Conceptual Architecture Report

CISC 322 - A1 Group 25

https://youtu.be/y5R8GtzhM1s

Group Members & Roles

Thomas Schrappe - Presenter, slides, sequence diagrams

Adrian Yanovich - Presenter, slides, External Interfaces, Implications for division of responsibilities, Evolvability and Modifiability

Colin Mclaughlin (Group Leader)- Abstract, Introduction, Use cases & Functionality, AI Report

Jaiman Sharma - Control & Data Flow Among Parts, Concurrency, Alternative Architectural Styles Considered

Mantaj Toor - Conceptual Architecture

Lingwei Huang - Member

VOID

- Open Source AI code editor
- Goal - Provide access to LLMs while maintaining complete control of data
- Tab autocomplete, in line editing, AI chat, access to variety of LLMs
- Bridges traditional IDE code editors and LLM capabilities

Why is Void Different?

- Uses VSCode Foundation
- Community Features
- Open-Source
- Privacy First Approach

Components & Conceptual Diagram

Platform Services. Core foundation: config, extensions, IPC, themes, telemetry

UI Components. AI interfaces: chat panel, menus, feature rendering

AI Services Request processing: code analysis, suggestions, tool management

Provider Integration. Unified API layer: multiple providers, custom endpoints

Extension System. VS Code compatibility: AI extension points, lifecycle

Database. Persistent storage: config, history, context

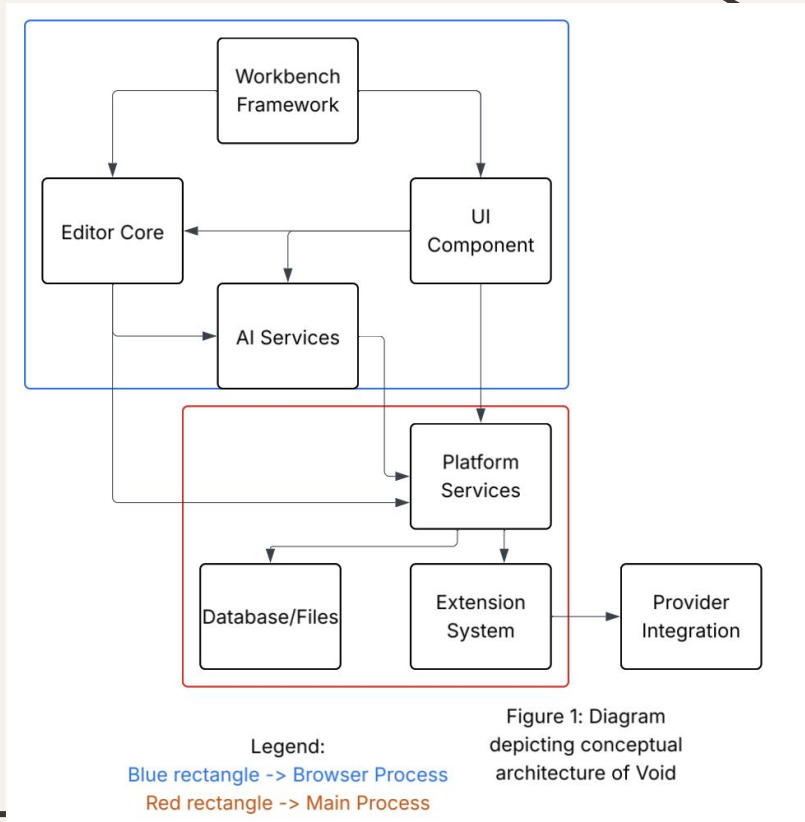


Figure 1: Diagram depicting conceptual architecture of Void

Derivation Process

- Document and Reference Study
- Use Case Identification
- Component and Dependency Extraction
- Validation Against Reference Architecture

Architectural Overview

- Main style: Layered
- Secondary Styles: Implicit invocation + Client Server

Architectural decisions:

- Why Layered Architecture?
- Separates concerns: UI, services, AI providers in distinct layers
- Alternative: Monolithic (rejected—poor modularity)

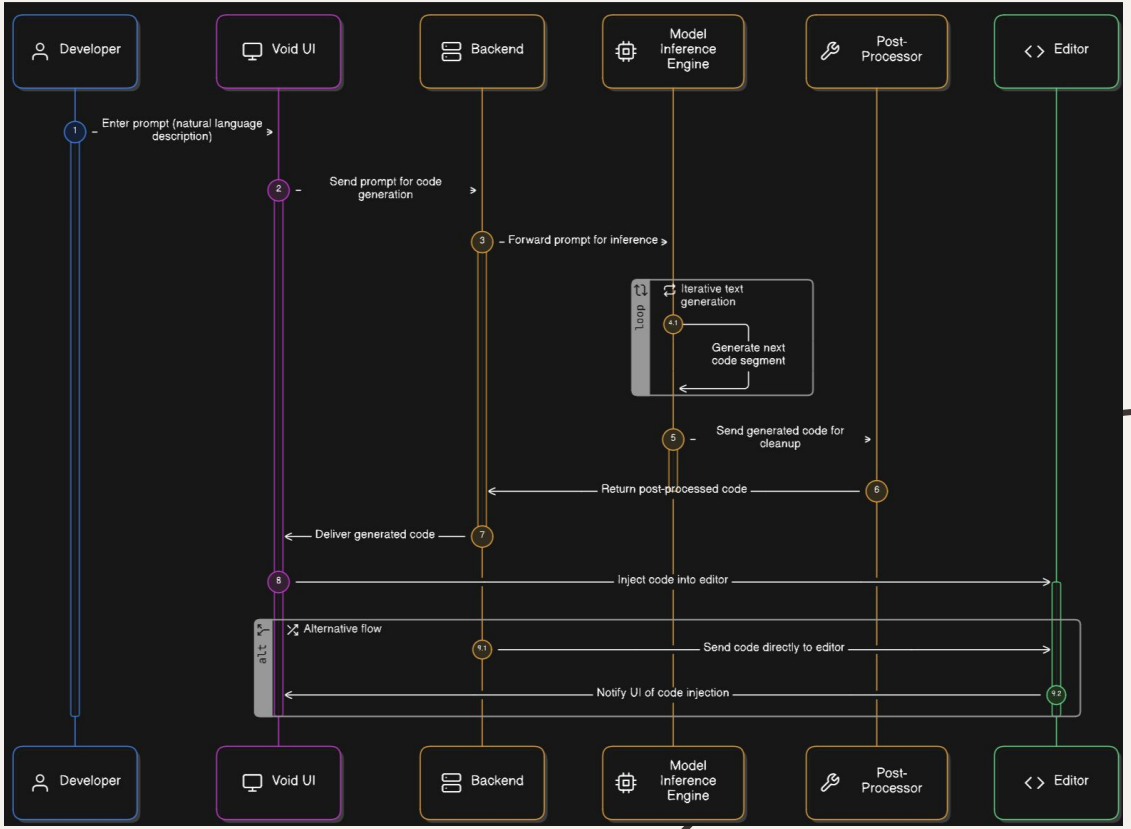
Why Client-Server (Electron)?

- Process isolation keeps AI operations separate from core IDE
- Prevents crashes in one component from affecting others

Use Case 1 – Chat-Based Code Generation

For this use case, the user could prompt Void to generate code by supplying a description of the code/what it should do.

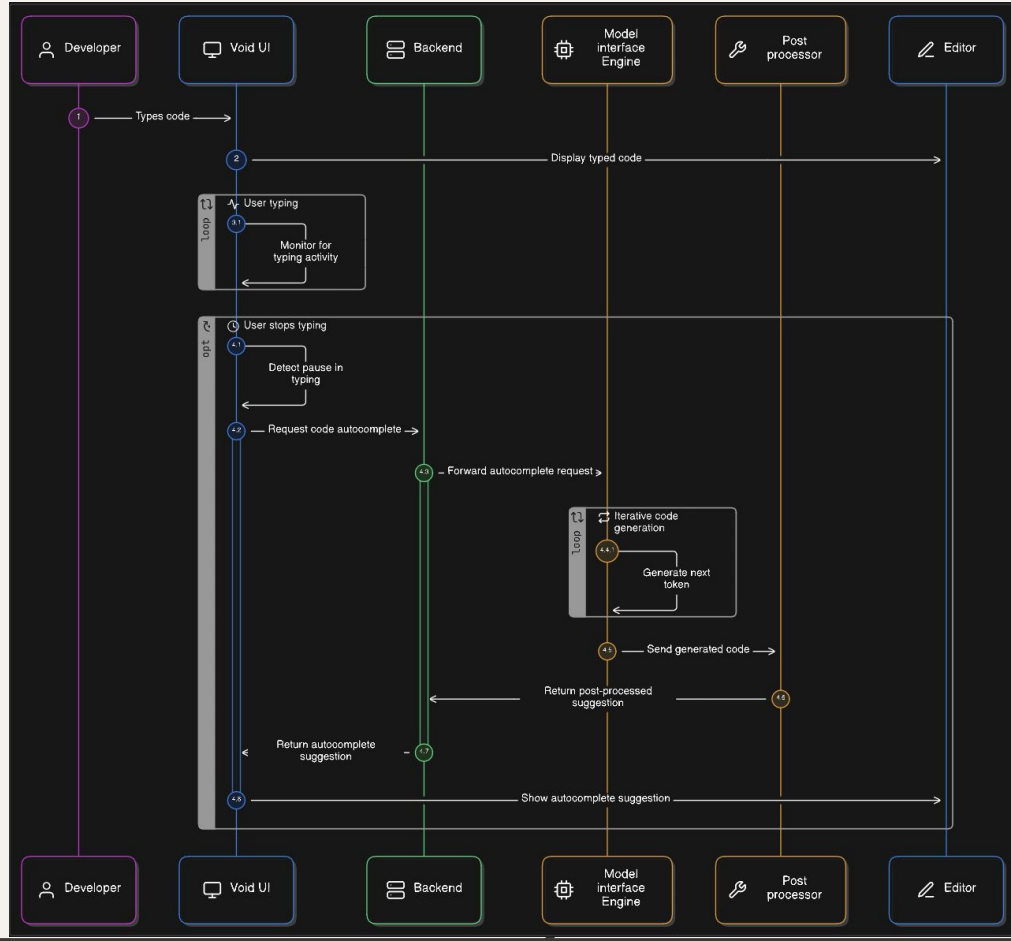
Void would then generate code and supply it to the front end.



Use Case 2 – Autocomplete Request Flow

For this use case, Void software supplies the user with code suggestions as the user is writing code. When the user stops typing for a specified amount of time, Void suggests autocomplete code.

This works similarly to the last example, user code is sent to backend,



Trade-offs & Quality Attributes

Attributes:

- Evolvability: easy to adapt Void to newer, better language models and integrate them
- Performance: Asynchronous processes allow Void to run LLM requests in background while user writes code

Trade-offs/Limitations:

- Errors and hallucinations
- Vulnerabilities introduced by AI replicating vulnerable code

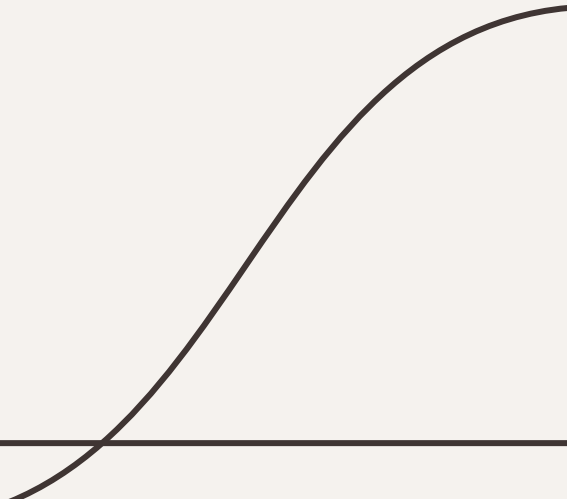
External Interfaces

- Void communicates with external interfaces through two main channels
- User components - chat panel, completion list, settings view
- AI model providers - OpenAI, Ollama APIs, through WebSockets and HTTP
- Through these channels Void is able to forward information from user to AI models to be processed and returned with an appropriate output

Lessons Learned

Lessons Learned:

- Separate components in different layers for isolating work
- Cloud computation for heavy computations (LLM)
- Freedom of choice with different LLMs provides user freedom and modularity



AI Teammate contributions & control

Used AI for:

- ChatGPT: After writing down sections, we used AI to condense the text and keep the same tone throughout the report.
- Claude: After finishing report, used AI to condense report into bullet points for the presentation slides
- Eraser: After creating rough examples of sequence diagrams, AI was used to create a clean, professional sequence diagram.

Overall impact of AI was relatively low. AI did not generate any significant parts of the project, only refined the parts we already had.

Questions