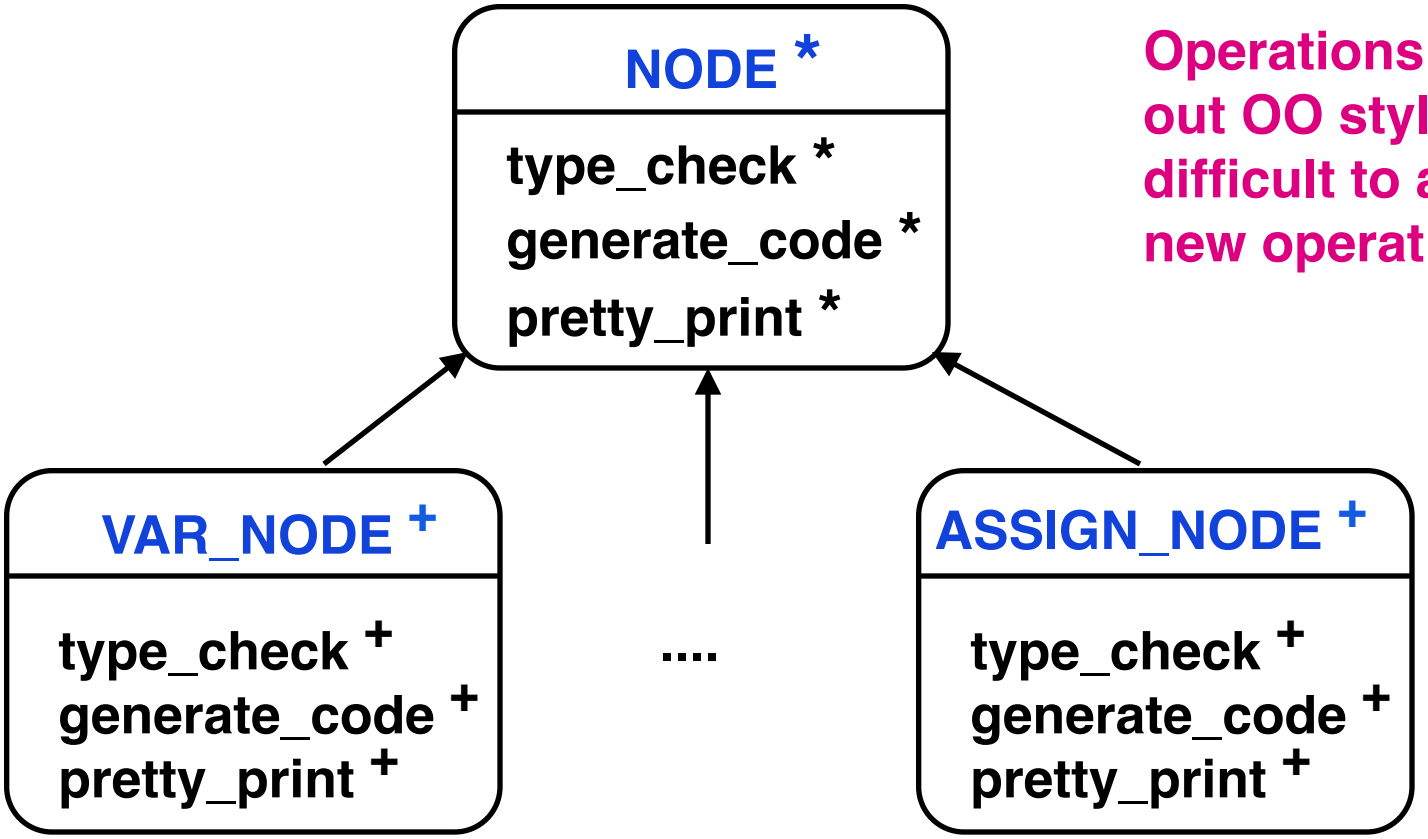
Visitor Pattern – Behavioural

- Intent
  - » **Represent an operation to be performed on all of the components of an object structure**
  - » **Define new operations on a structure without changing the classes representing the components**

© Gunnar Gotshalks

Motivation

- Compiler using an abstract syntax tree

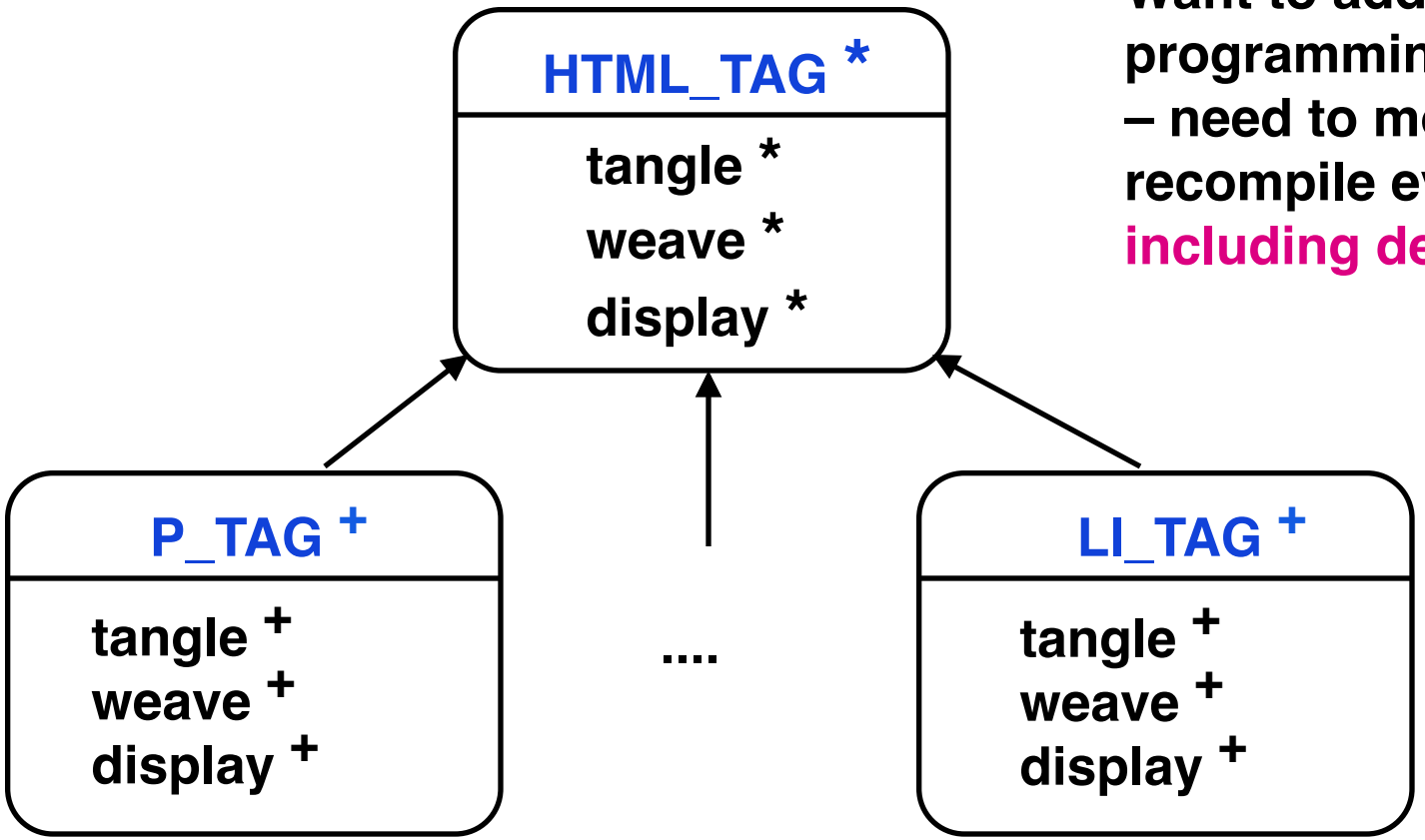


Operations spread out OO style but ... difficult to add a new operation

© Gunnar Gotshalks

Motivation – 2

- Consider programs to process HTML tags

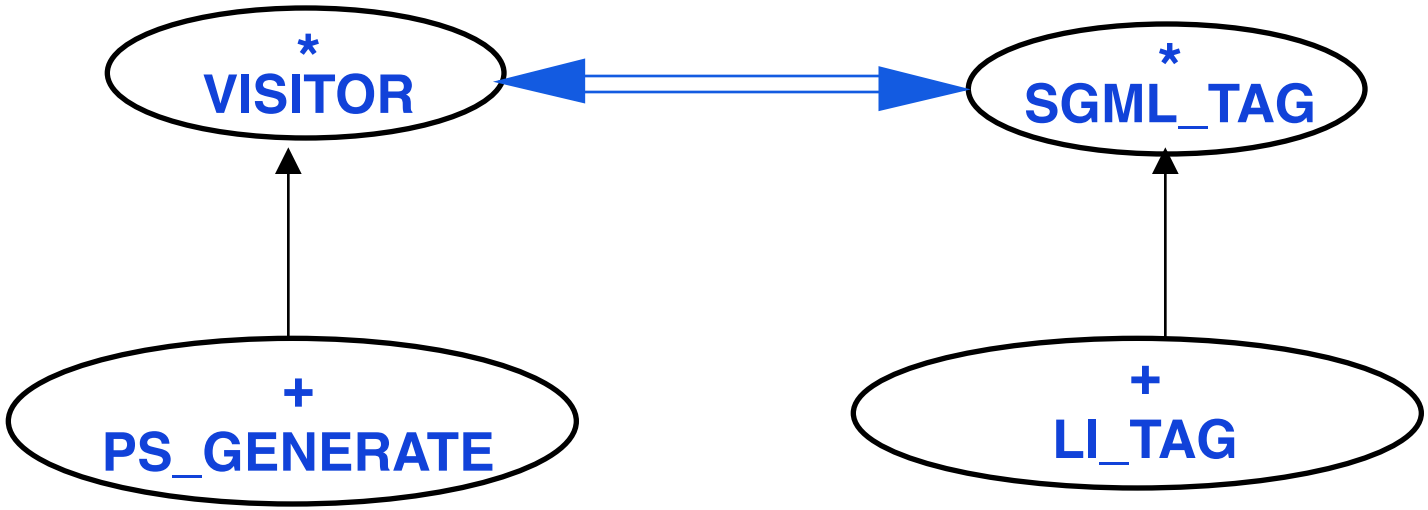


Want to add literate programming processors – need to modify all classes recompile everything including deferred classes

© Gunnar Gotshalks

SGML Example Architecture

- Tag classes are independent of every operation
- Nodes accept visitors and direct them to the appropriate operation – through polymorphism

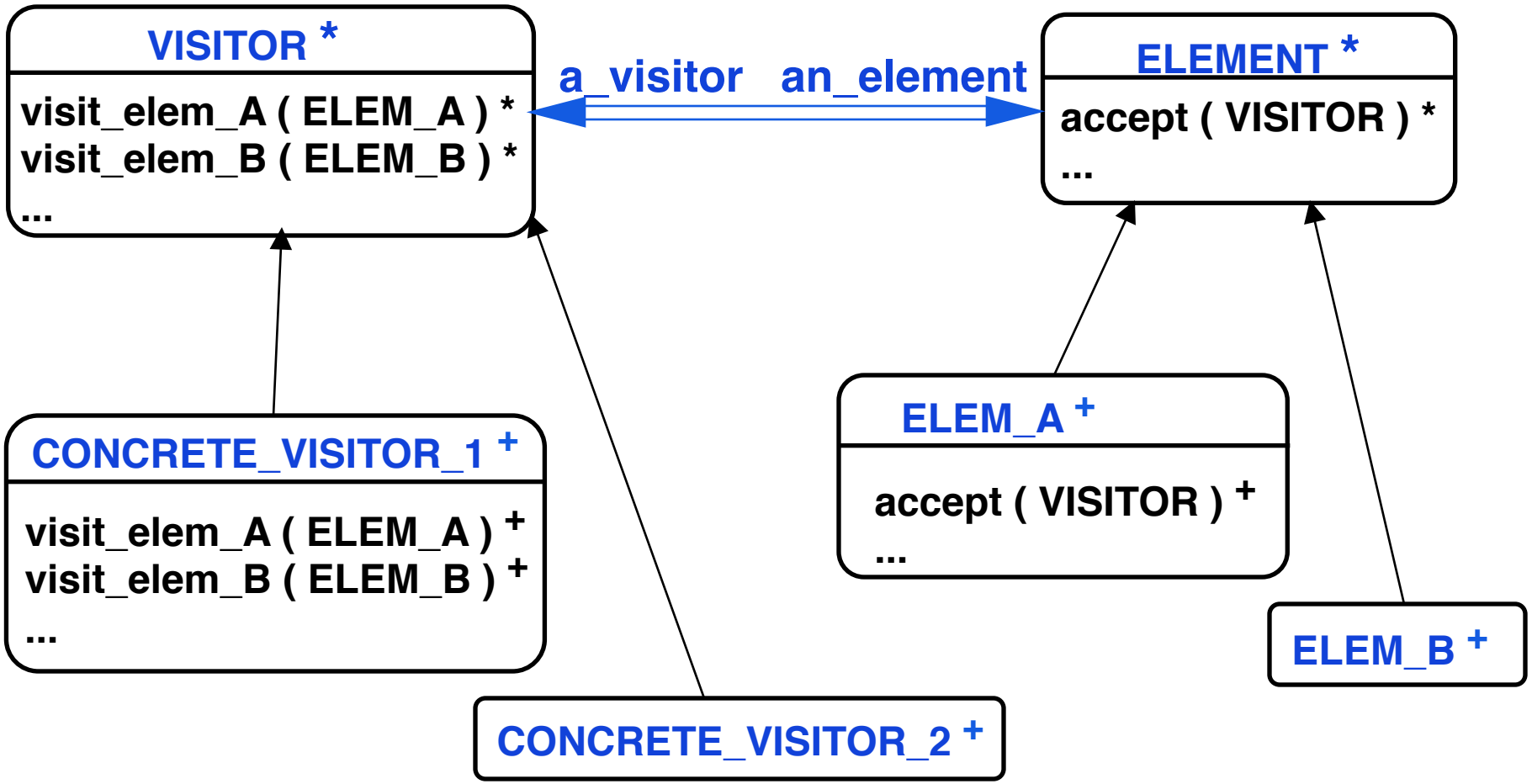


One visitor subclass for each operation

Same structure for each tag

© Gunnar Gotshalks

Abstract Architecture

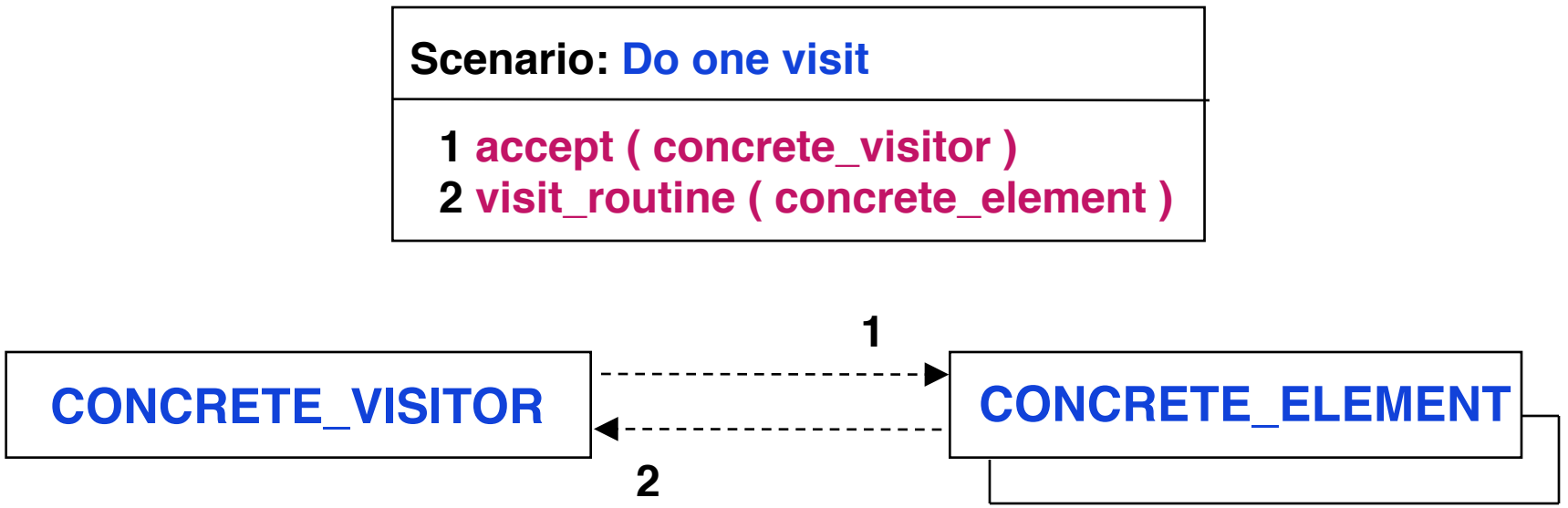


© Gunnar Gotshalks

Scenario

- Concrete visitor loops over the elements
- For each element concrete visitor

Selects which method in the visitor to execute



© Gunnar Gotshalks

Participants

- Element

  Declares accept method for Visitors

- Concrete element

  Implements accept method for Visitors

- Visitor

  Declares visit operation for each concrete element class

© Gunnar Gotshalks

Participants – 2

- Concrete visitor
  - » **Implements every visit operation declared by the Visitor**
    - **Each visit operation implements a fragment of the algorithm defined for the concrete visitor**
  - » **Provides the context for the algorithm composed of all of the visit fragments**
    - **State accumulates with each visit**
  - » **Implements the high level organization**
    - > **Iteration over the components**
    - > **Processing each in turn**

© Gunnar Gotshalks

Applicability

- Object structure contains many classes of objects with differing interfaces and want to perform operations that depend on their concrete classes
- Many distinct and unrelated operations need to be performed
  - » **Do not want to or are unable to clutter the concrete classes with these operations**
  - » **Keep the related sub-operations (specific to each concrete class) together**
  - » **Put operations into only those applications that need them**

© Gunnar Gotshalks

Applicability – 2

- The classes defining the object structure rarely change, but you often want to define new operations over the structure

  Changing object structure means

  Redefining interface to all visitors, which is costly

© Gunnar Gotshalks

TAG Implementation

```
deferred class HTML_TAG feature
    accept ( visitor : VISITOR) deferred end
 ... -- other features ...
end
```

```
class LI_TAG inherit  HTML_TAG feature
    accept (visitor : VISITOR) do
        visitor.visit_li_tag ( Current )
    end
 ... -- other features ...
end
```

© Gunnar Gotshalks

VISITOR Implementation

```
deferred class VISITOR feature

   -- Have one "visit" routine for each tag (component)

   visit_LI_TAG ( tag : LI_TAG) deferred end
   visit_P_TAG ( tag : P_TAG) deferred end
   visit_UL_TAG (tag : UL_TAG) deferred end
   ...
end
```

© Gunnar Gotshalks

Concrete Visitor Implementation

```
class CONCRETE_VISITOR inherit  VISITOR feature
    get_elements    -- Attaches elements to the iterator
    while not elements.allDone do
        elements.item.accept ( Current )
        elements.next
    end

    visit_li_tag (tag : LI_tag) do semantic action ... end
    visit_ul_tag (tag : UL_tag) do semantic action ... end
    visit_p_tag (tag : P_tag) do semantic action ... end

   ... -- and all the rest of the tags

end
```

© Gunnar Gotshalks

Consequences

- Adding new operations is easy
  - » **New operation implements visitor interface for the components**
  - » **All the fragments of the visitor algorithm are in one file – related behaviours are together**
    - **Easier to make sure that components are working in unison**
  - » **Unrelated operations and fragments are in other visitor classes**
  - » **Contrast with having to change each of the component classes to have the operation fragment**
    - **Each class has a fragment of each of the operations**

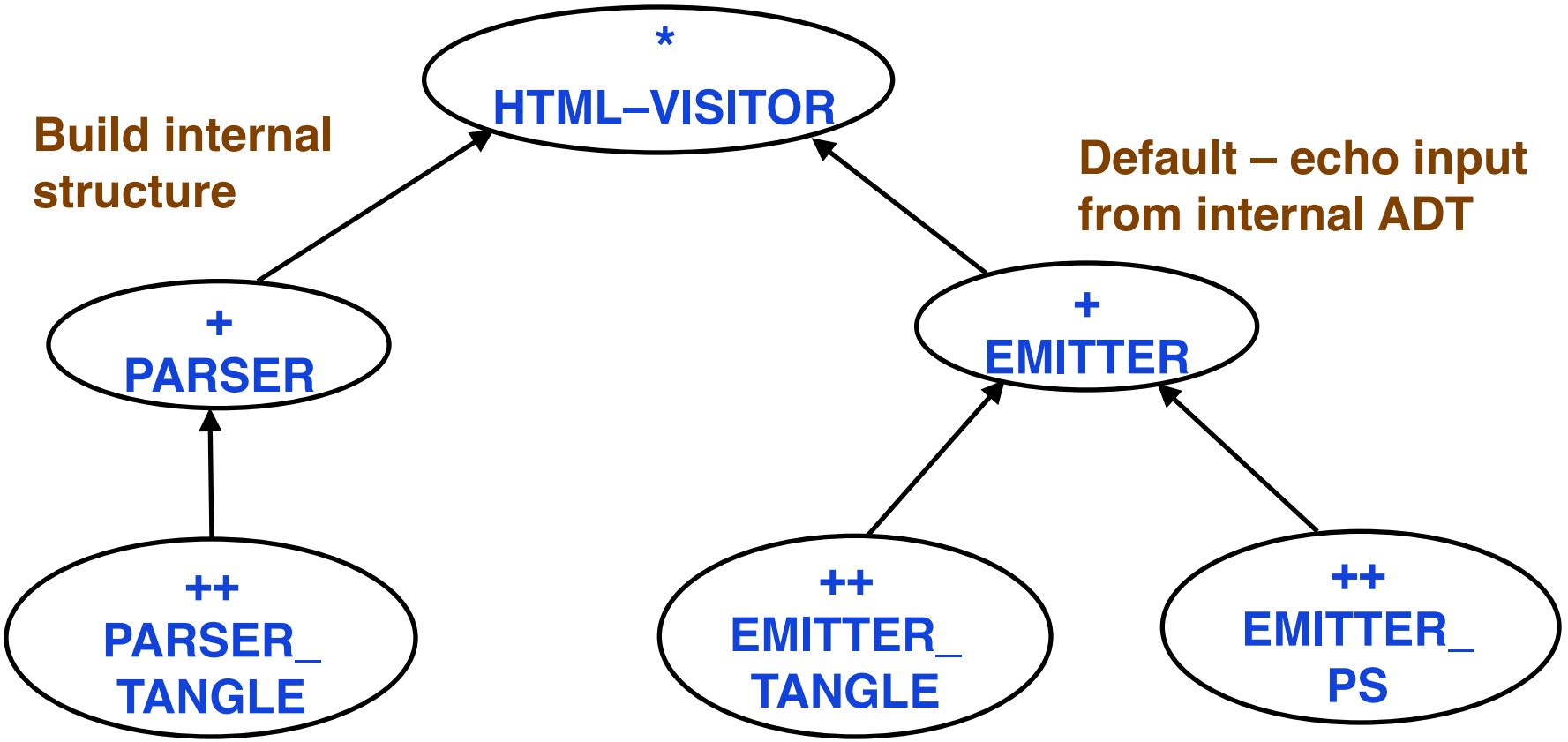
© Gunnar Gotshalks

Consequences – 2

- Adding new concrete elements is difficult
  - » **Need to modify visitor class**
  - » **Need to modify each concrete visitor**
    - > **Can sometimes simplify as many elements have common behaviour (default behaviour) that can be specified at concrete visitor level 1.**
    - > **Create subclasses of level 1 for more specific behaviour for the new elements**
      - – **Only program the new elements**
  - » **For many structures components do not change rapidly so this is not a problem**

© Gunnar Gotshalks

Example Multi-Level Visitor



© Gunnar Gotshalks

Consequences – 3

- Works across class hierarchies
  - » **Contrast with Iterator Pattern**
  - » **Contrast with multi-panel & do-undo applications**

© Gunnar Gotshalks

Related Patterns

- Visitor pattern is used to apply an operation over a Composite
- Visitor pattern is used to do the interpretation for an Interpreter pattern
- Visitor pattern is used to process the items obtained by using the Iteration pattern to iterate over a collection.

© Gunnar Gotshalks