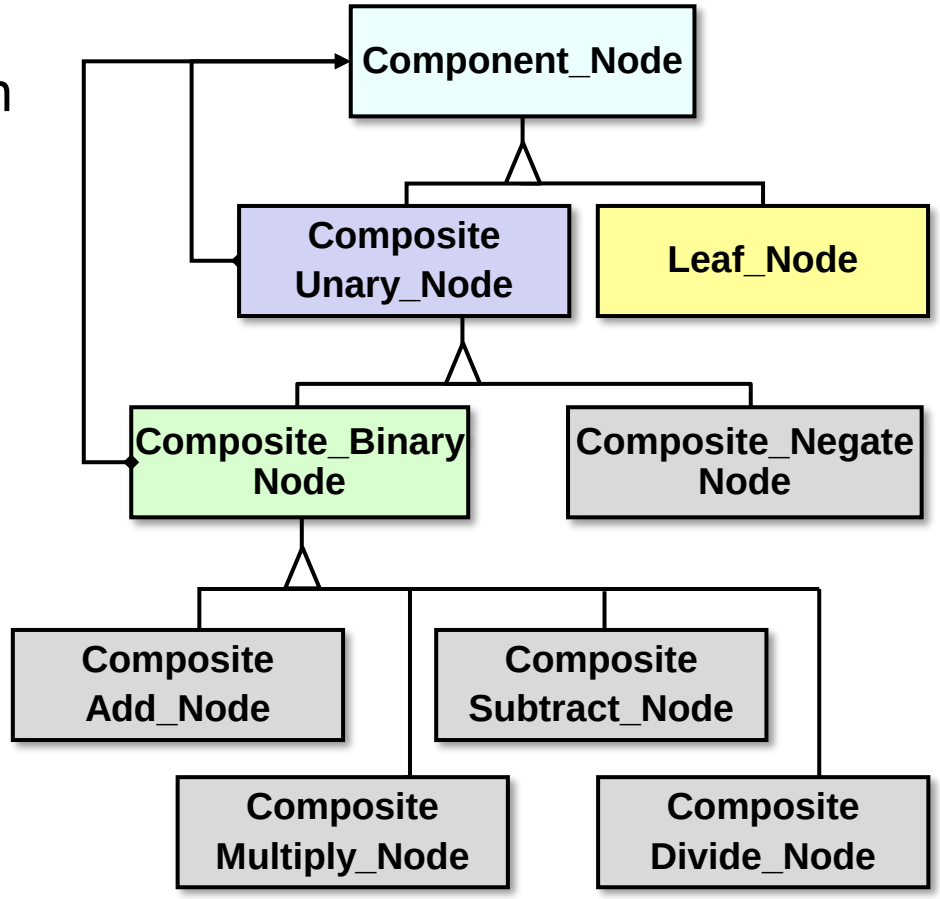
The Composite Pattern

Motivating Example

Douglas C. Schmidt

Learning Objectives in This Lesson

- Recognize how the *Composite* pattern can be applied to make the expression tree object structure more uniform & extensible.

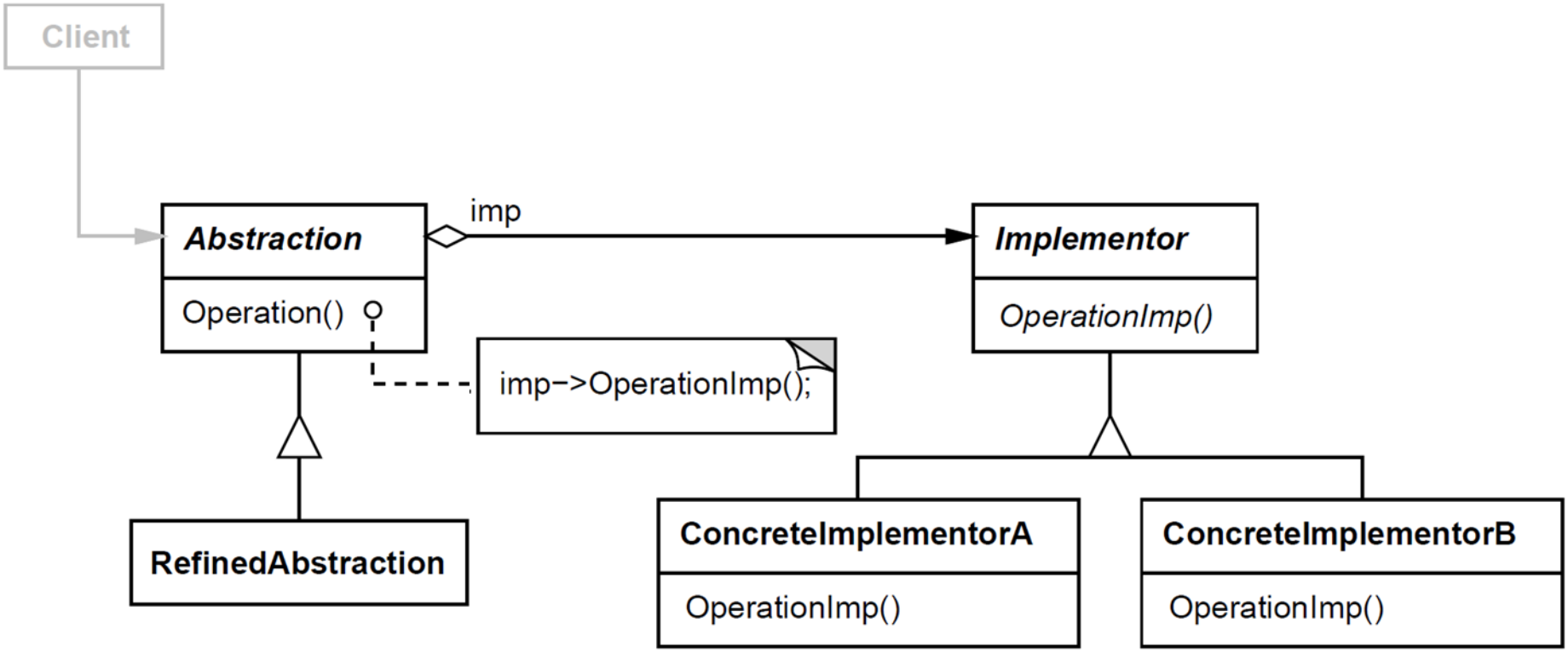


Douglas C. Schmidt

Motivating the Need for the Composite Pattern in the Expression Tree App

Learning Objectives in This Lesson

- Recognize how the *Bridge* pattern can be applied to make the expression tree structure easier to access & evolve transparently.
- Understand the structure & functionality of the *Bridge* pattern.



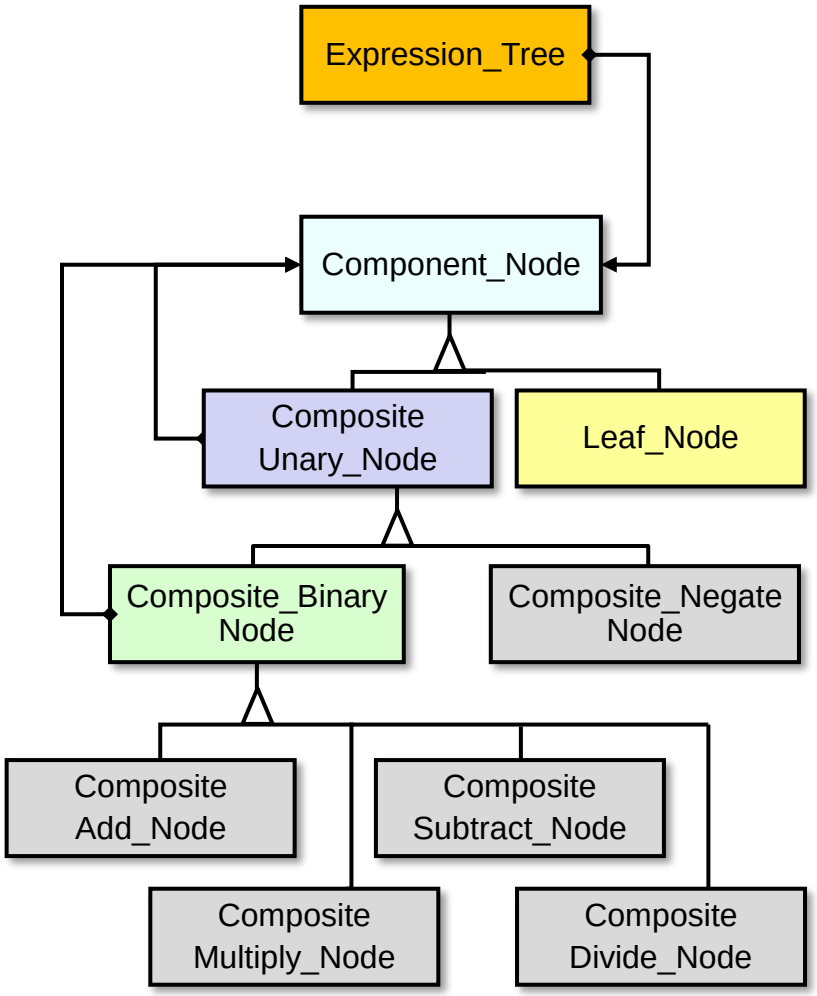
Douglas C. Schmidt

Structure & Functionality of the Bridge Pattern

Bridge GoF Object Structural

Intent

- Separate an abstraction from its implementation(s) so the two can vary independently

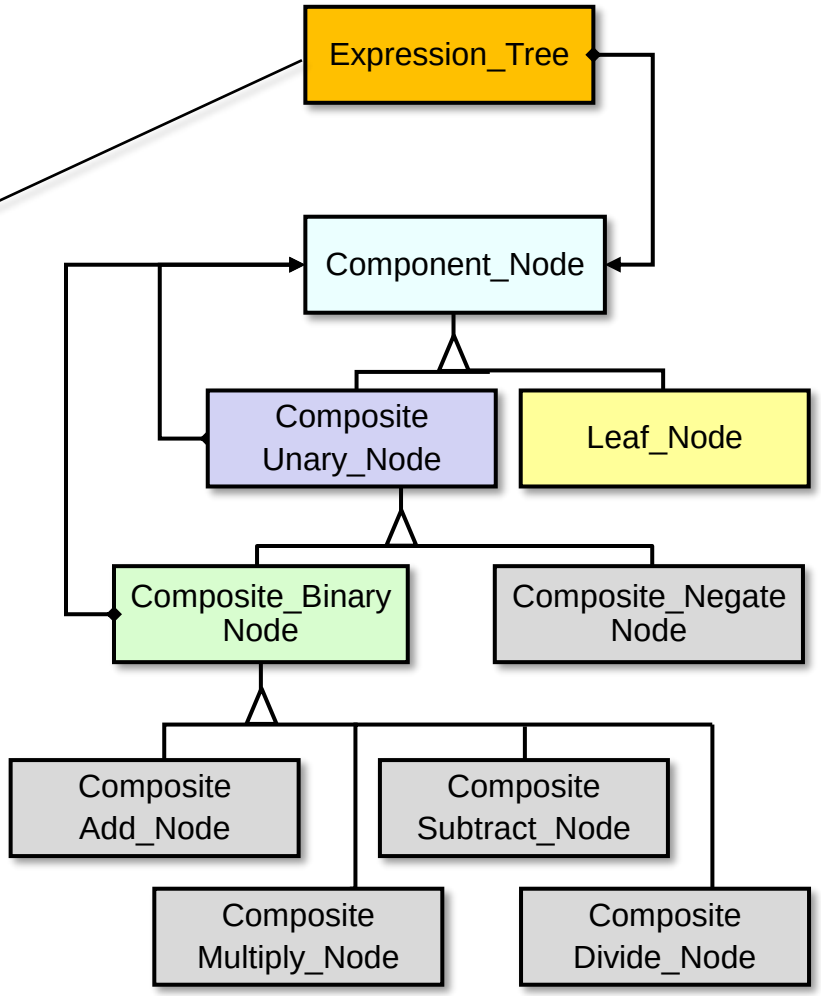


See en.wikipedia.org/wiki/Bridge_pattern

Applicability

- When the abstraction & extensible implementation(s) can vary independently

e.g., the ***`Expression_Tree`*** *service can be refined without affecting clients*

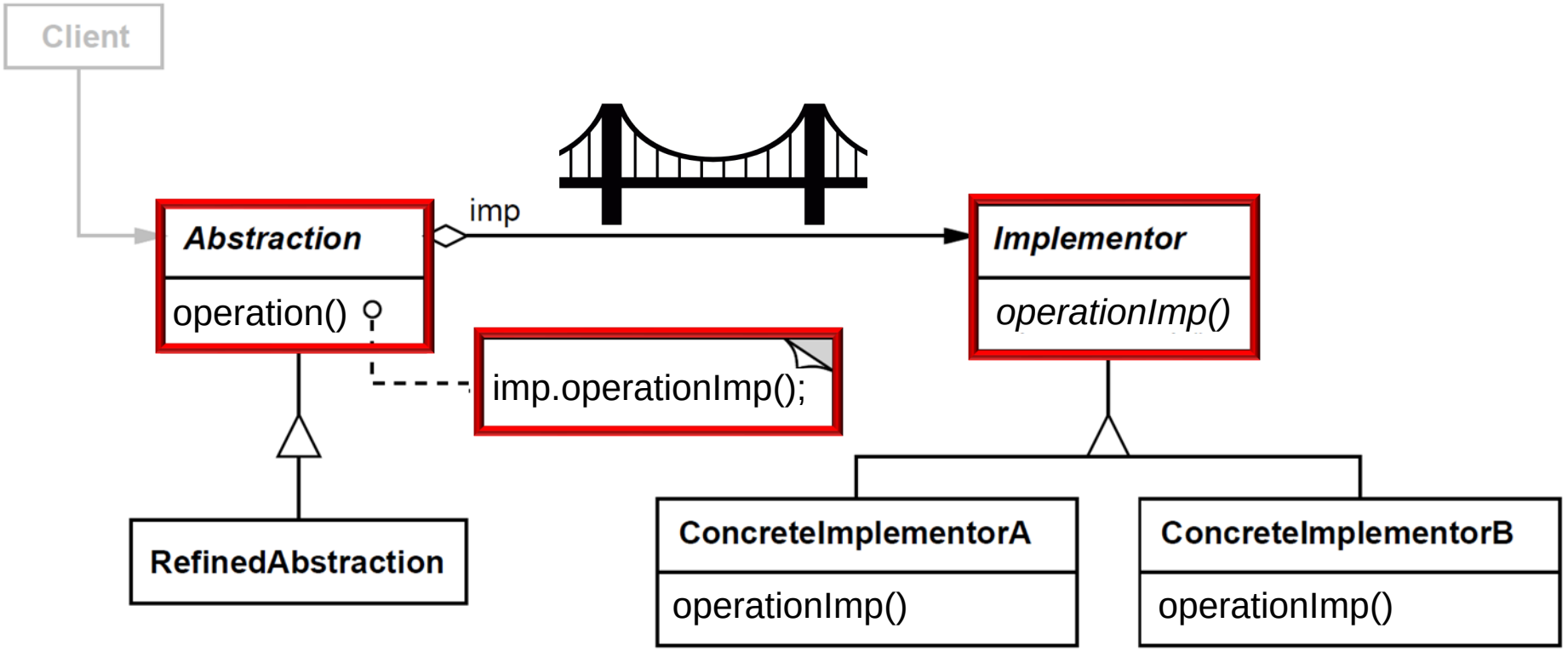


The Bridge Pattern

Structure & Functionality

Douglas C. Schmidt

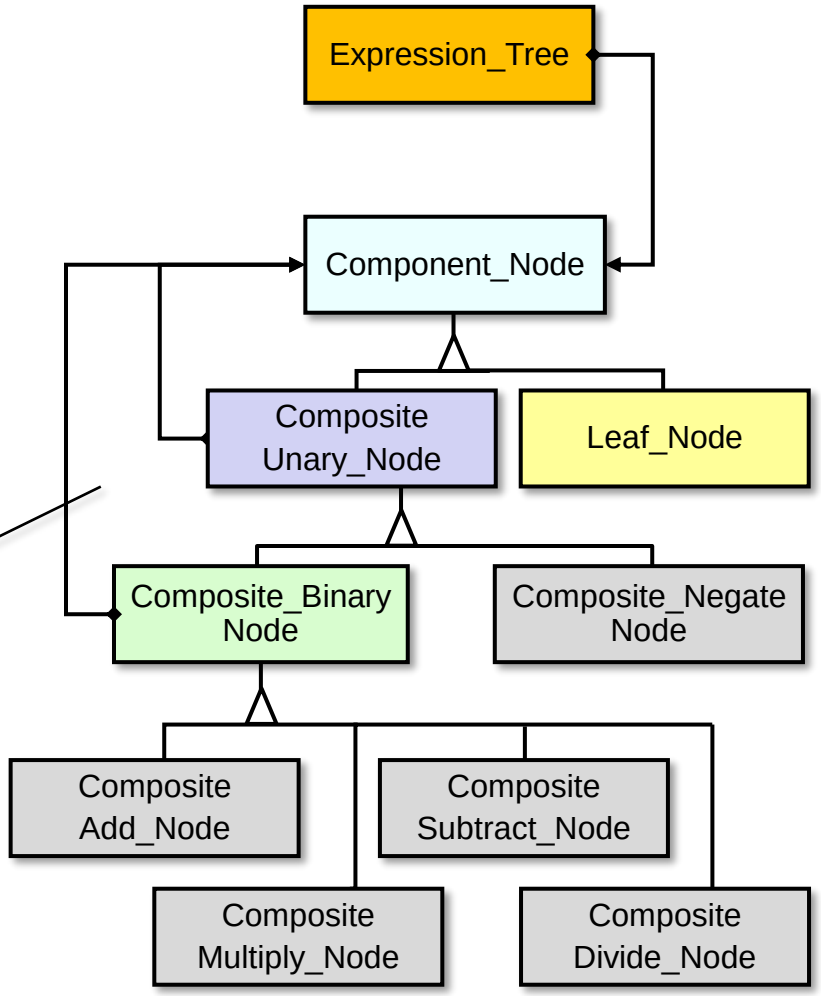
Structure & participants



Applicability

- When the abstraction & extensible implementation(s) can vary independently
- When there's a need to change implementor hierarchies at design-time or runtime without breaking client code

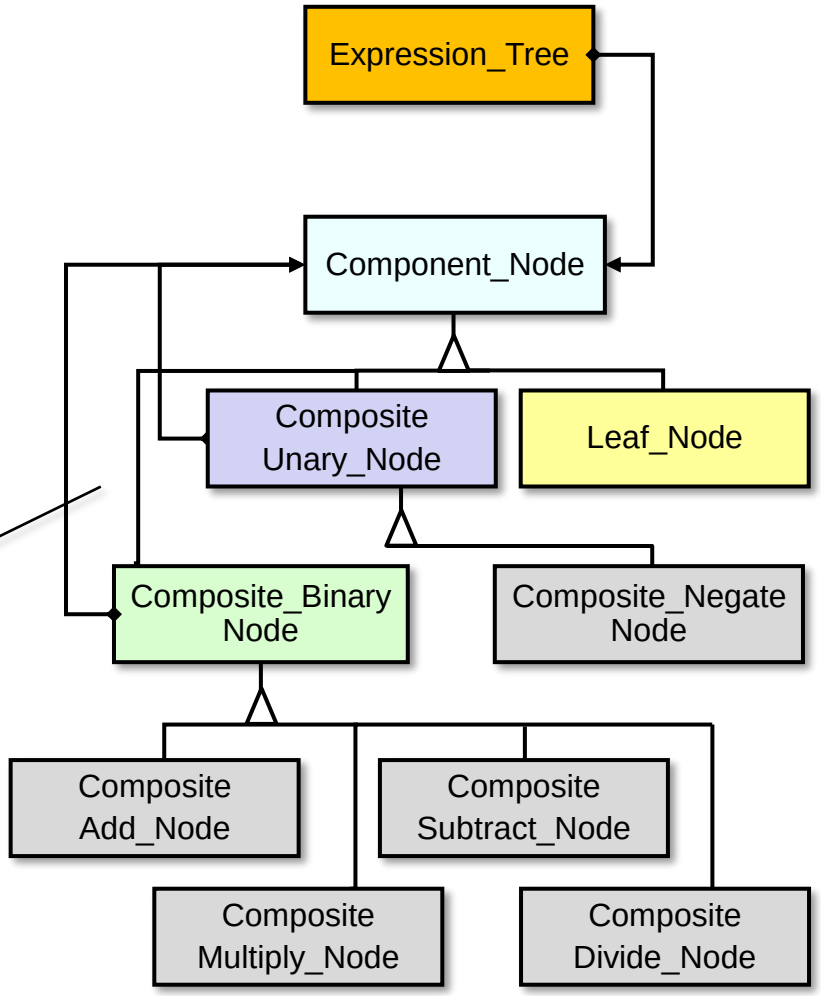
e.g., the ***`Component_Node`*** *implementor hierarchy can change without affecting clients.*



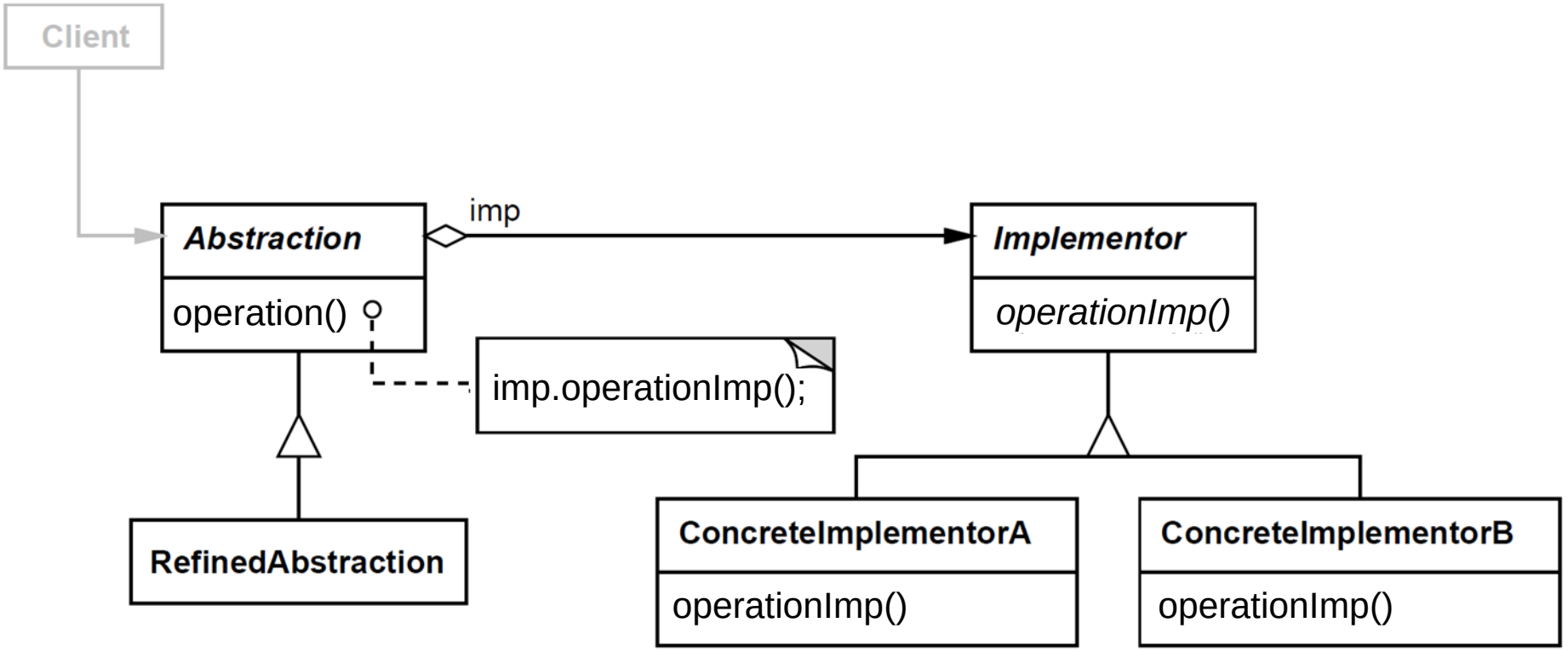
Applicability

- When the abstraction & extensible implementation(s) can vary independently
- When there's a need to change implementor hierarchies at design-time or runtime without breaking client code

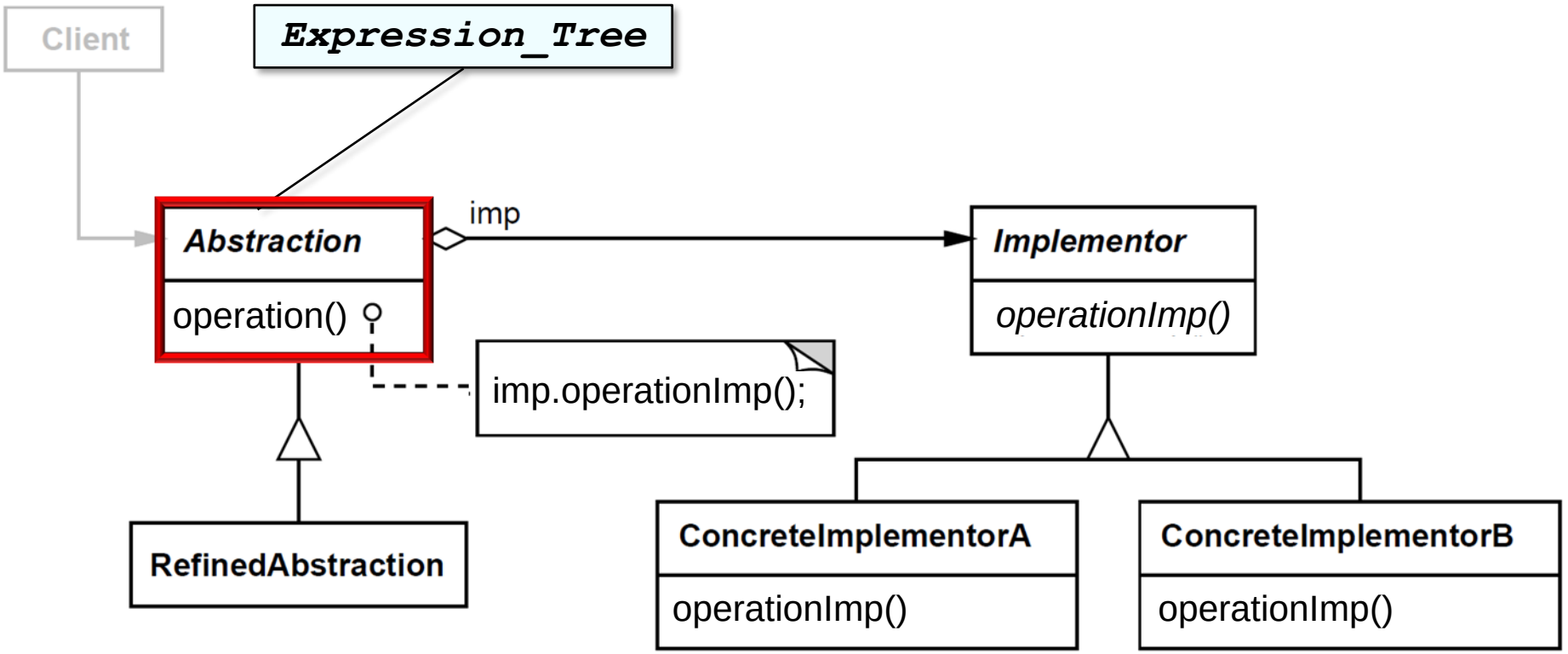
e.g., the ***`Component_Node`*** *implementor hierarchy can change without affecting clients.*



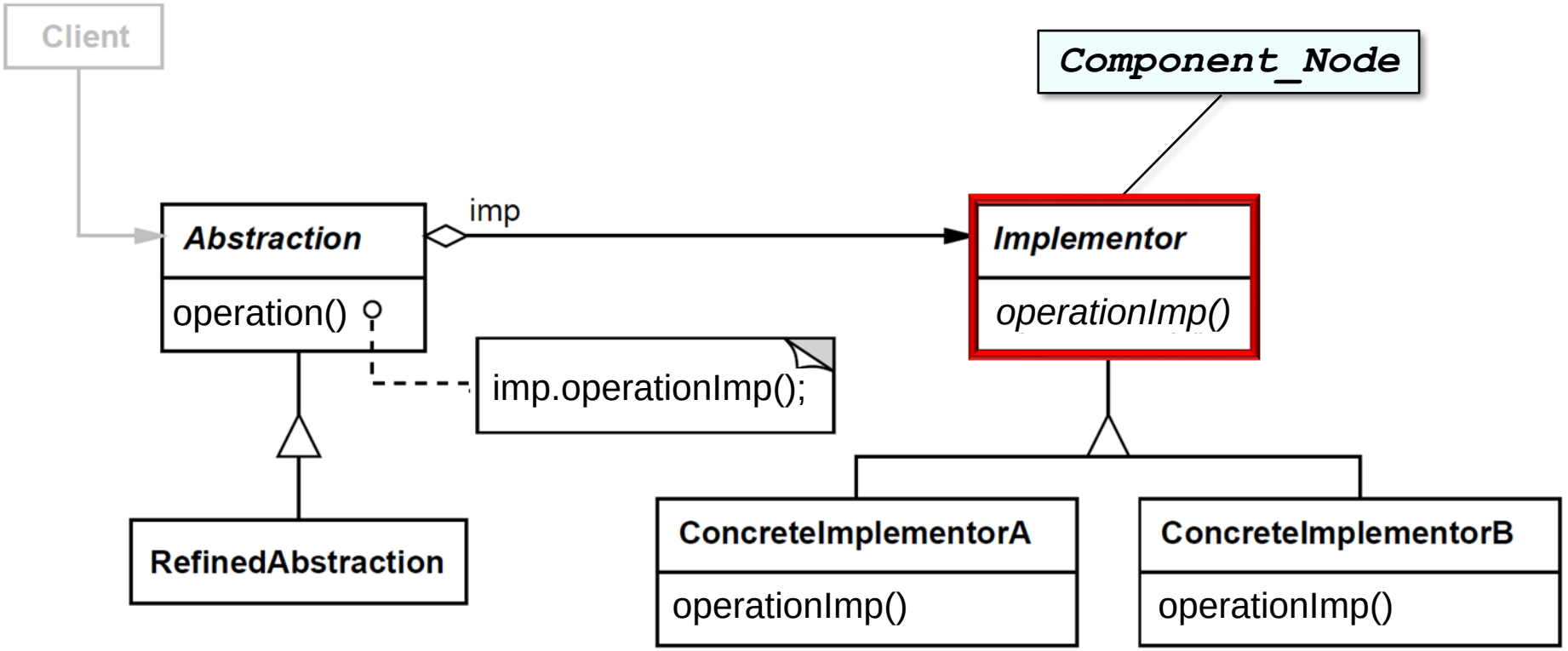
Structure & participants



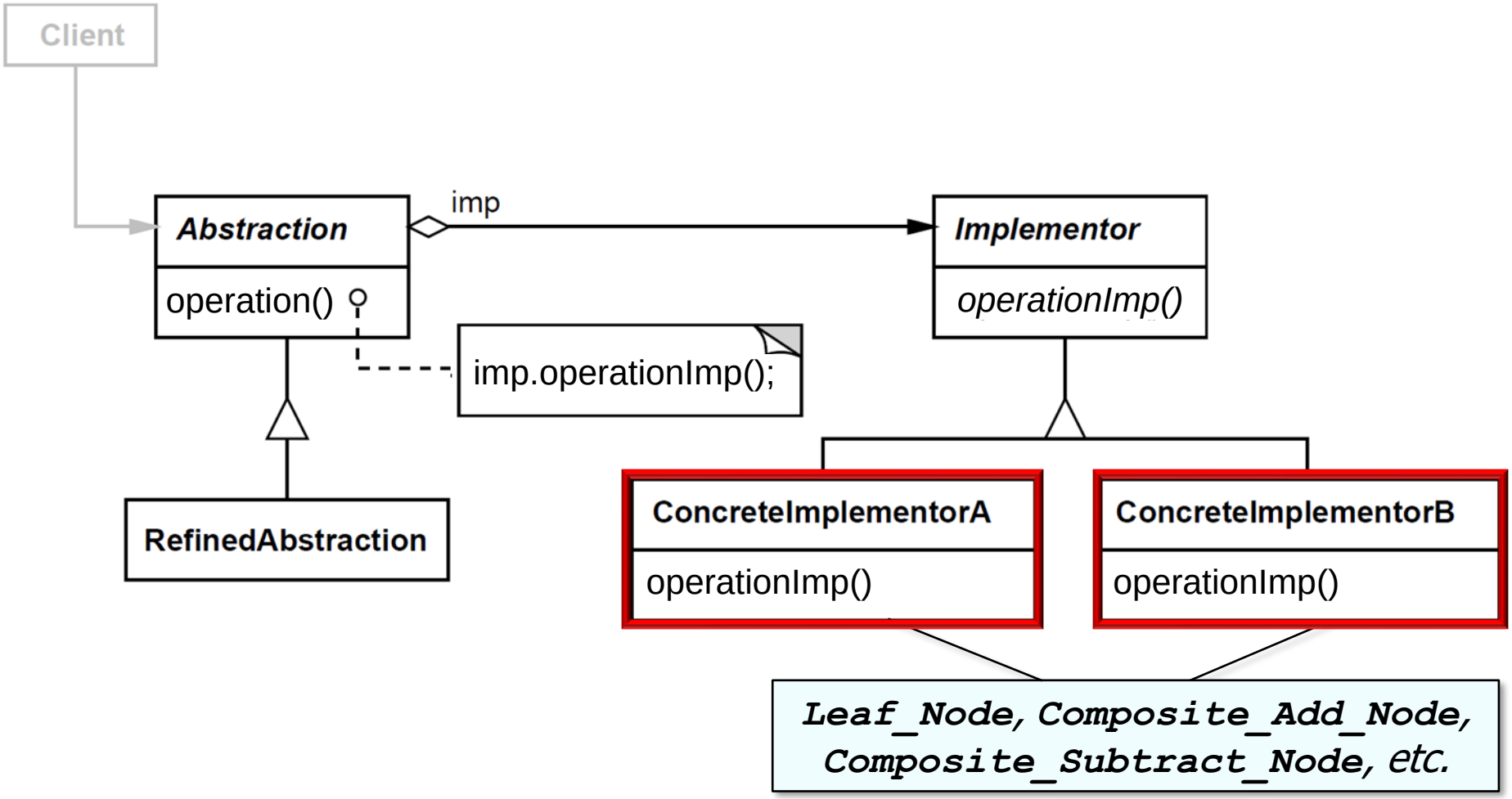
Structure & participants



Structure & participants



Structure & participants



Structure & participants