires, and unstated intentions. This enables a deeper level of collaboration, which manifests in the following capabilities:

ToM-Driven Intent Elicitation: The system moves beyond literal interpretations of the user's prompts. It leverages its ToM framework to form hypotheses about the user's underlying goals. This process allows the system to differentiate between two crucial types of requirements:

- **Explicit Requirements**, which are directly articulated and confirmed by the user during the dialogue (e.g., "I need a button to submit the form").
- **Inferred Requirements**, which are needs the AI deduces based on the user's goals, domain conventions, and non-functional necessities like security or usability (e.g., "For a form with a password field, the system should implement password recovery").

Crucially, all inferred requirements are surfaced to the user for validation (e.g., "It looks like you are creating user accounts. Should I add a 'Forgot Password' feature?"). This unique mechanism makes implicit assumptions explicit, allowing the user to make informed decisions about features they might not have known to ask for.

Generation of Auditable Artifacts: The entire conversational process is captured, creating a transparent and traceable record of decisions and rationales. This "living documentation" clearly delineates between explicit and inferred requirements, serving as an essential specification for both the Realization Engineering phase and future software maintenance.

Intermediate Representation (IR): The system guides the user to define explicit success criteria and operational scenarios in natural language. These are automatically translated into artifacts such as use cases, requirements, and expected outcome representations, so they can be communicated from the user-facing Intent Engineering phase to the primarily agent-powered Realization Engineering phase. With inspiration from formats such as Gherkin (Wynne and Hellesoy 2012) and FRET (Sheridan et al. 2025) we have designed an IR that balances human readability with machine processability. This ensures the resulting specifications are not only clear but also verifiable, providing concrete guarantees for the final application.

Realization Engineering: A Multi-Agent, Spec-Driven Architecture

Upon user validation of the formal specifications, the system transitions to the Realization Engineering phase. This component is architected as a specialized Agent Execution Environment (AEE), where a team of AI agents collaborate on the development process. This phase is strictly spec-driven, ensuring that the generated software adheres precisely to the user-approved requirements, thereby mitigating the risks of unconstrained AI code generation. The demonstration will present the architecture of this multi-agent system, which features:

Collaborative, Specialized Agents: The AEE consists of distinct agents with specialized roles, including ① a *Spec Agent* that transforms the natural language use cases into a set of formal OpenAPI specifications covering the core functionalities of the software; ② a *Test Plan Agent* that derives high-level natural language test cases from these specifications, describing the desired behaviors of the software; ③ a *Code Agent* responsible for software implementation by following the specifications and test cases provided by the preceding agents; and ④ a *Debug Agent* that verifies the generated code against specifications and tests, and automatically revises any misalignments detected. This modular separation of concerns enhances the robustness and scalability of the autonomous development process.

Test-Driven Development for Vibe Coding: The system operationalizes a test-driven development (TDD) paradigm for reliable AI-driven code generation. Specifically, the *Test Plan Agent* first generates comprehensive test cases from the specifications before any code is written. Then, the *Code Agent* implements the functionality to satisfy both the formal specifications and the pre-defined test cases. This test-first approach provides several advantages: ① it establishes concrete, executable criteria for correctness before implementation, reducing ambiguity in AI-generated code; ② it creates a specification-to-implementation traceability chain, ensuring that every aspect of the user's validated intent is reflected in both tests and code; and ③ it enables continuous automated verification throughout the development process, allowing AI agents to autonomously detect and correct deviations from the intended behavior without human intervention. This disciplined, specification-driven workflow transforms vibe coding from an abstract concept into a rigorous engineering practice.

Automated Verification and Feedback Loops: The *Debug Agent* manages a continuous verification loop. First, it checks for static issues (e.g., syntax errors, import failures, and API usage discrepancies) and revises the code until they are resolved. Next, it executes the code against the unit tests in a secure sandbox. The agent iteratively revises the code to fix any test failures until all tests pass, ensuring convergence between the implementation and the user's validated intent. This enables fully autonomous, specification-compliant code generation.

Conclusion

We have presented the SE4SM framework, which bridges the gap between the accessibility of vibe coding and the discipline of software engineering. By formalizing the translation of user intent into verifiable specifications and then employing a structured, multi-agent architecture for implementation, our system offers a viable path toward producing reliable, maintainable, and trustworthy AI-generated software. This demo will provide an end-to-end walkthrough of this process in action.

References

Karpathy, A. 2025. There's a new kind of coding I call "vibe coding". https://x.com/karpathy/status/1886192184808149383. Accessed: 2025-10-29.

Ko, A. J.; Abraham, R.; Beckwith, L.; Blackwell, A.; Burnett, M.; Erwig, M.; Scaffidi, C.; Lawrance, J.; Lieberman, H.; Myers, B.; Rosson, M. B.; Rothermel, G.; Shaw, M.; and Wiedenbeck, S. 2011. The state of the art in end-user software engineering. *ACM Computing Surveys*, 43(3): 1–44.

Sabouri, S.; Eibl, P.; Zhou, X.; Ziyadi, M.; Medvidovic, N.; Lindemann, L.; and Chattopadhyay, S. 2025. Trust Dynamics in AI-Assisted Development: Definitions, Factors, and Implications. In *2025 IEEE/ACM 47th International Conference on Software Engineering (ICSE)*, 1678–1690. IEEE.

Sheridan, O.; Becker, L. B.; Farrell, M.; Luckcuck, M.; and Monahan, R. 2025. *Sharper Specs for Smarter Drones: Formalising Requirements with FRET*, 350–362. Springer Nature Switzerland. ISBN 9783031885310.

Wang, Q.; Saha, K.; Gregori, E.; Joyner, D.; and Goel, A. 2021. Towards Mutual Theory of Mind in Human-AI Interaction: How Language Reflects What Students Perceive About a Virtual Teaching Assistant. In *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems*, CHI '21, 1–14. ACM.

Wynne, M.; and Hellesoy, A. 2012. *The Cucumber Book: Behaviour-Driven Development for Testers and Developers*. Pragmatic Bookshelf. ISBN 1934356808.