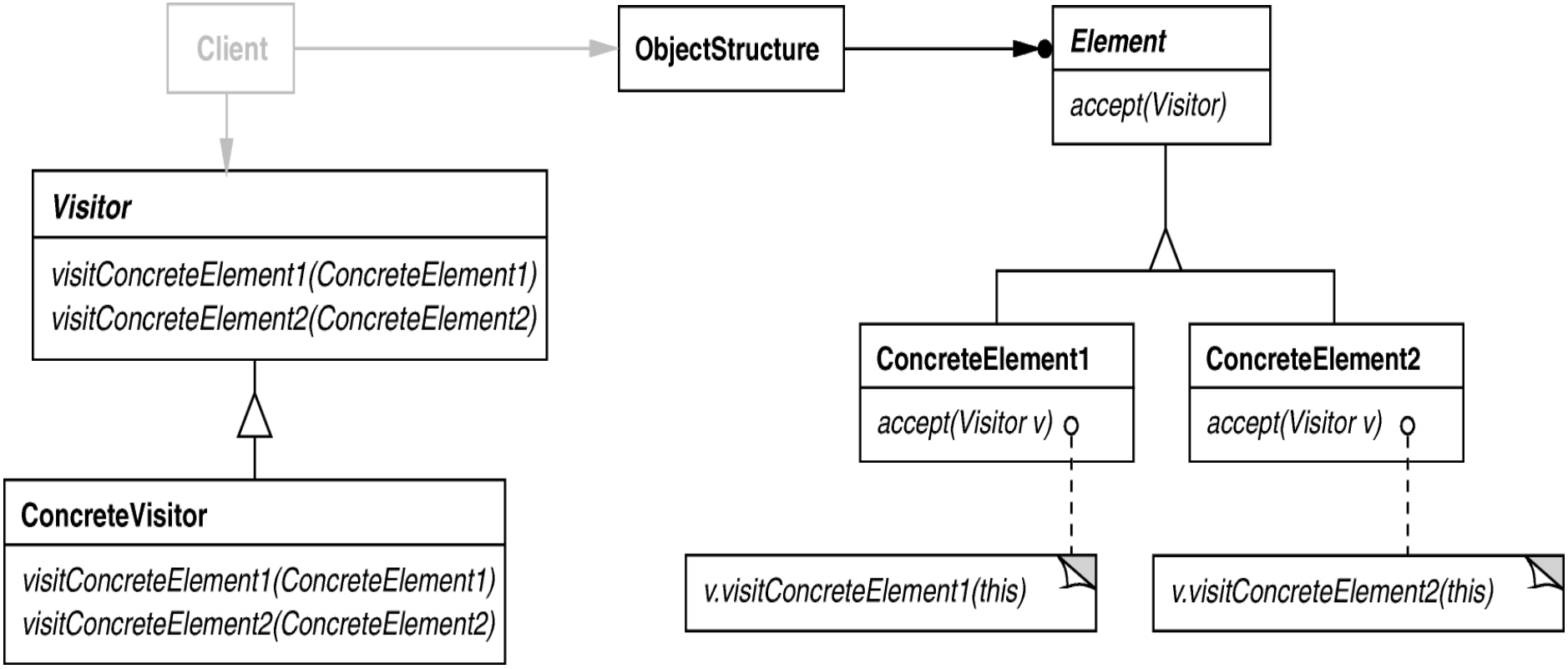
The Visitor Pattern

Structure & Functionality

Douglas C. Schmidt

Learning Objectives in This Lesson

- Recognize how the *Visitor* pattern can be applied to enhance expression tree operation extensibility.
- Understand the *Visitor* pattern.



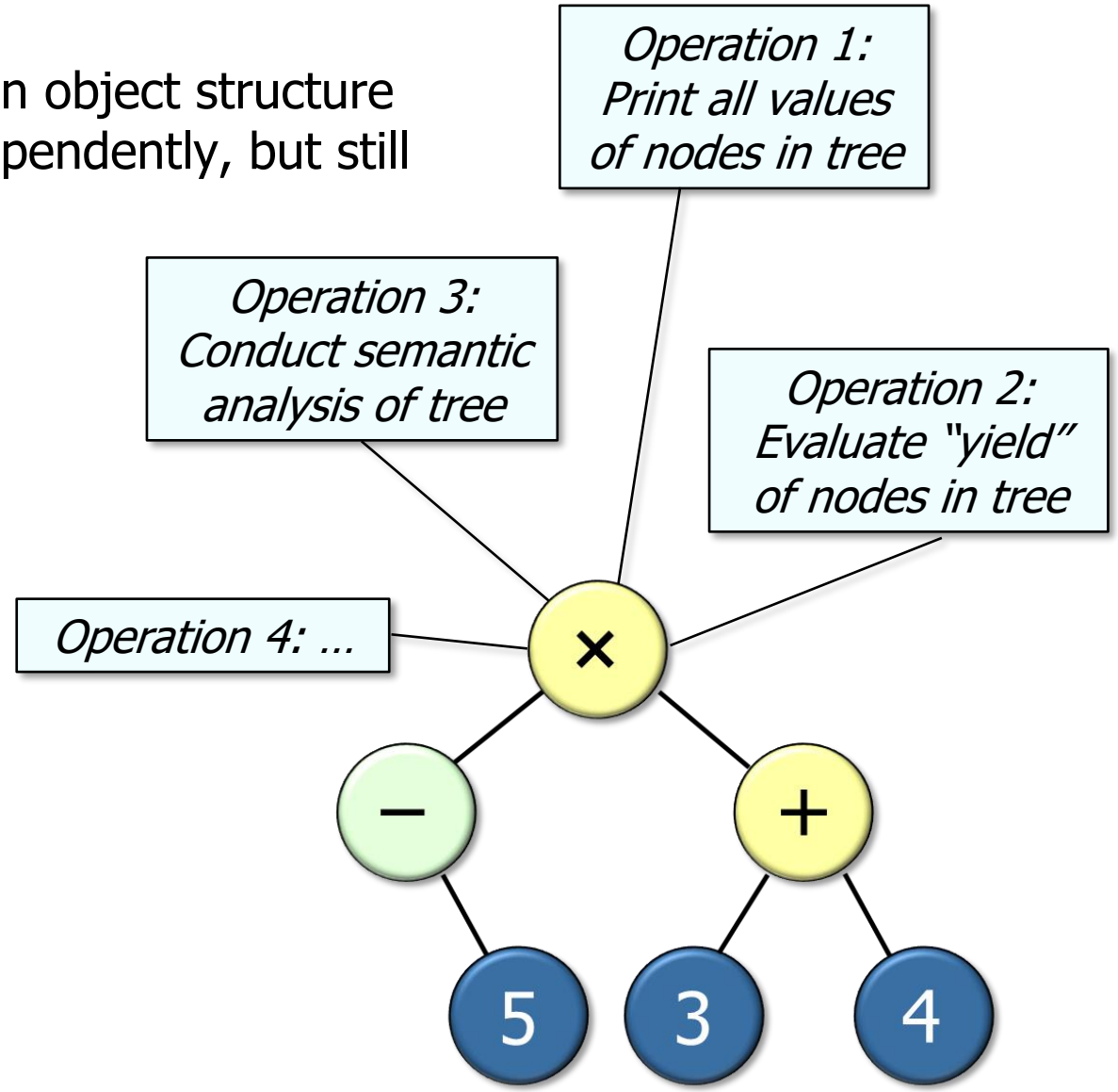
Visitor is one of the most complicated GoF patterns (along with *State*).

Douglas C. Schmidt

Structure & Functionality of the Visitor Pattern

Intent

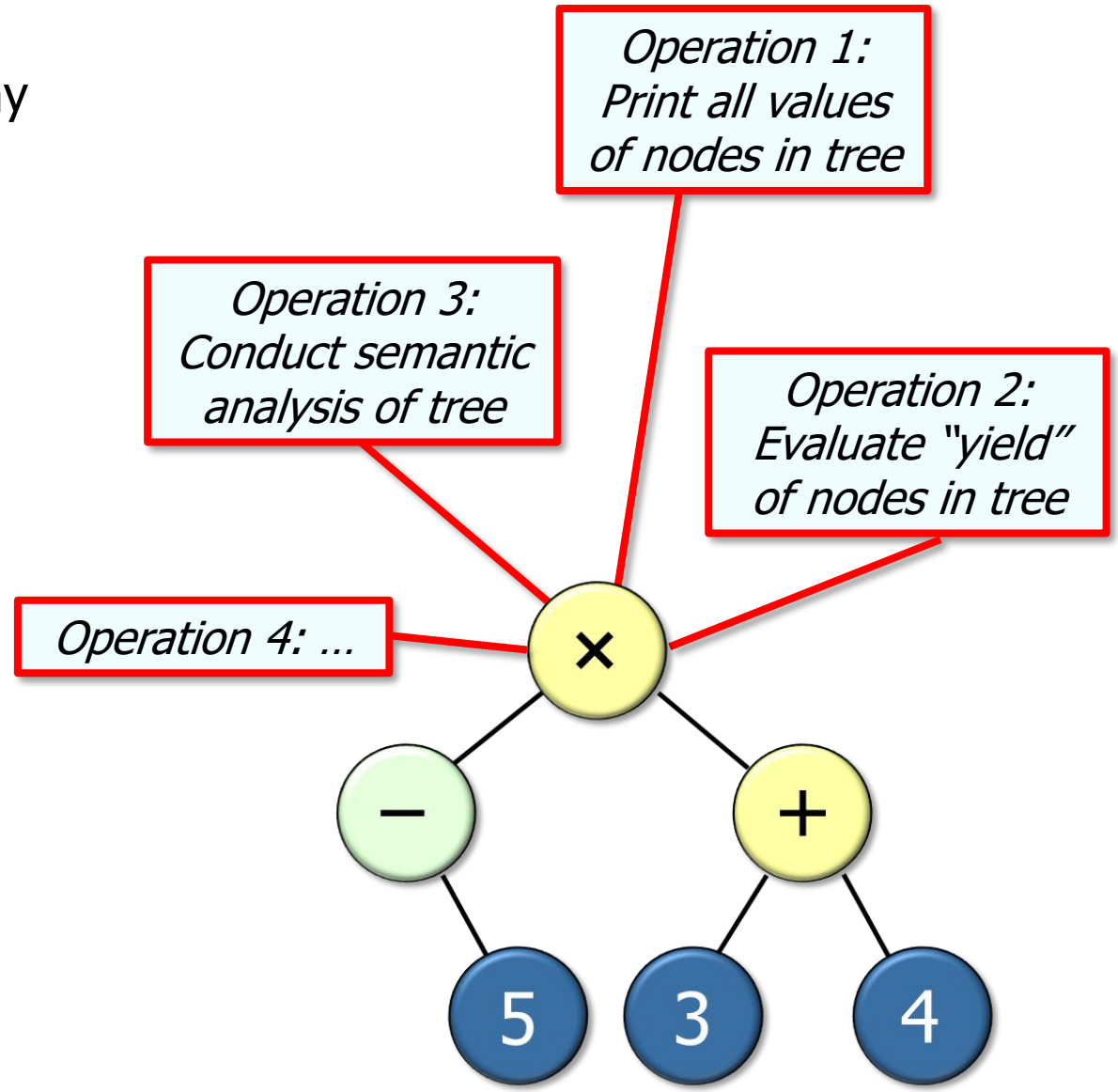
- Centralize operations on an object structure so that they can vary independently, but still behave polymorphically



See en.wikipedia.org/wiki/Visitor_pattern

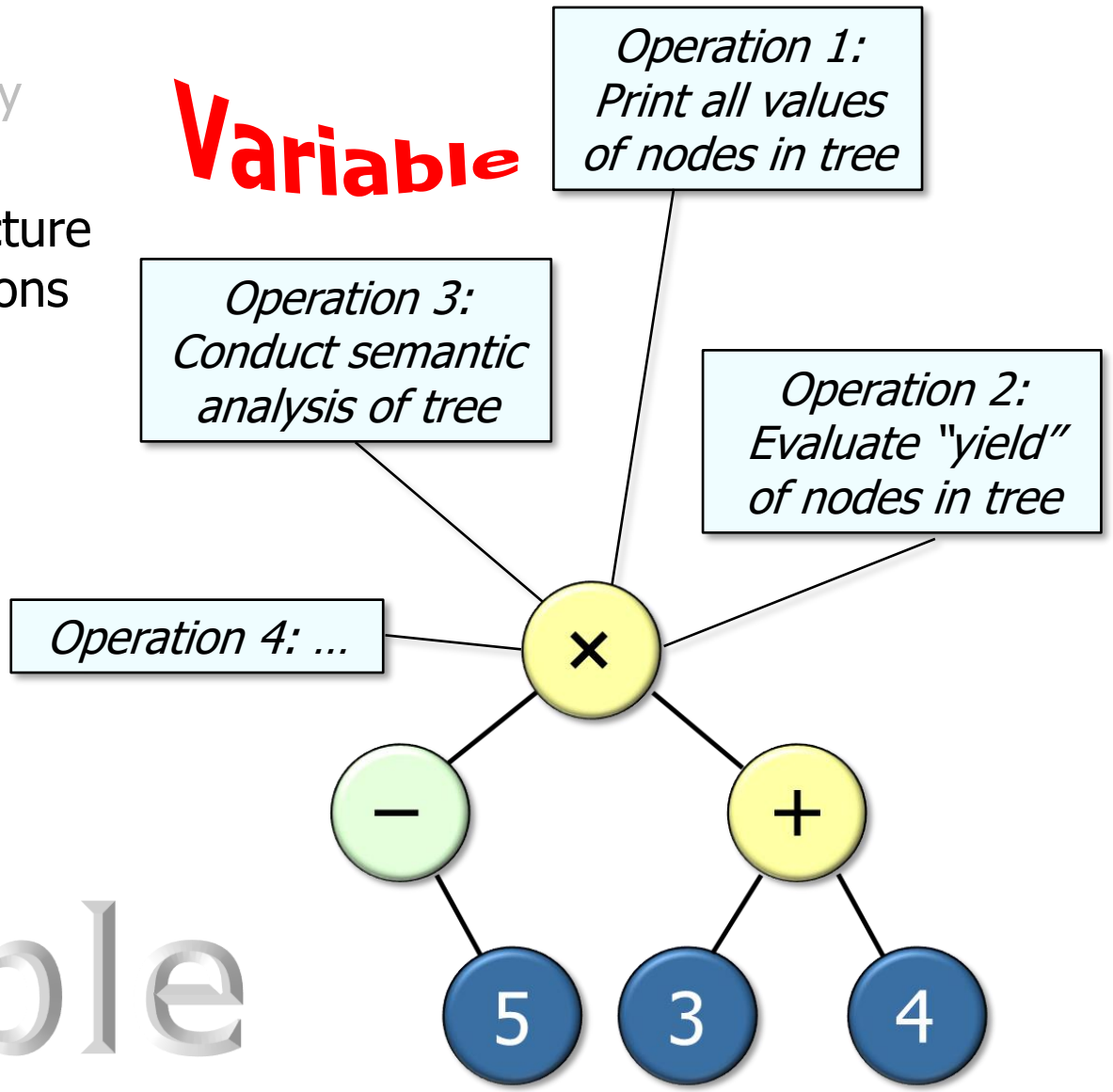
Applicability

- When classes involve many unrelated operations



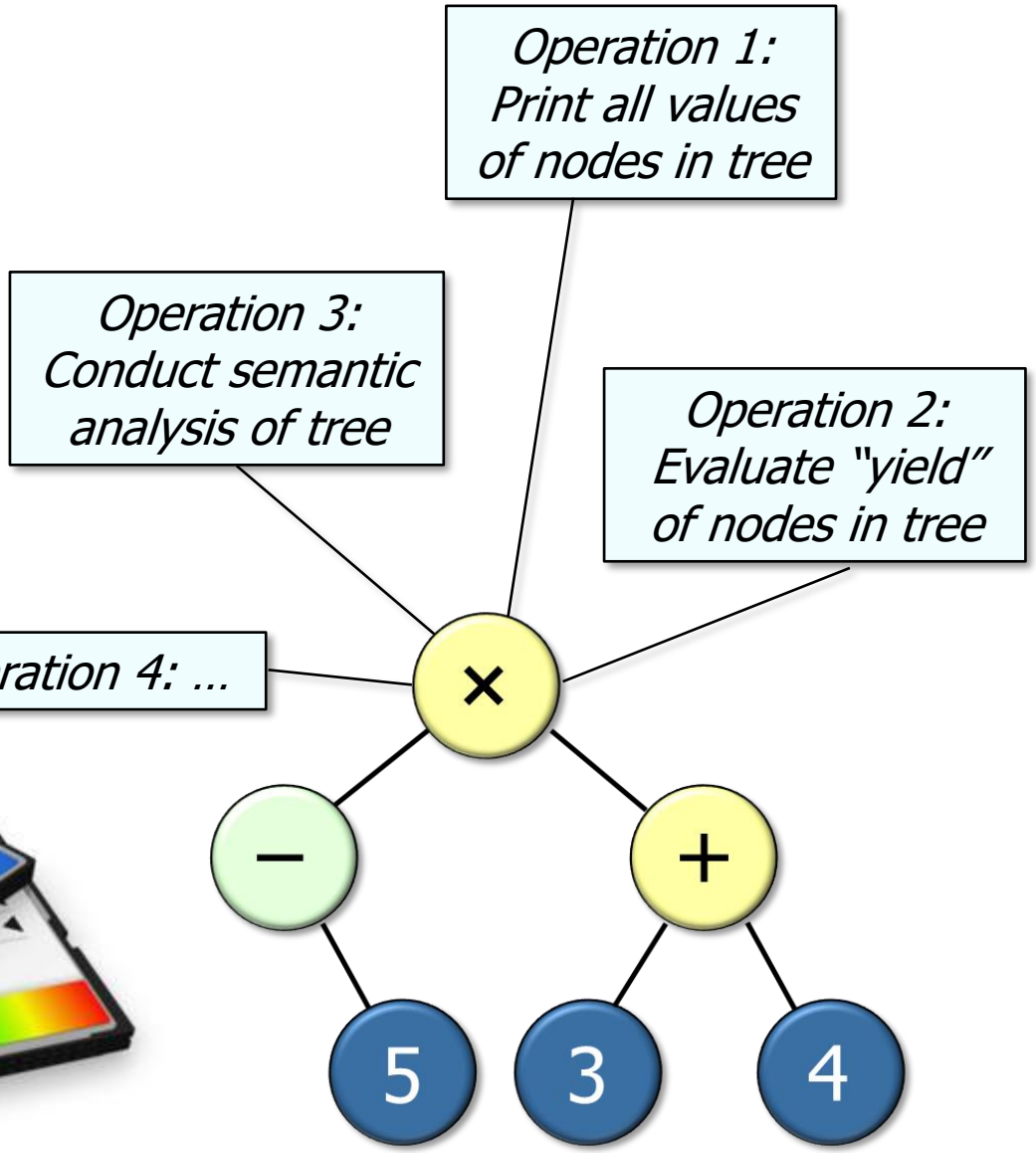
Applicability

- When classes involve many unrelated operations
- Class relationships in structure rarely change, but operations on them *do* change

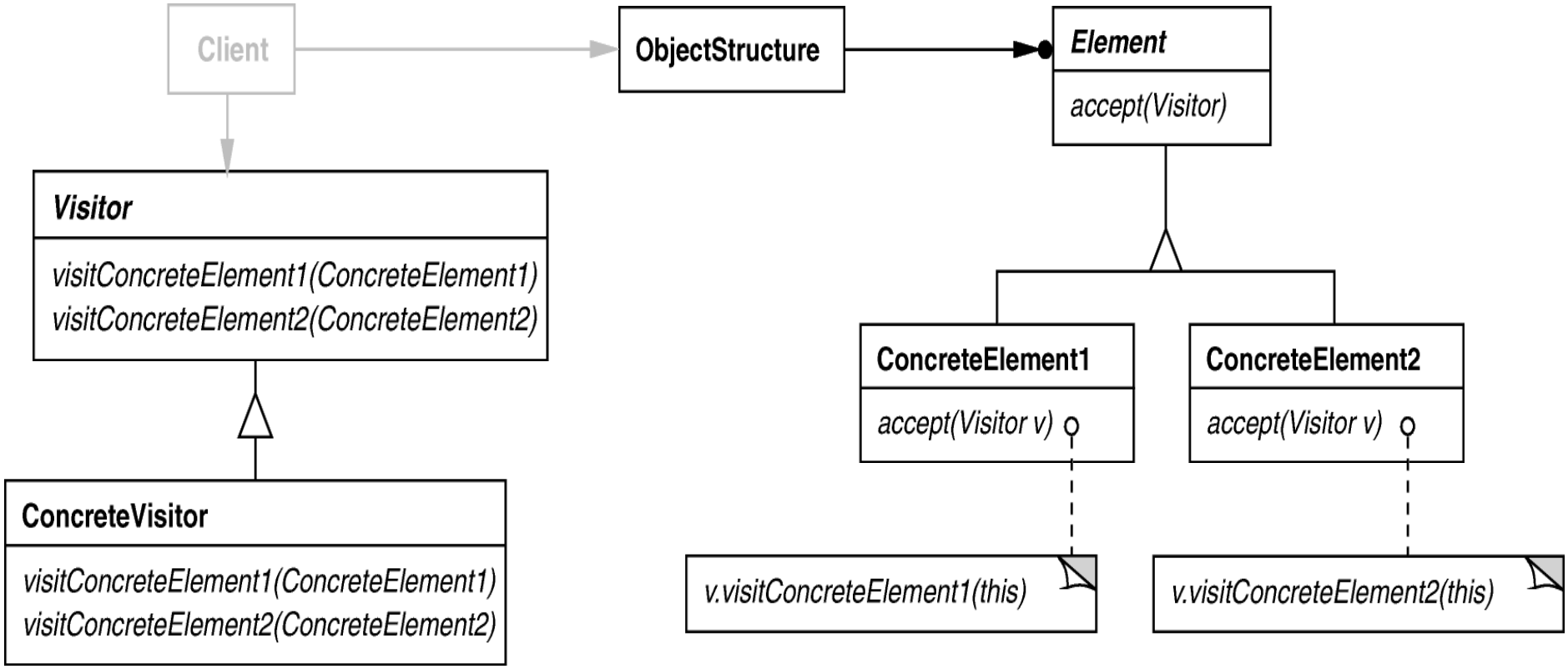


Applicability

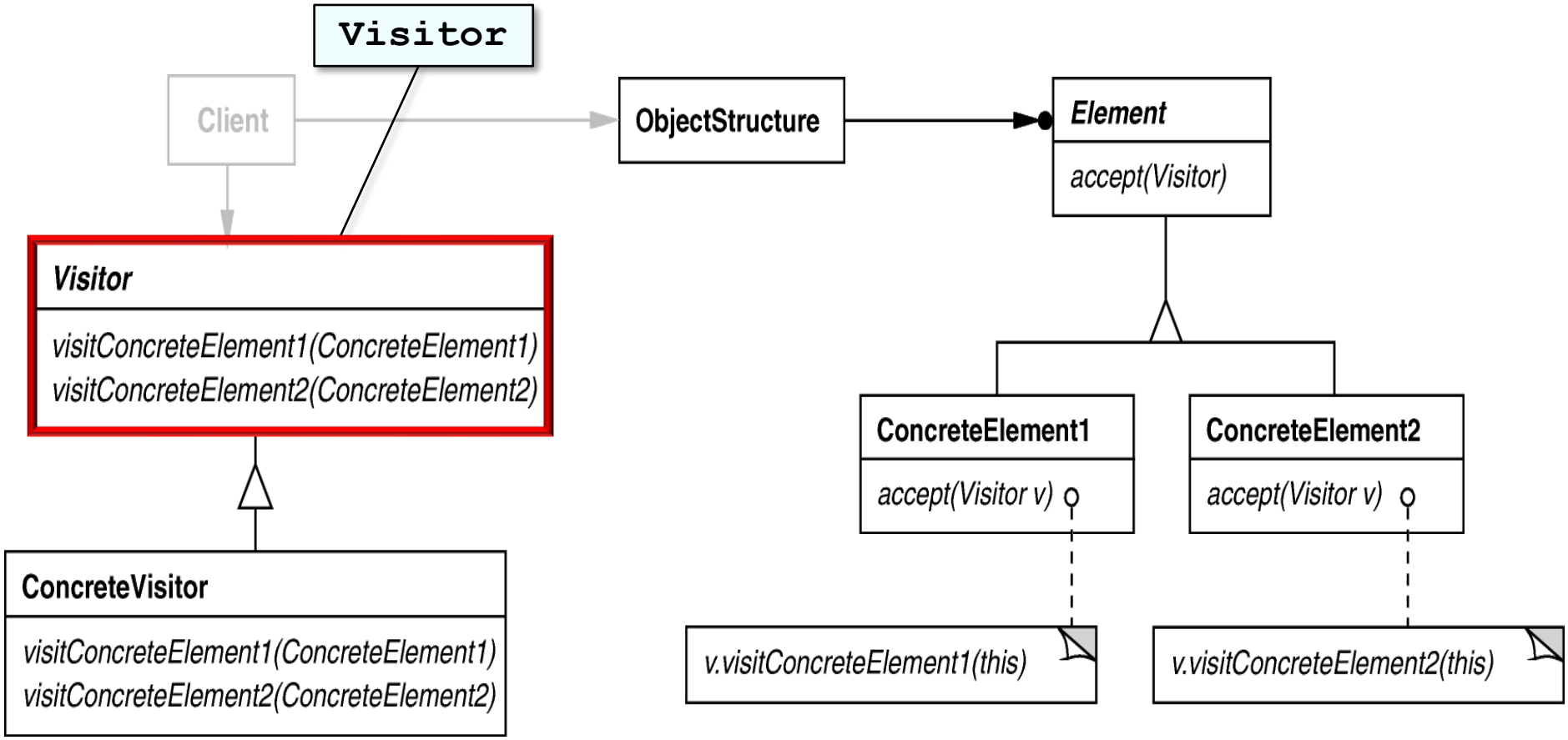
- When classes involve many unrelated operations
- Class relationships in structure rarely change, but operations on them *do* change
- Algorithms keep the state that's updated during traversal



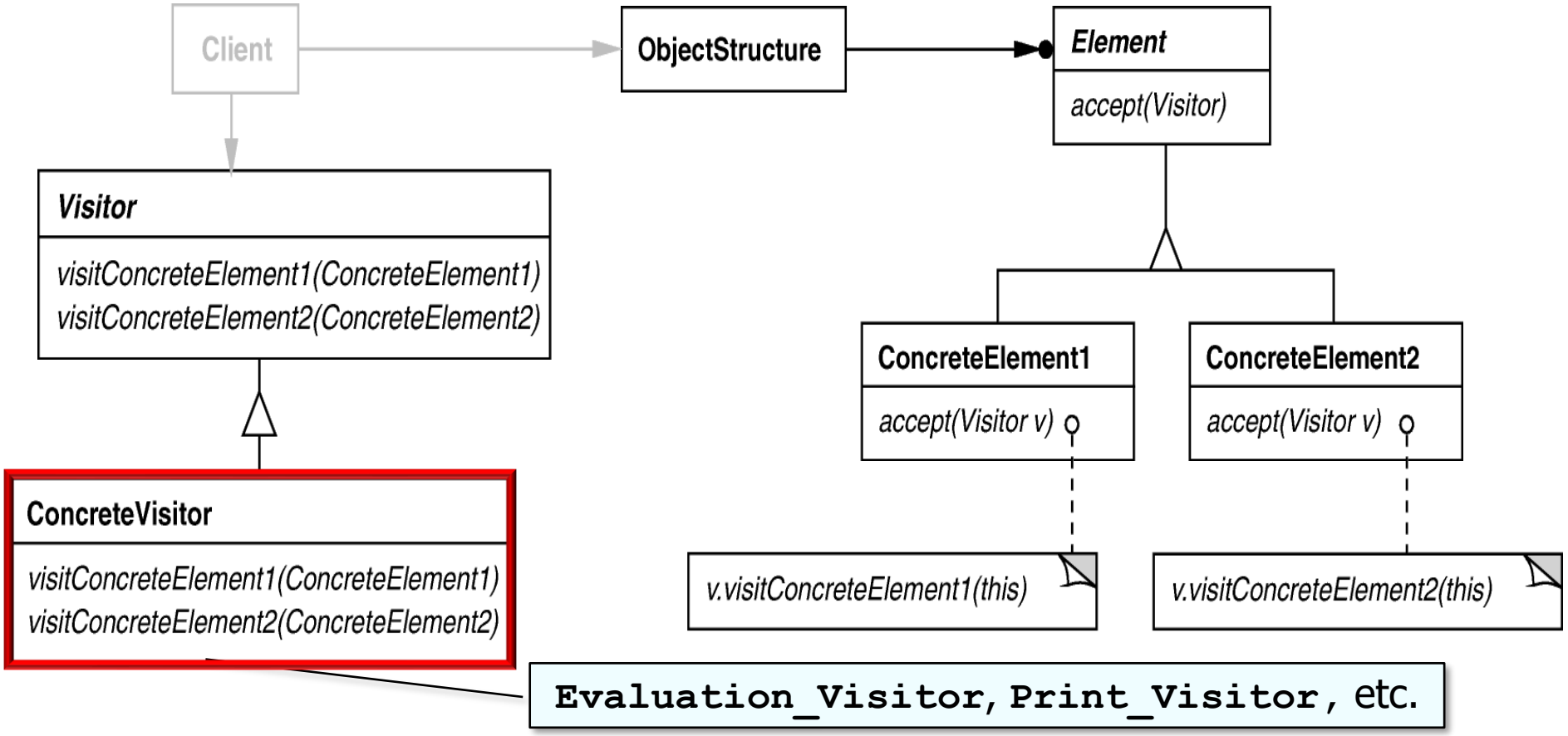
Structure & participants



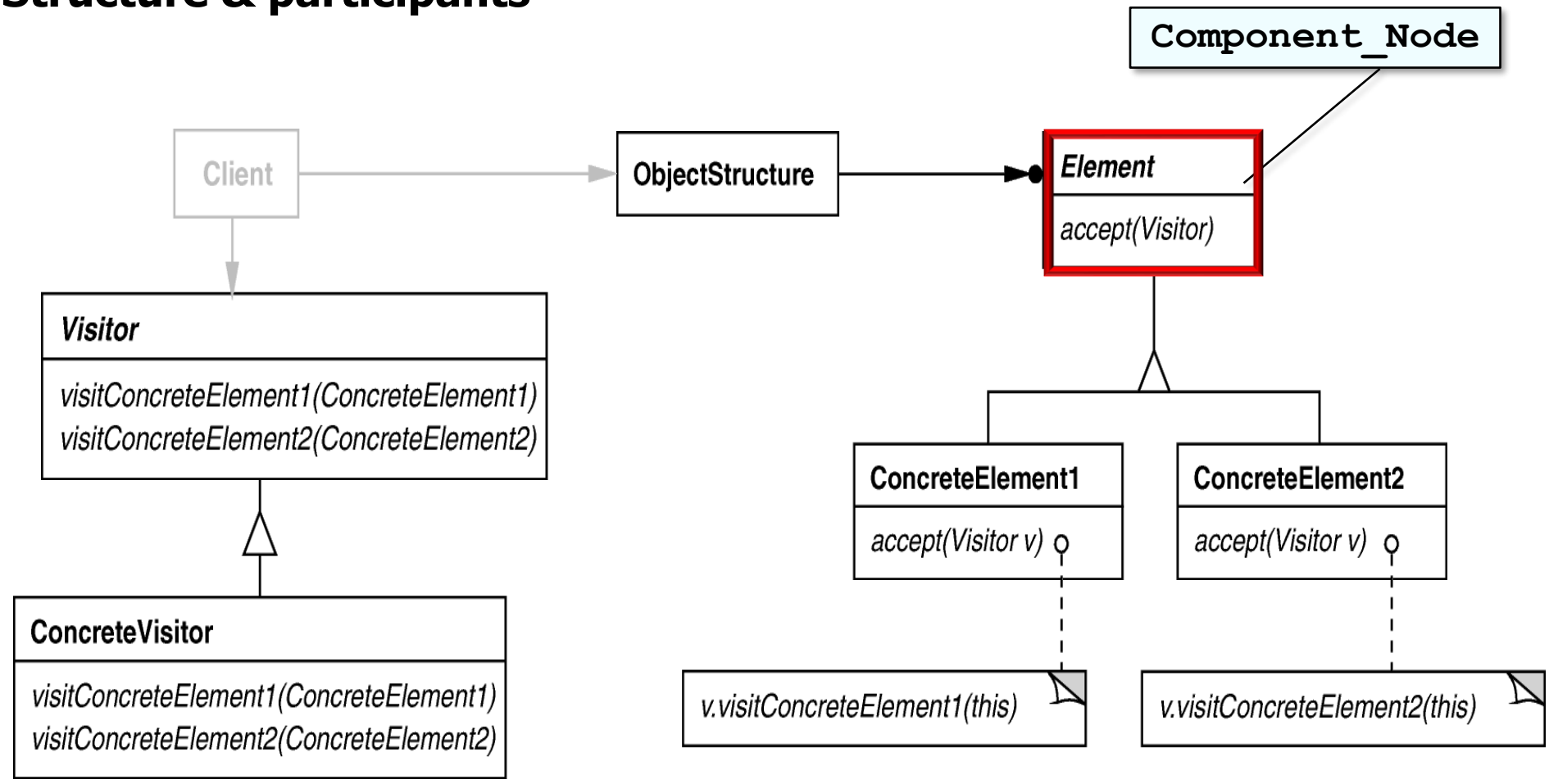
Structure & participants



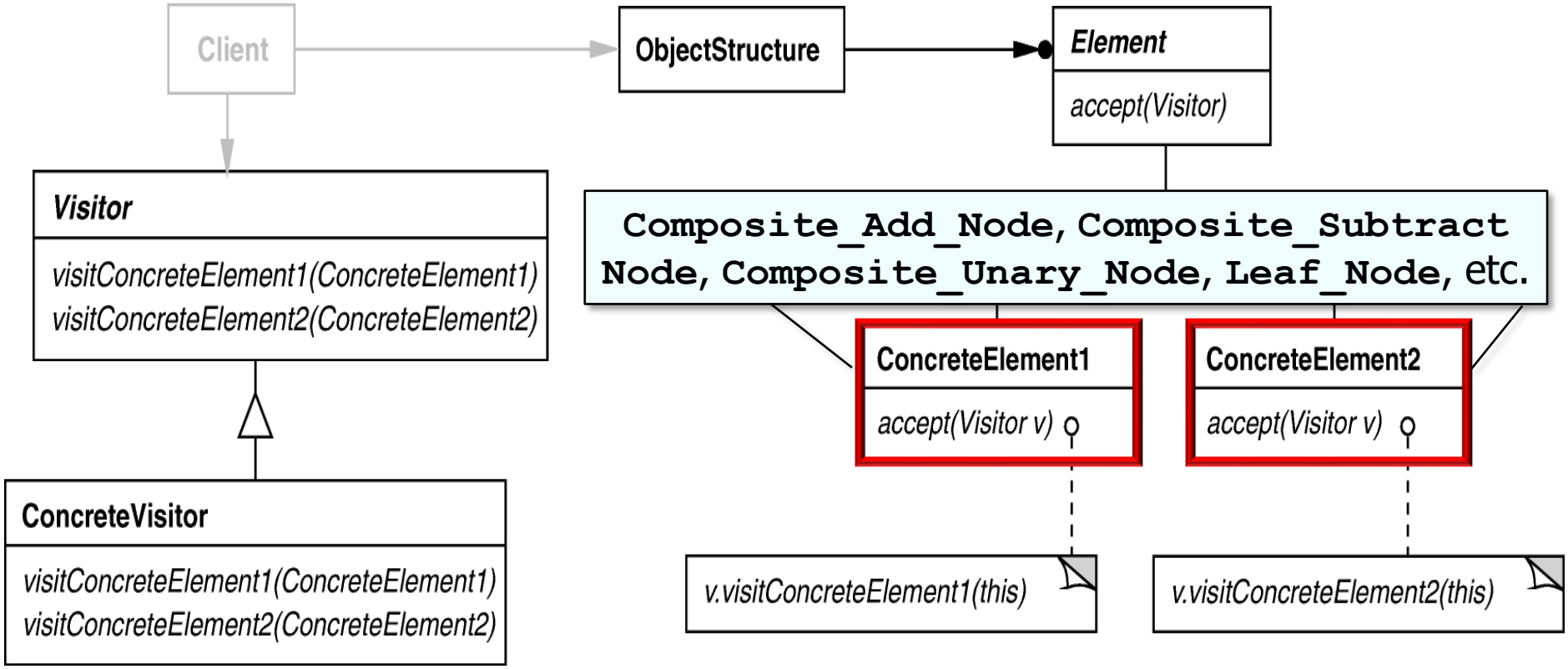
Structure & participants



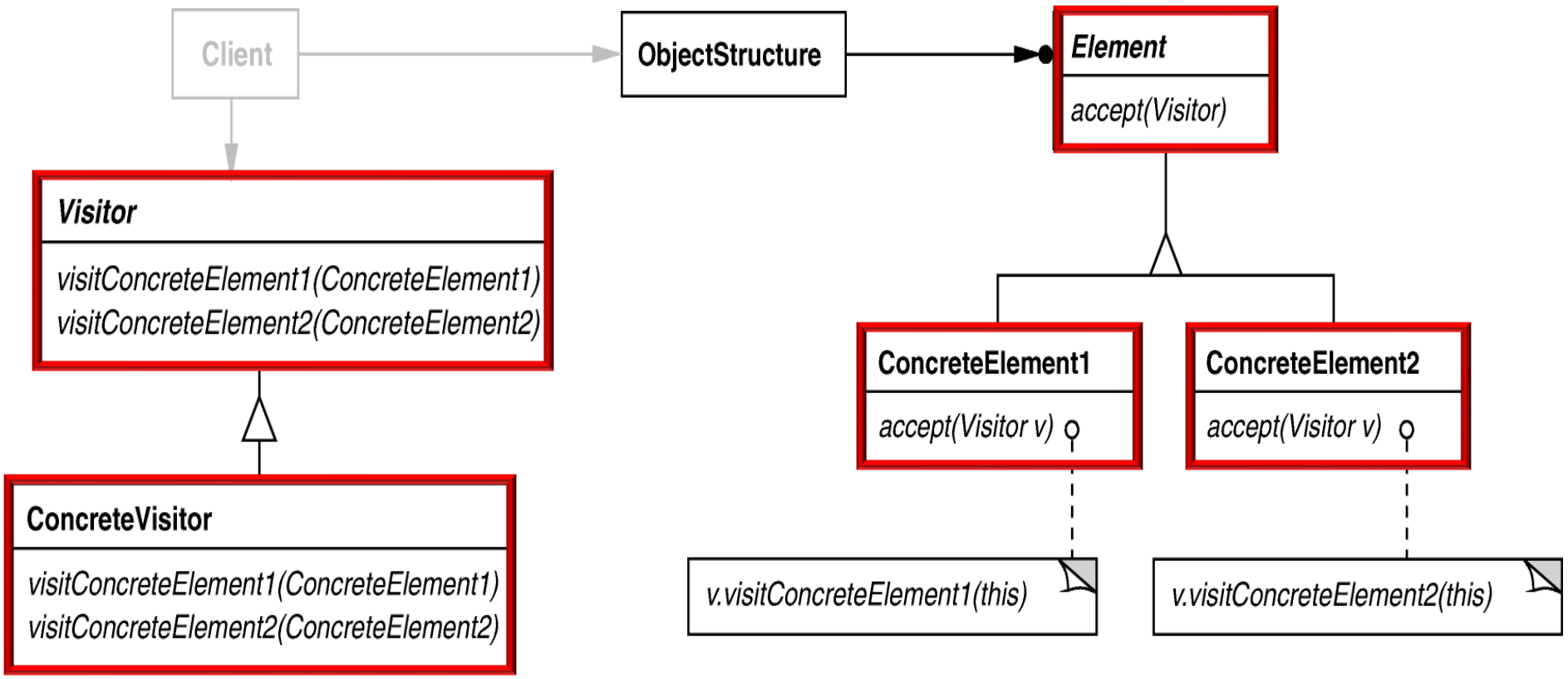
Structure & participants



Structure & participants

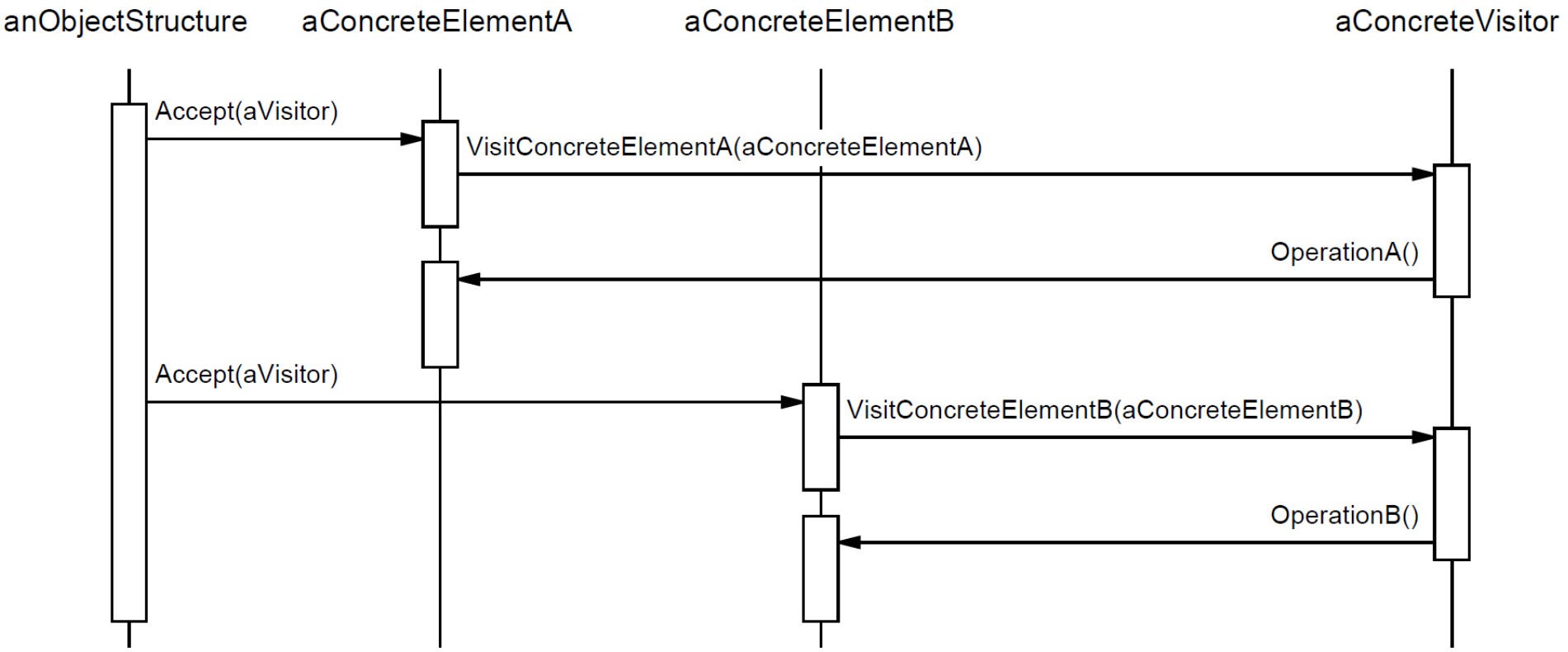


Structure & participants



Visitior's dual inheritance hierarchy + dynamic/static polymorphism is tricky.

Collaborations



This generic object interaction diagram doesn't shed much light on *Visitor*!