Individual Addendum — Miles Keffer

Miles Keffer

December 10, 2025

Individual Addendum — Miles Keffer

1. Personal Contribution

My contributions focused primarily on the **parser**, the **language grammar**, and the **project documentation**, along with writing the testing scripts used during development. I worked to ensure that the grammar was expressive enough to support the features we intended for Vulpes, while remaining small, teachable, and predictable for students.

Specific tasks completed

- Designed and iterated on the **formal grammar** of the Vulpes language.
- Implemented the **parser**, including handling of expressions, statements, function definitions, namespacing, and optional type/parameter annotations.
- Collaborated closely with the AST developer to ensure each grammar rule mapped to a clear AST representation.
- Developed **testing scripts** to exercise parsing behaviors and catch early regressions.
- Contributed significantly to the **documentation**, improving readability, examples, and syntax explanations.
- Assisted with refining sample `.vlp` files and validating parser correctness across many constructs.

Role in the project

My role centered on owning the **syntactic structure of the language**, ensuring that the parser was robust, consistent, and aligned with our educational goals.

---

2. Skills Applied and Gained

Technical skills applied

- Implementing recursive-descent and pattern-driven parsing strategies.
- Designing clear and maintainable grammar rules.
- Working with C++17 in a multi-module compiler project.
- Improving documentation clarity for technical audiences.

Skills gained or strengthened

- A deeper understanding of **language design trade-offs**, particularly where convenience features affect grammar complexity.
- Improved debugging strategies for parser-level issues such as precedence, associativity, and ambiguity.
- Stronger collaboration skills, especially coordinating subsystem interfaces with the lexer, AST, and codegen developers.
- Greater confidence in authoring **technical documentation** that is both accurate and accessible.

---

3. Challenges Faced

Balancing expressiveness with simplicity

Supporting optional parameter names, optional return types, and shorthand constructs made the grammar more complex than initially expected. Ensuring the parser remained comprehensible and maintainable required several iterations and refactoring passes.

Avoiding hidden ambiguities

Some features introduced subtle parsing ambiguities, especially in expression parsing and function signatures. These issues required careful ordering of rules and consistent use of precedence and associativity to avoid surprising behavior.

---

4. Reflection

What went well

- The final parser is **reliable, predictable, and readable**, which aligns strongly with the course's teaching objectives.
- Collaboration within the team was smooth; regular communication allowed grammar, AST, and codegen to evolve together rather than diverge.
- Documentation improvements resulted in a clear language specification that future students can learn from quickly.

What I would do differently

- Begin developing automated parser tests earlier to support faster iteration.
- Establish formal grammar diagrams sooner (EBNF, flowcharts) to avoid some of the early ambiguities.
- Push for earlier alignment on convenience features to reduce mid-project redesigns.

---

5. Teamwork Evaluation

I worked consistently and constructively with the team, communicating changes to the grammar and parser behavior whenever they affected downstream components. Collaboration with Sam in particular was productive, as our responsibilities intersected at AST boundaries and expression semantics.

Our team communicated well, coordinated responsibilities effectively, and iterated quickly when issues arose. The working relationship remained positive and efficient throughout the project.
