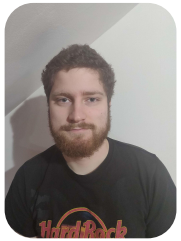
Feature-Induced Architecture Violations



Simon Friedel

Sebastian Böhm

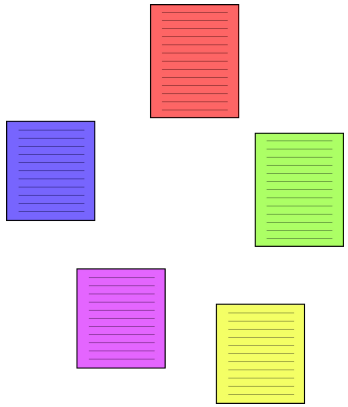
Florian Sattler

Sven Apel

Architecture

Manage high complexity

Architecture

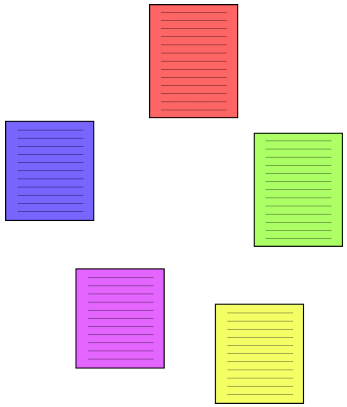


Manage high complexity

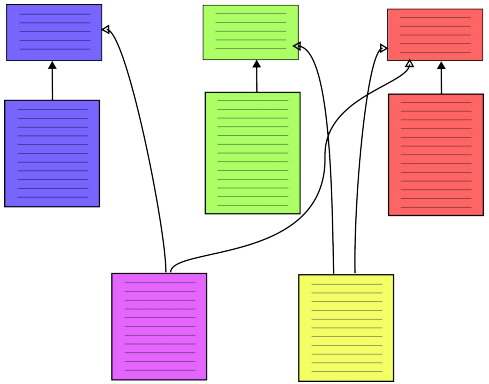
Split into modules

Architecture

Manage high complexity



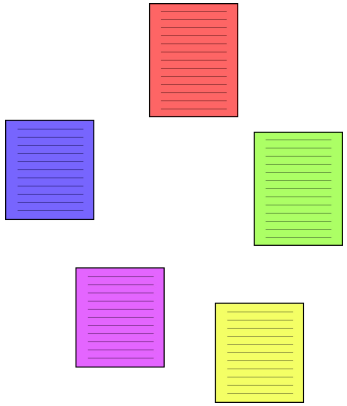
Split into modules



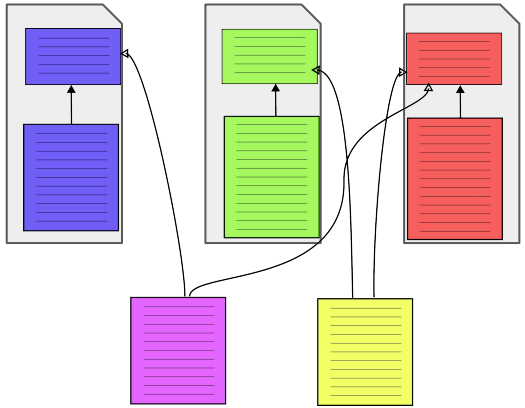
Interfaces to connect independent parts

Architecture

Manage high complexity



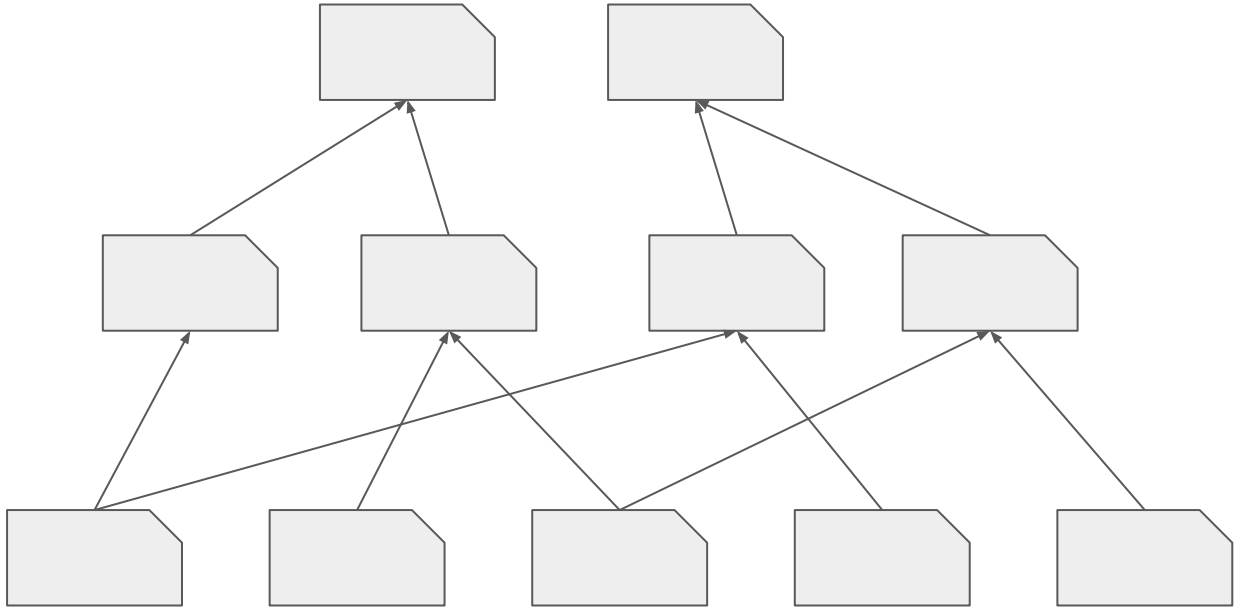
Split into modules



Interfaces to connect independent parts

Architectural Hierarchy

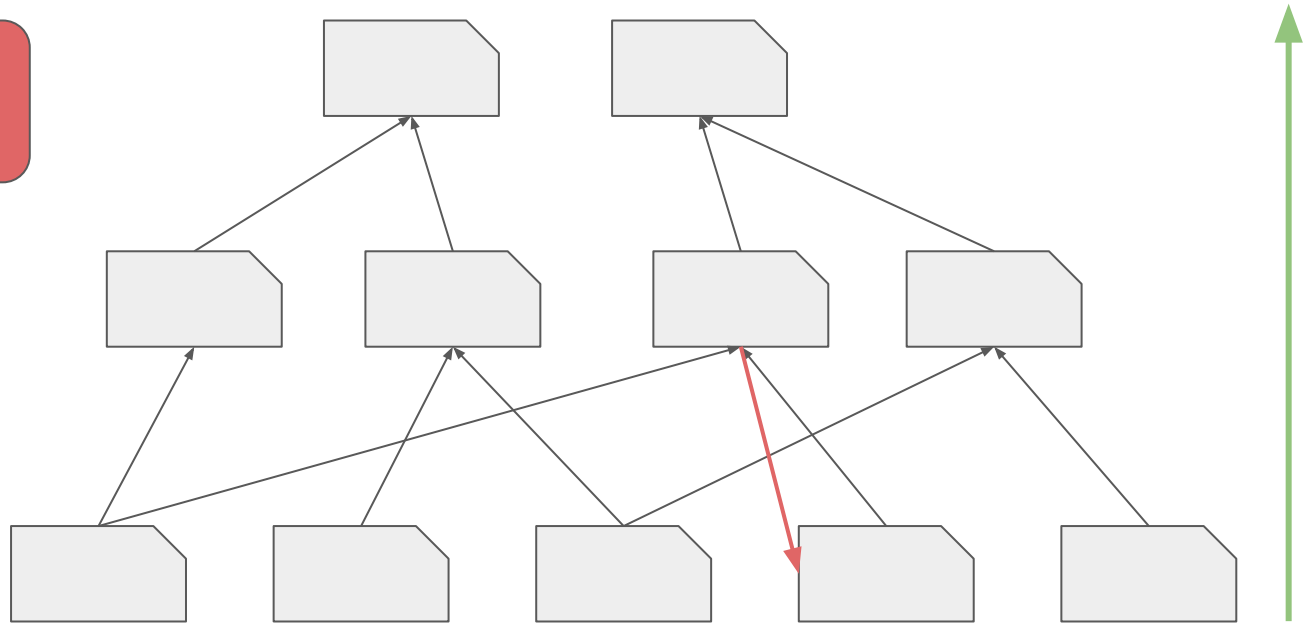
Natural hierarchy through dependencies and inheritance



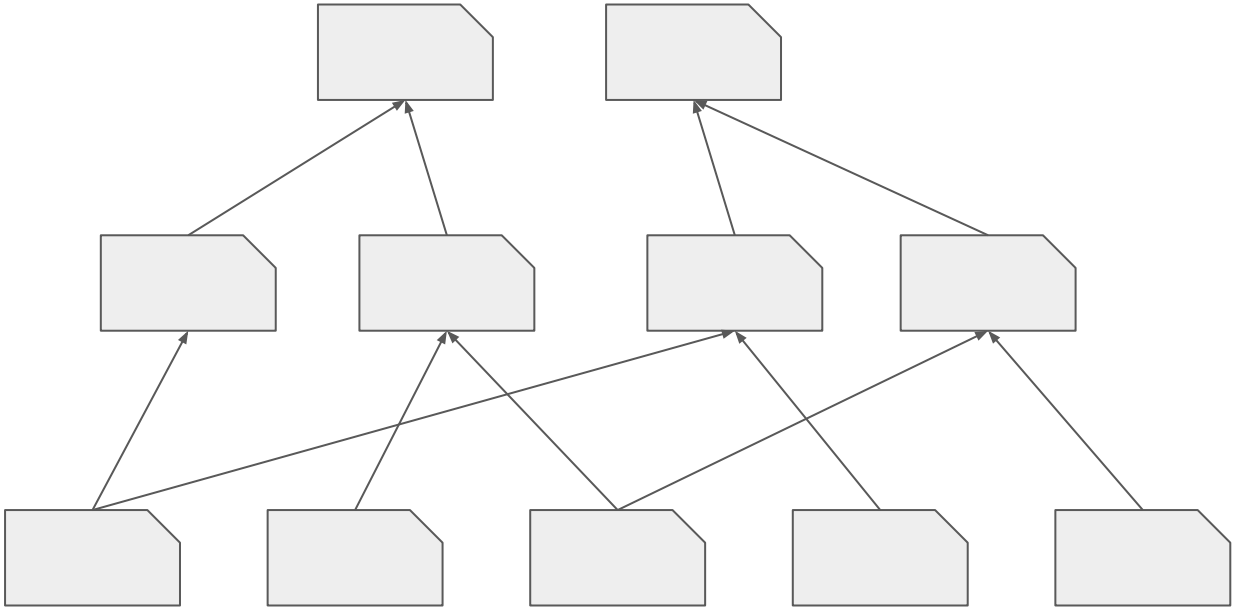
Architectural Antipattern

Dependency on lower layer

Cycles



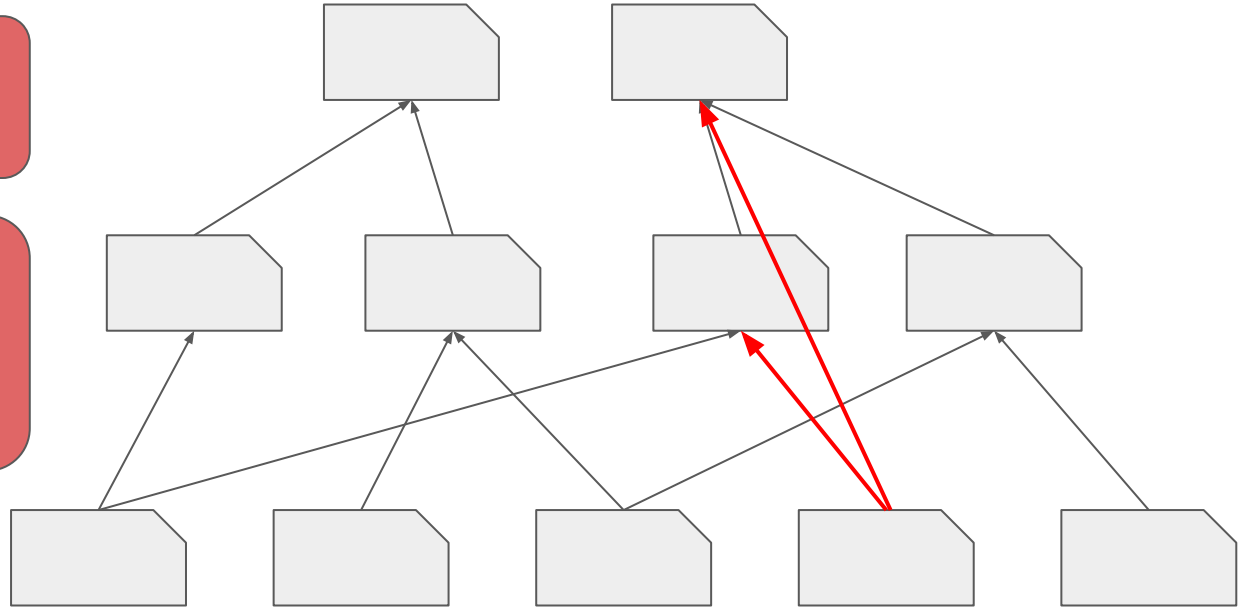
Architectural Antipattern



Architectural Antipattern

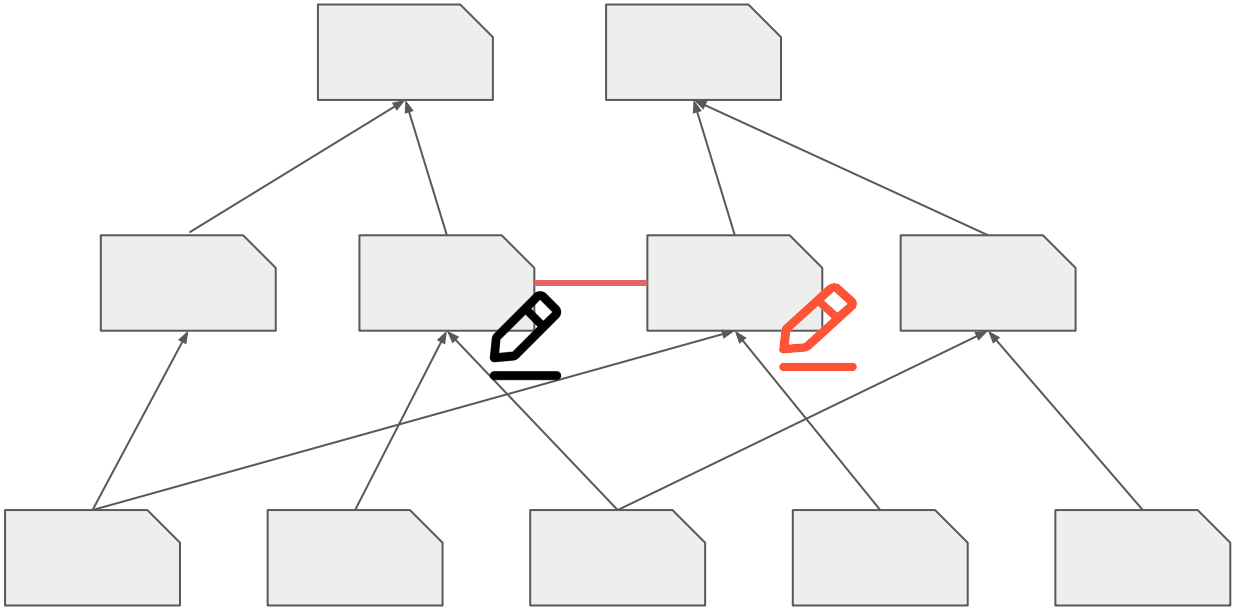
Unhealthy Inheritance

Dependency on multiple parts of an inheritance hierarchy



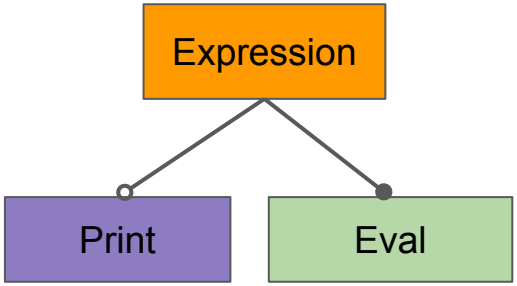
Architectural Antipattern

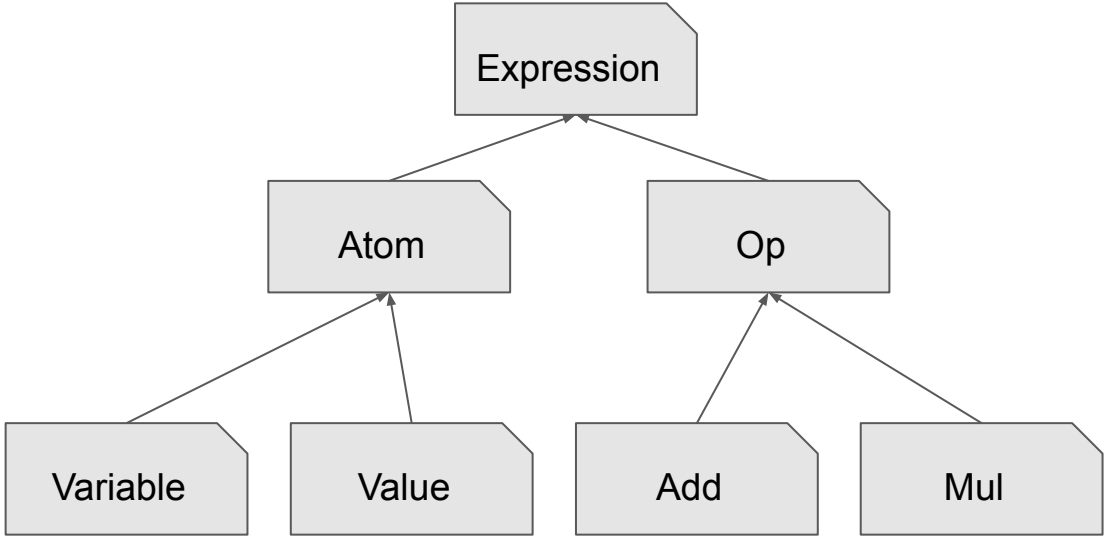
Connection between independent modules



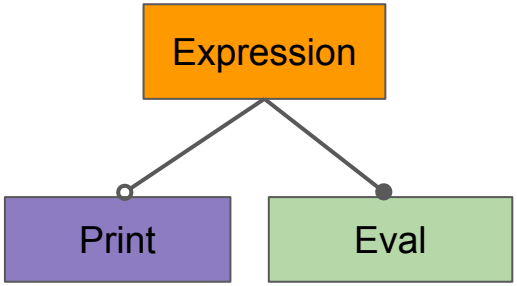
Architecture and Features

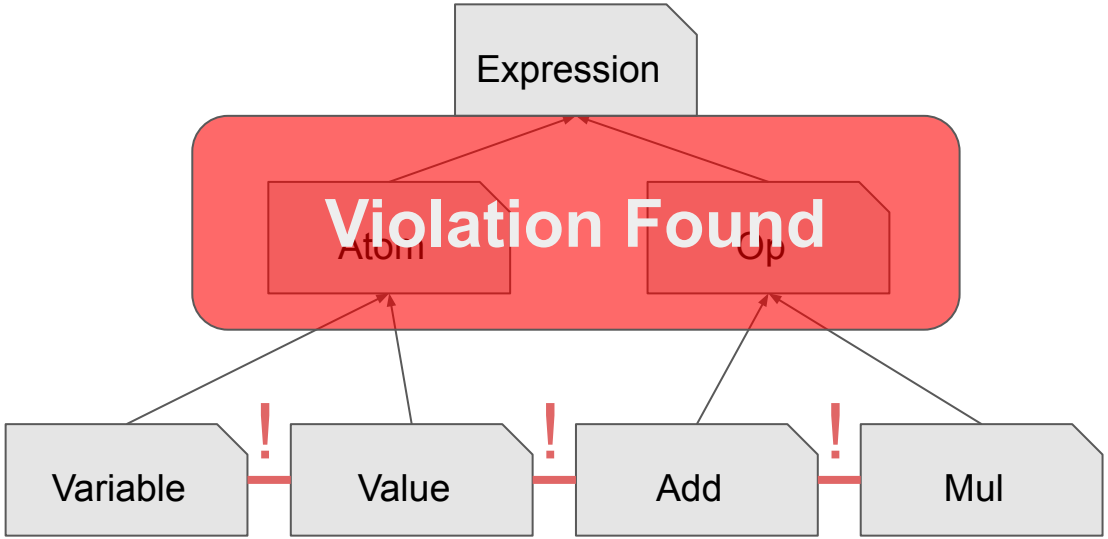
Architecture and Features



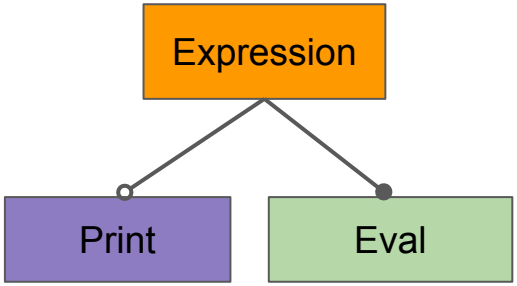


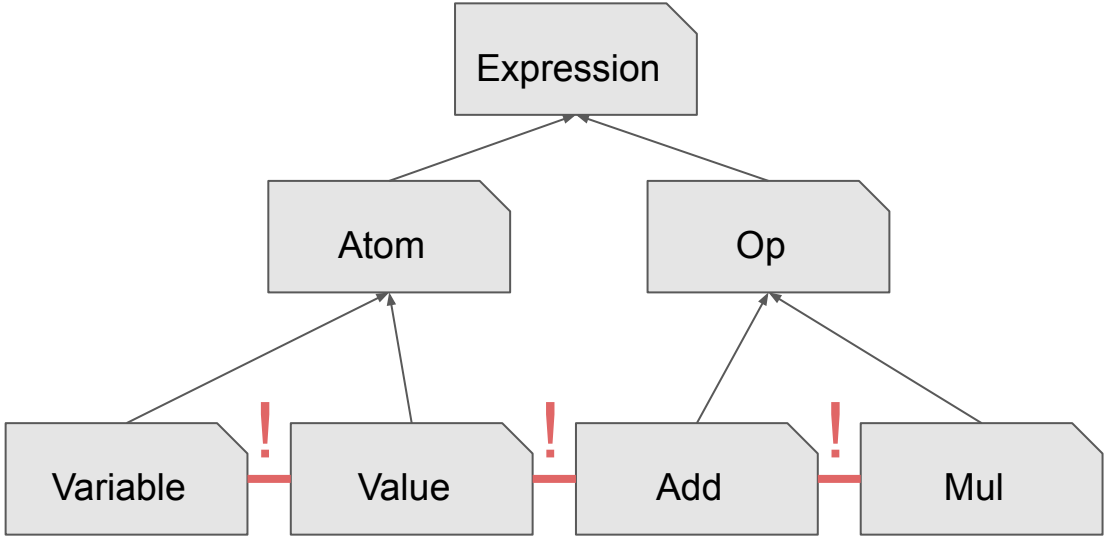
Architecture and Features



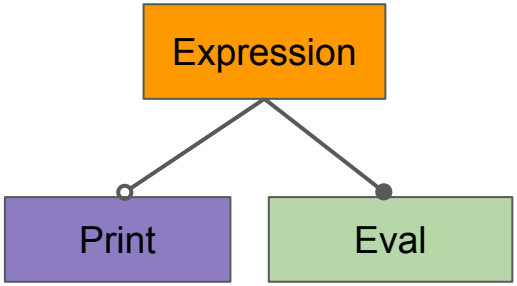


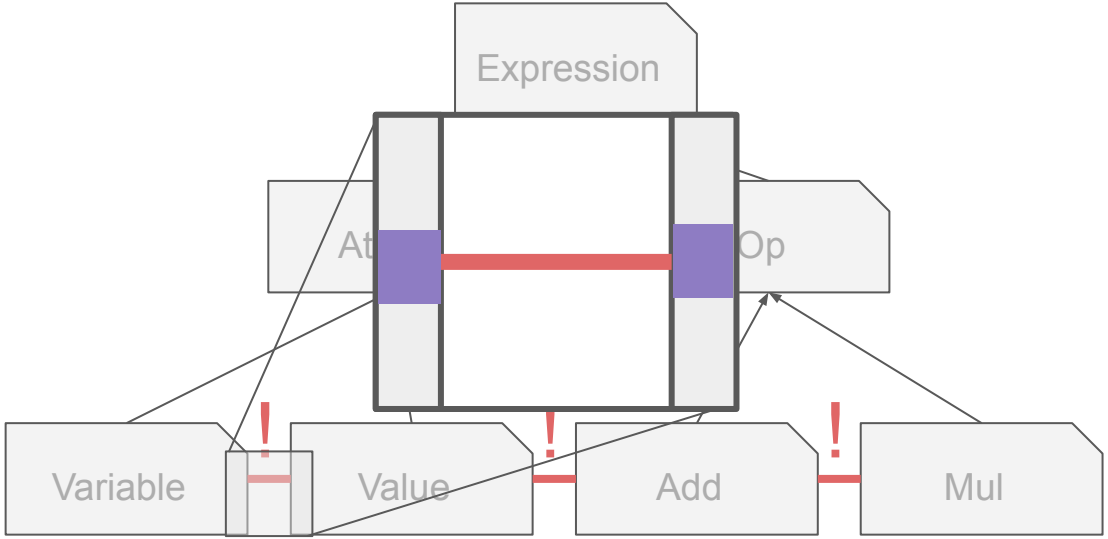
Architecture and Features



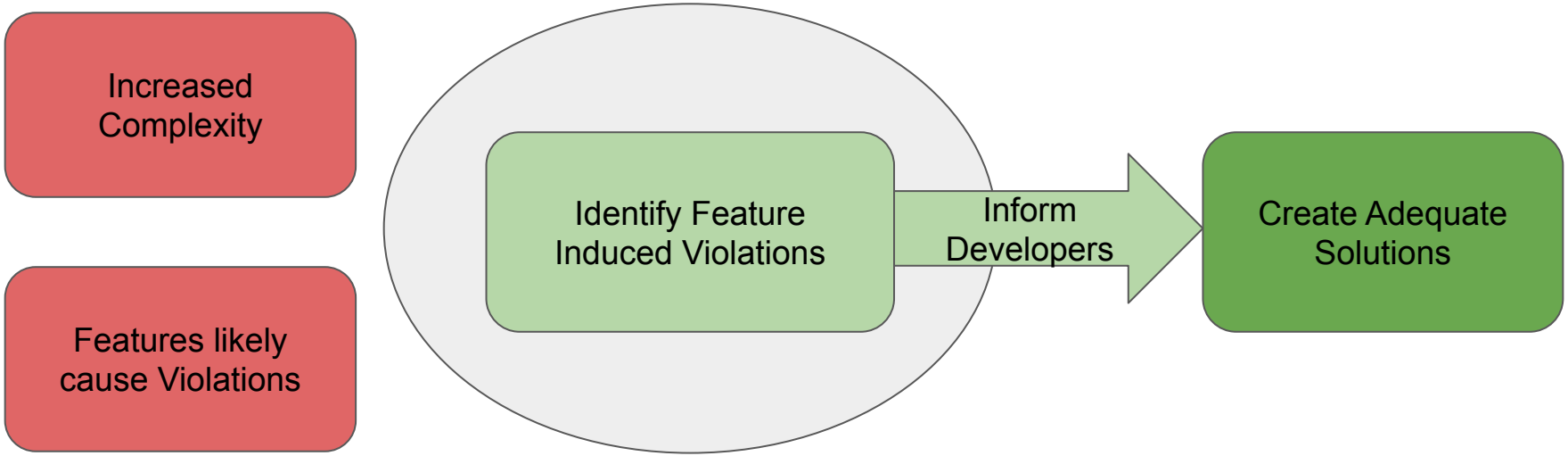


Architecture and Features

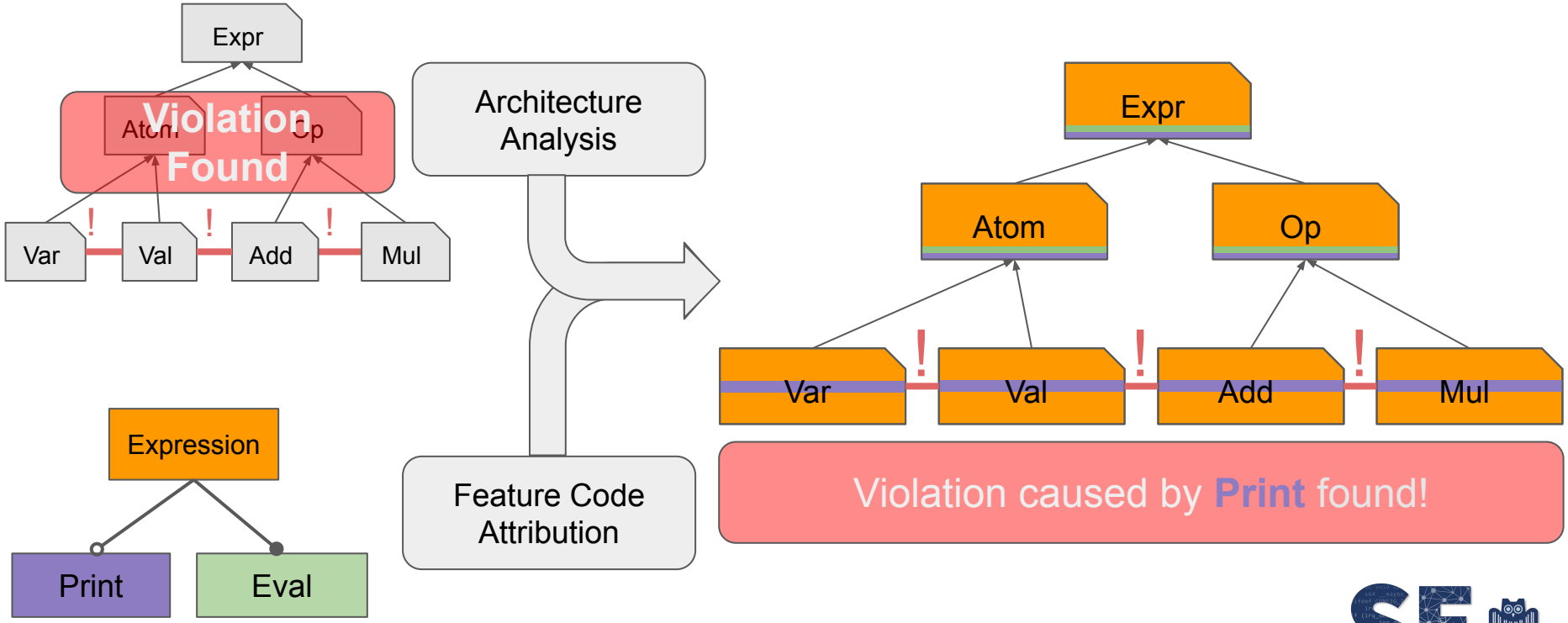




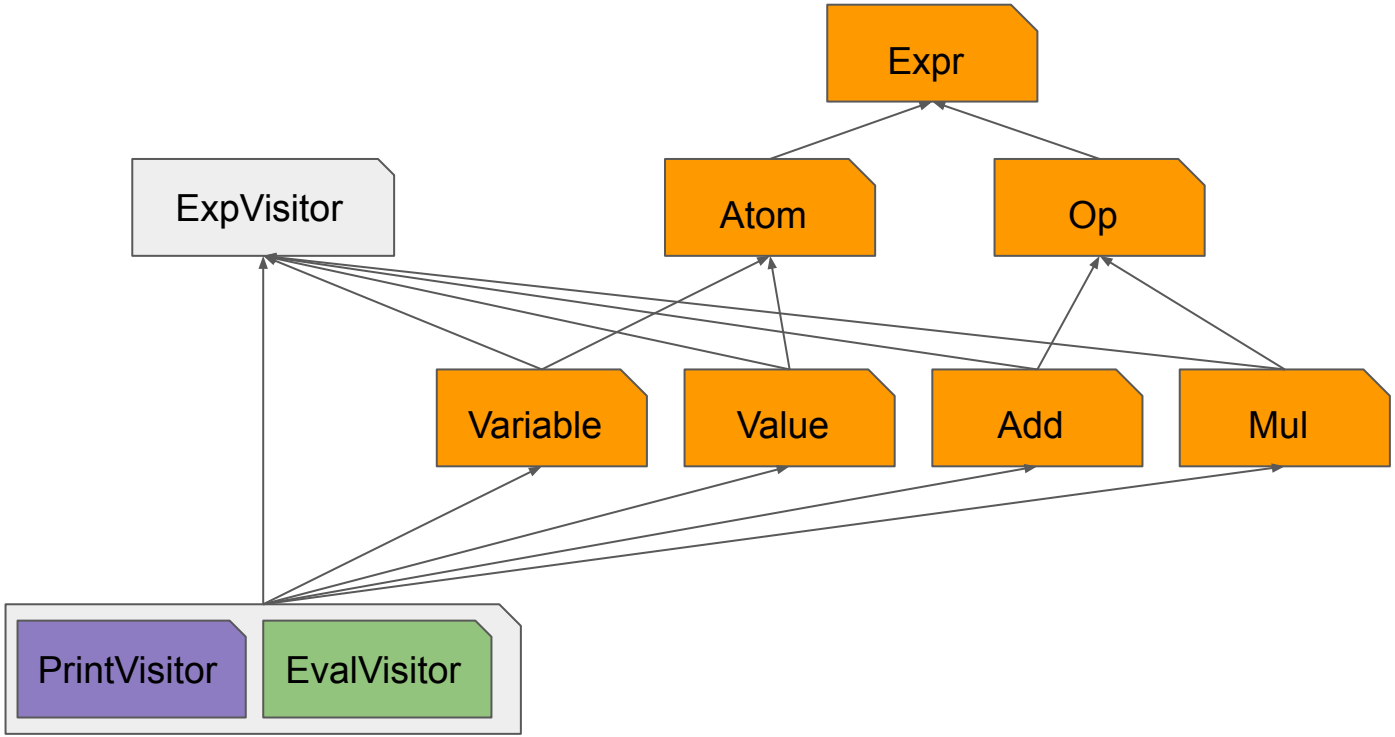
Architecture and Features



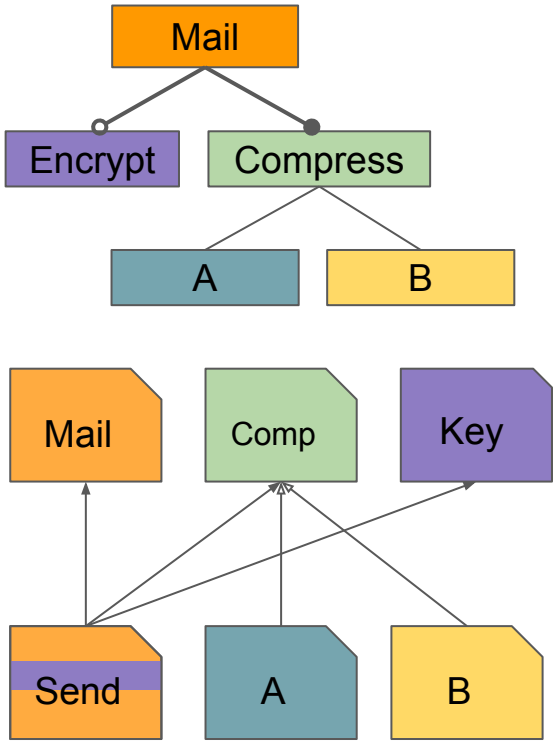
Detection/Identification

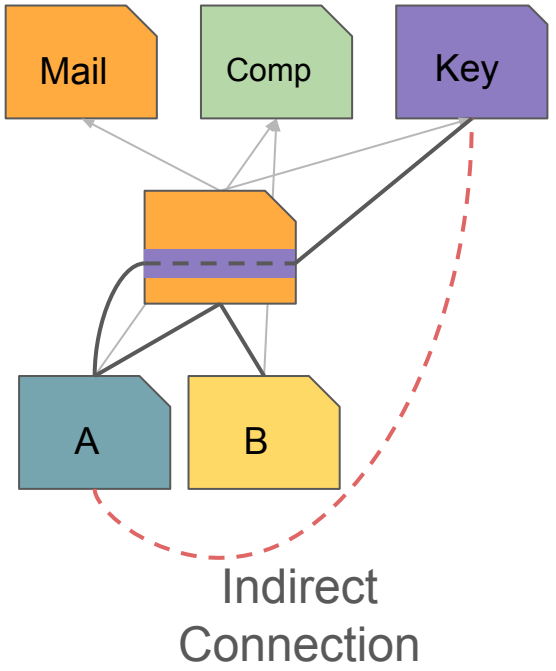


Potential Solution



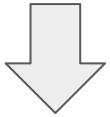
Improved Precision





Conclusion

- We need architectural rules to manage complexity of software
- Developers often circumvent these rules
- Feature implementation may often be a offender due to crosscutting concerns
- State of the art architectural analysis lacks feature information



- Feature location informed architecture analysis
- New architecture smell through dataflow analysis