Gives a framework for composable verification [1]. Proof scripts follow the same modular structure as the actual code, and compose in a similar manner.

How do the proofs compose? Each component documents its assumptions in an interface. New components must satisfy this interface to work together. The approach is implemented in Coq by modeling the Java language.

Strengths

- Suggests a discipline, not a particular framework or library.
- Old proofs do not need to be re-checked as new features are added.

Weaknesses

- Proof assumptions need to find a sweet spot between generality and domain-specifics because new features need to fit within these assumptions. (Unless you go back and refactor.)
- Need to work in Coq, and build your codebase from the ground up.

References

[1] Benjamin Delaware, William R. Cook, and Don Batory. Product lines of theorems. In *OOPSLA*, 2011.