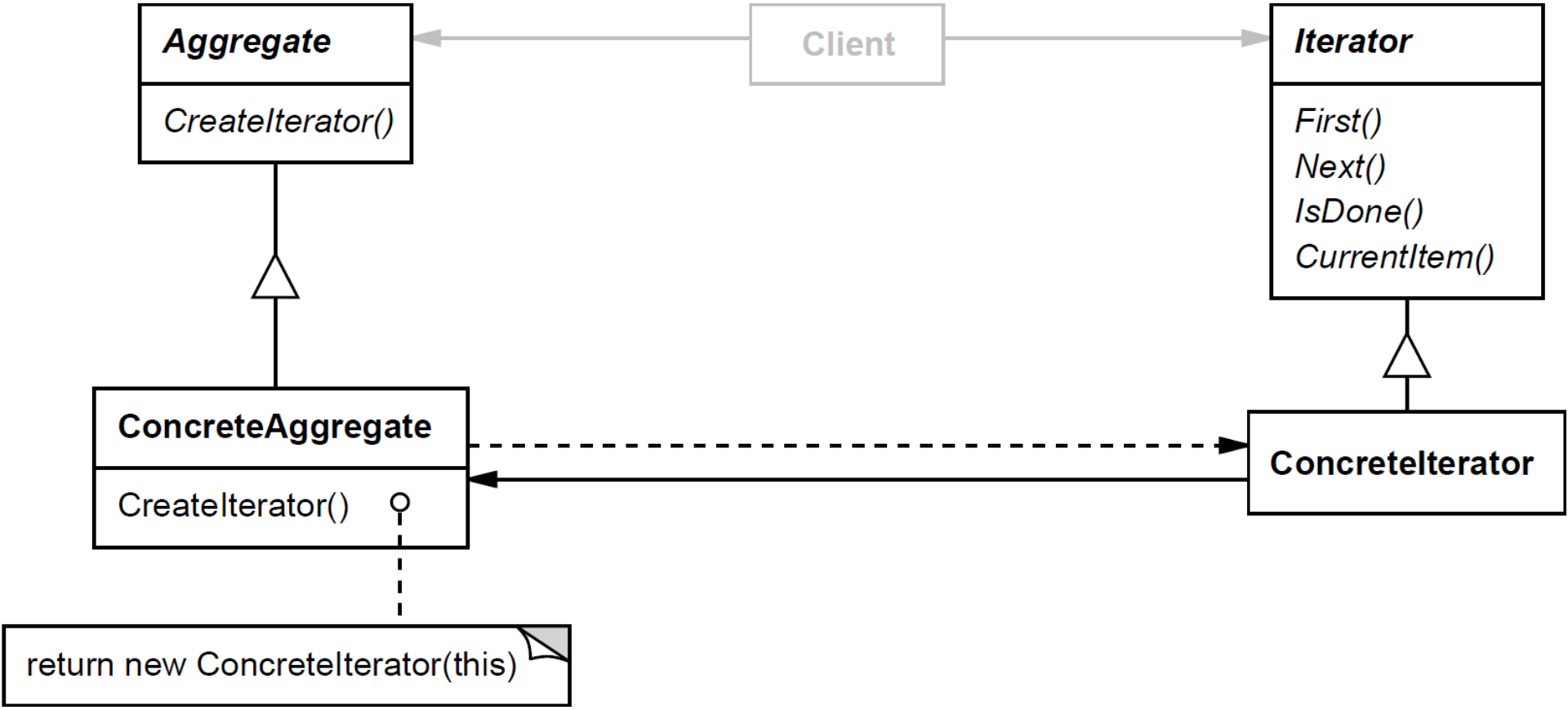
The Iterator Pattern

Structure & Functionality

Douglas C. Schmidt

Learning Objectives in This Lesson

- Recognize how the *Iterator* pattern can be applied to access all nodes in an expression tree flexibly & extensibly.
- Understand the structure & functionality of the *Iterator* pattern.



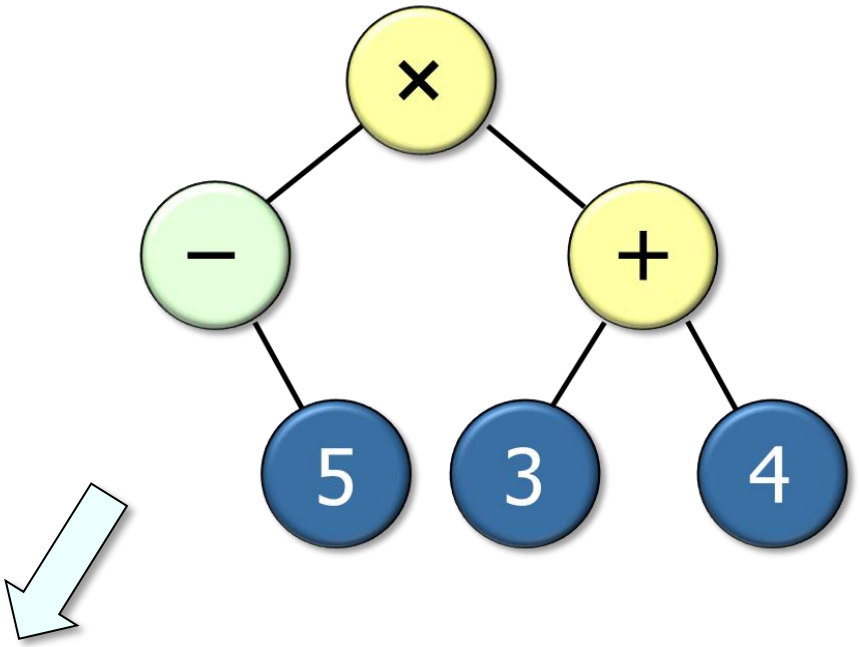
Douglas C. Schmidt

Structure & Functionality of the Iterator Pattern

Iterator — GoF Object Behavioral

Intent

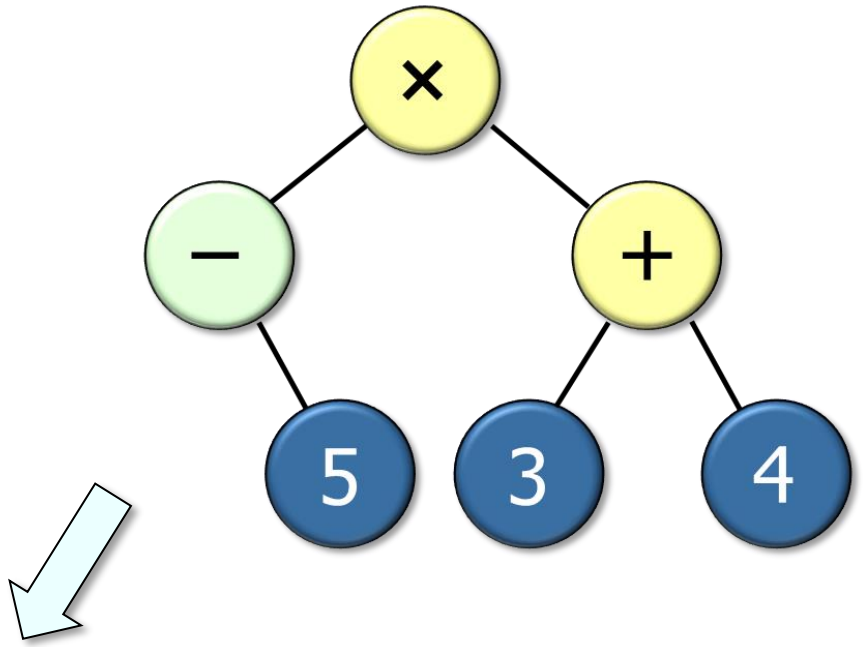
- Access elements of an aggregate without exposing its representation



- "In-order" traversal = `-5×(3+4)`
- "Pre-order" traversal = `×-5+34`
- "Post-order" traversal = `5-34+×`
- "Level-order" traversal = `×-+534`

Applicability

- Require multiple traversal algorithms over an aggregate



- "In-order" traversal = `-5×(3+4)`
- "Pre-order" traversal = `×-5+34`
- "Post-order" traversal = `5-34+×`
- "Level-order" traversal = `×-+534`

Applicability

- Require multiple traversal algorithms over an aggregate
- Require a uniform traversal interface over different aggregates

properties	valid expressions
copy-constructible, *copy-assignable* and *destructible*	`X b(a);` `b = a;`
Can be incremented	`++a` `a++`
Supports equality/inequality comparisons	`a == b` `a != b`
Can be dereferenced as an *rvalue*	`*a` `a->m`
Can be dereferenced as an *lvalue* (only for *mutable iterator types*)	`*a = t` `*a++ = t`
default-constructible	`X a;` `X()`
Multi-pass: neither dereferencing nor incrementing affects dereferenceability	`{ b=a; *a++;` `*b; }`

Can be decremented	`--a`
	`a--`
	`*a--`

Supports arithmetic operators + and –	`a + n`
	`n + a`
	`a - n`
	`a - b`

Supports inequality comparisons (<, >, <= and >=) between iterators	`a < b`
	`a > b`
	`a <= b`
	`a >= b`

Supports compound assignment operations += and –=	`a += n` `a -= n`
Supports offset dereference operator ([])	`a[n]`

Applicability

- Require multiple traversal algorithms over an aggregate
- Require a uniform traversal interface over different aggregates
- When aggregate classes & traversal algorithm(s) must vary independently

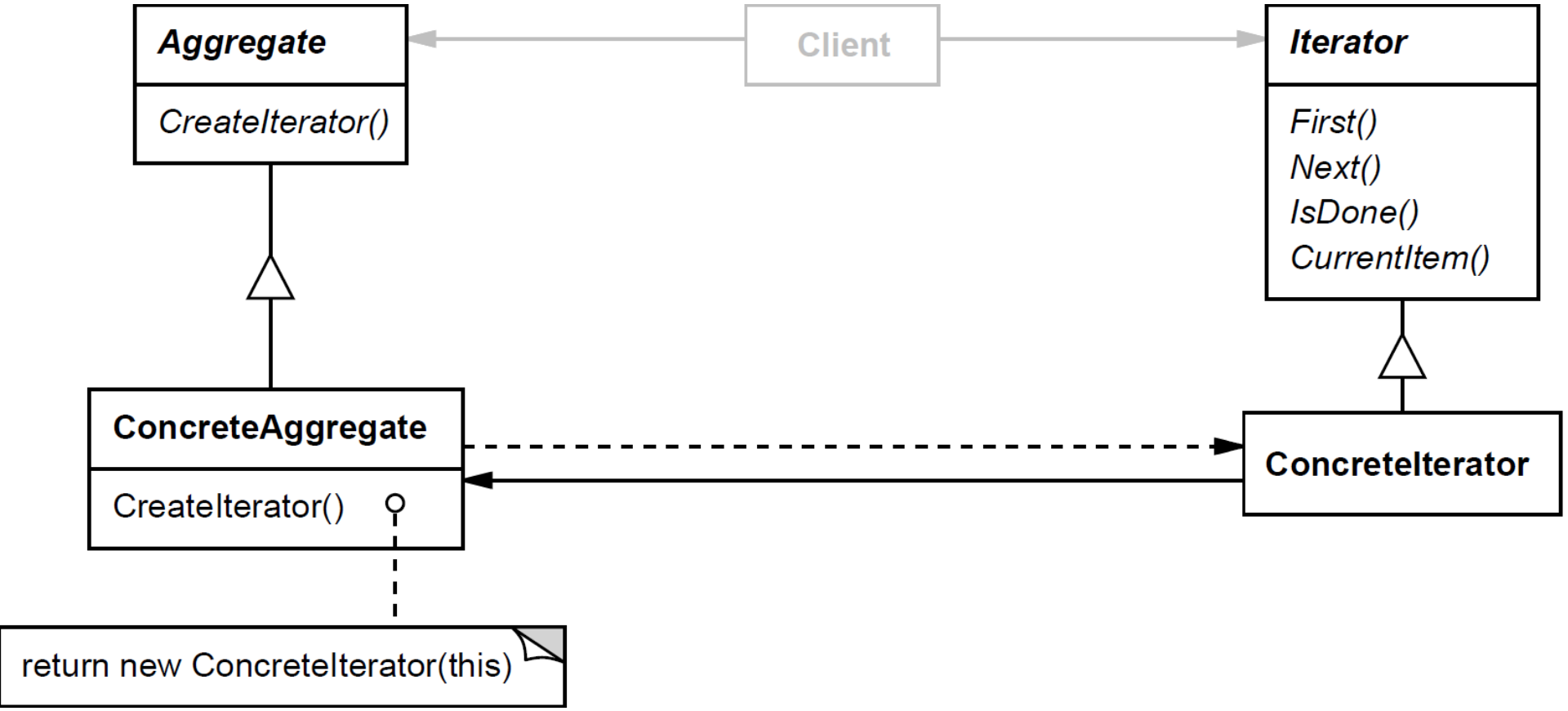
Properties	Containers

`++It, It++, *It`
`It == It2, It != It2`
	`std::unordered_set`
	`std::unordered_map`
	`std::unordered_multiset`
	`std::unordered_multimap`
	`std::forward_list`

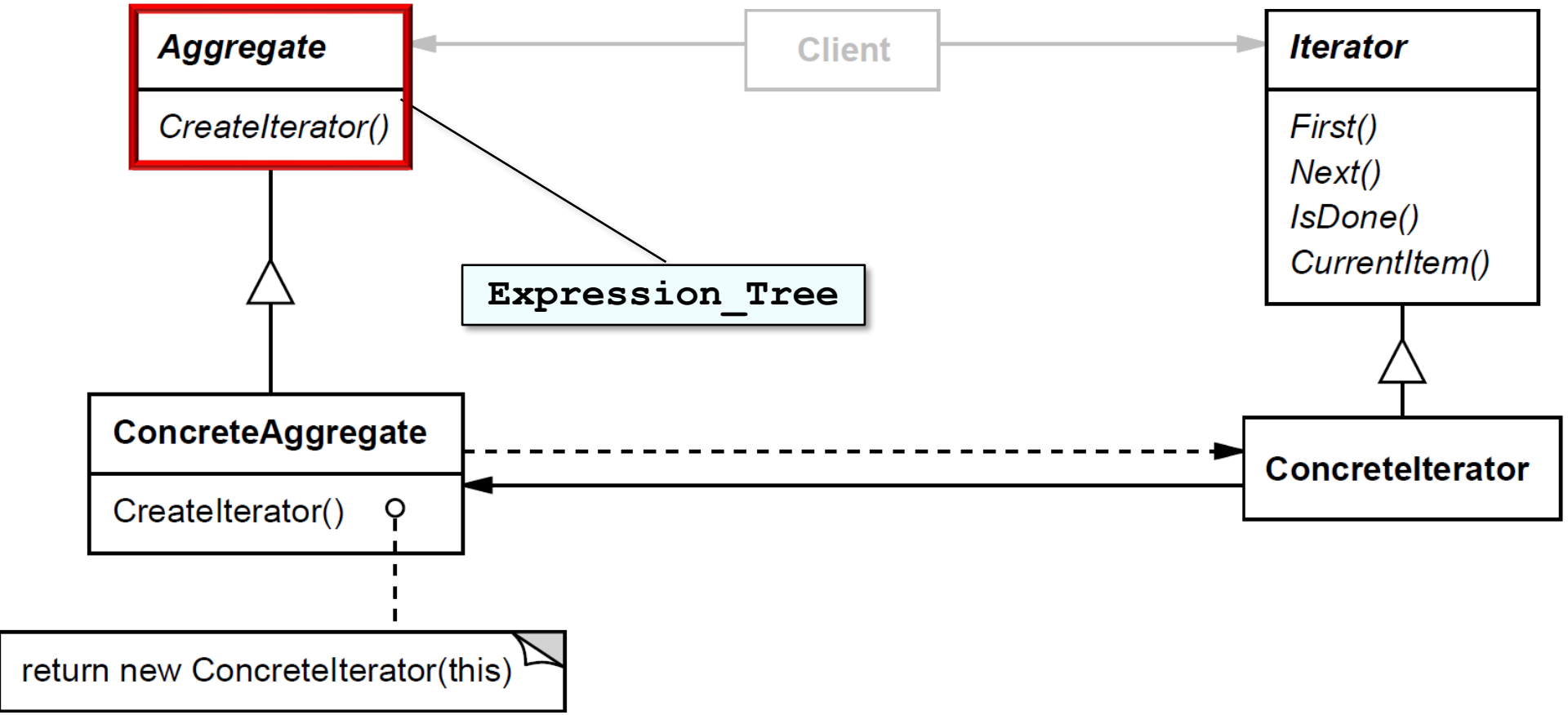
`--It, It--`	`std::set`
	`std::map`
	`std::multiset`
	`std::multimap`
	`std::list`

`It[i]`
`It += n, It -= n`
`It + n, It - n`
`n + It`
`It - It2`
`It < It2, It <= It2,`
`It > It2, It >= It2`
	`std::array`
	`std::vector`
	`std::deque`
	`std::string`

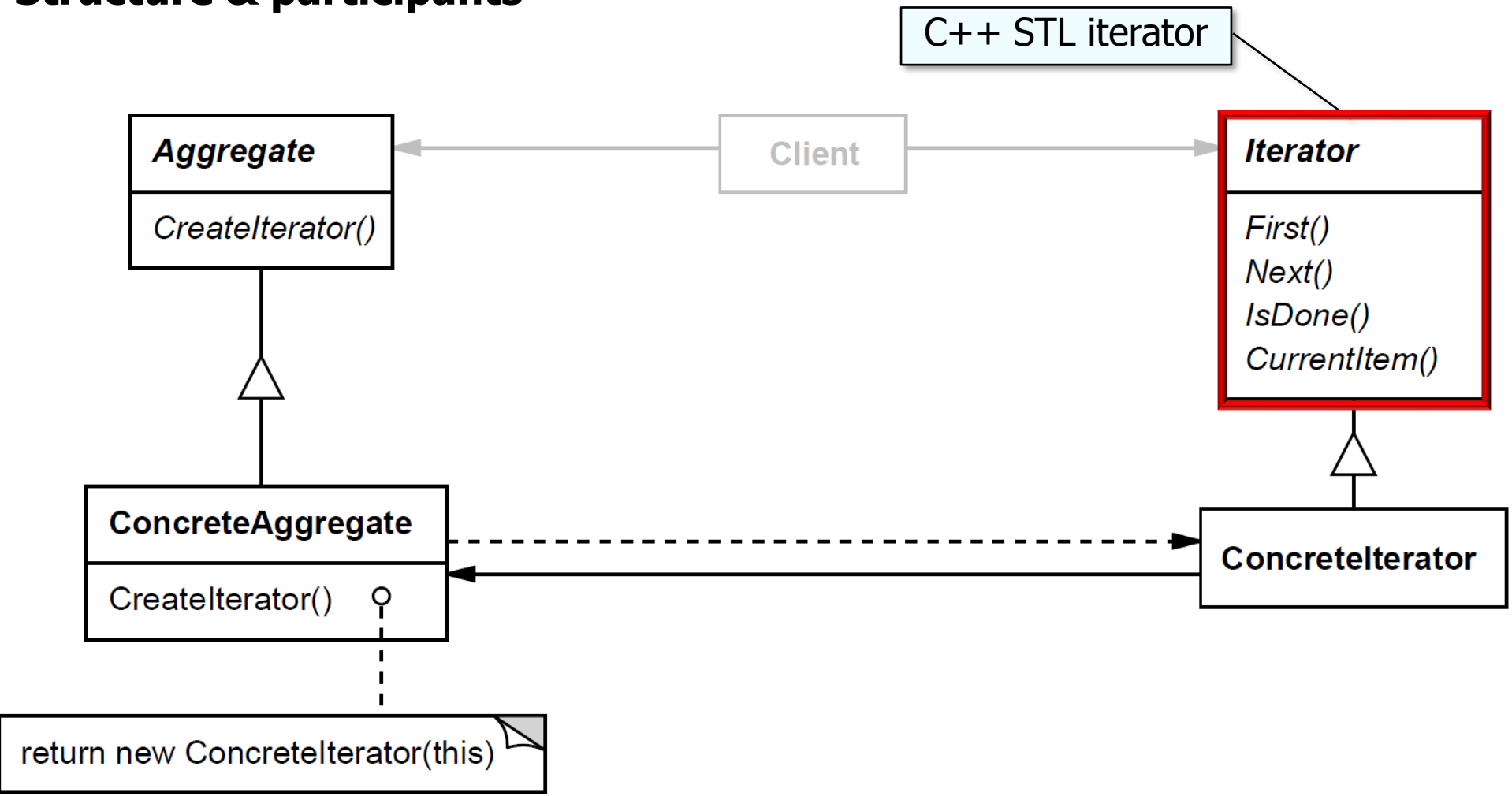
Structure & participants



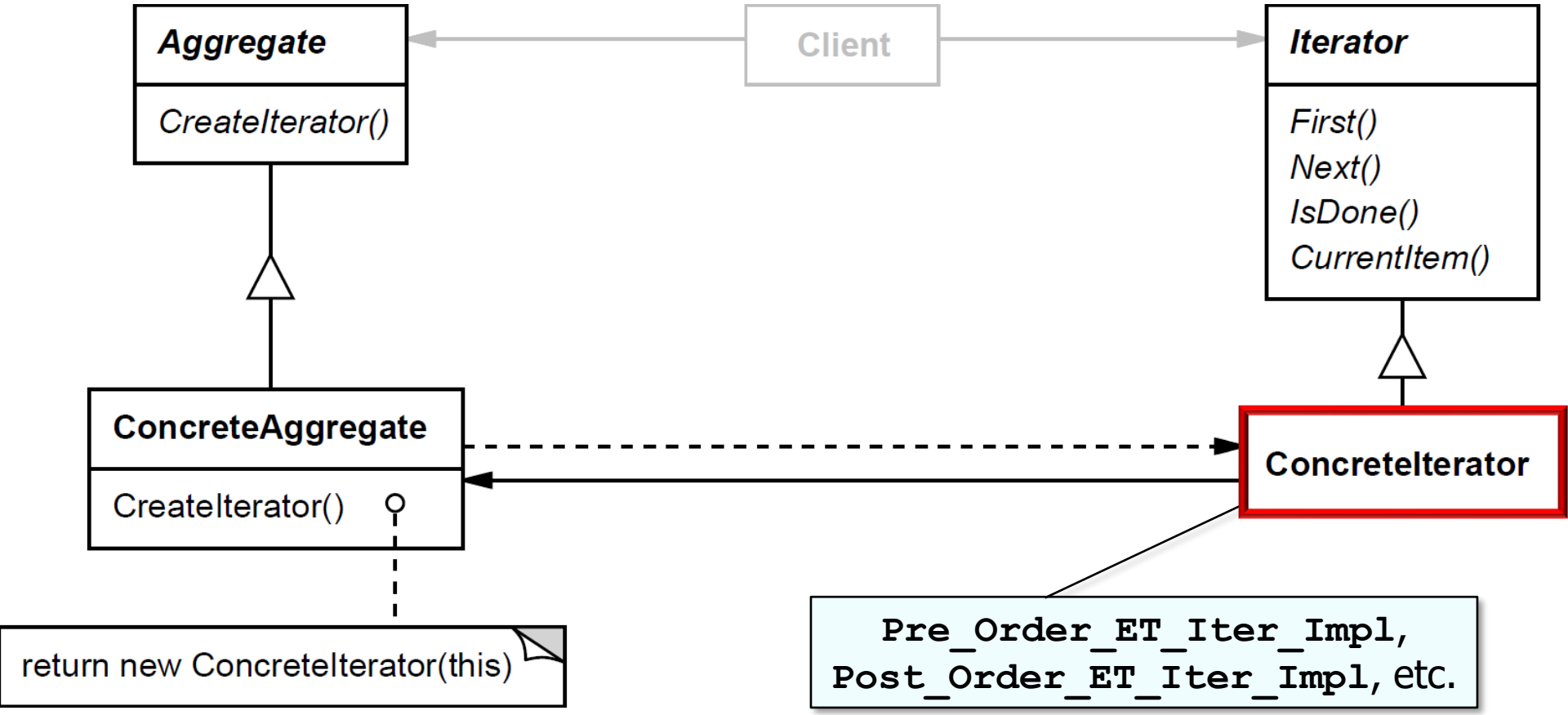
Structure & participants



Structure & participants



Structure & participants



Structure & participants