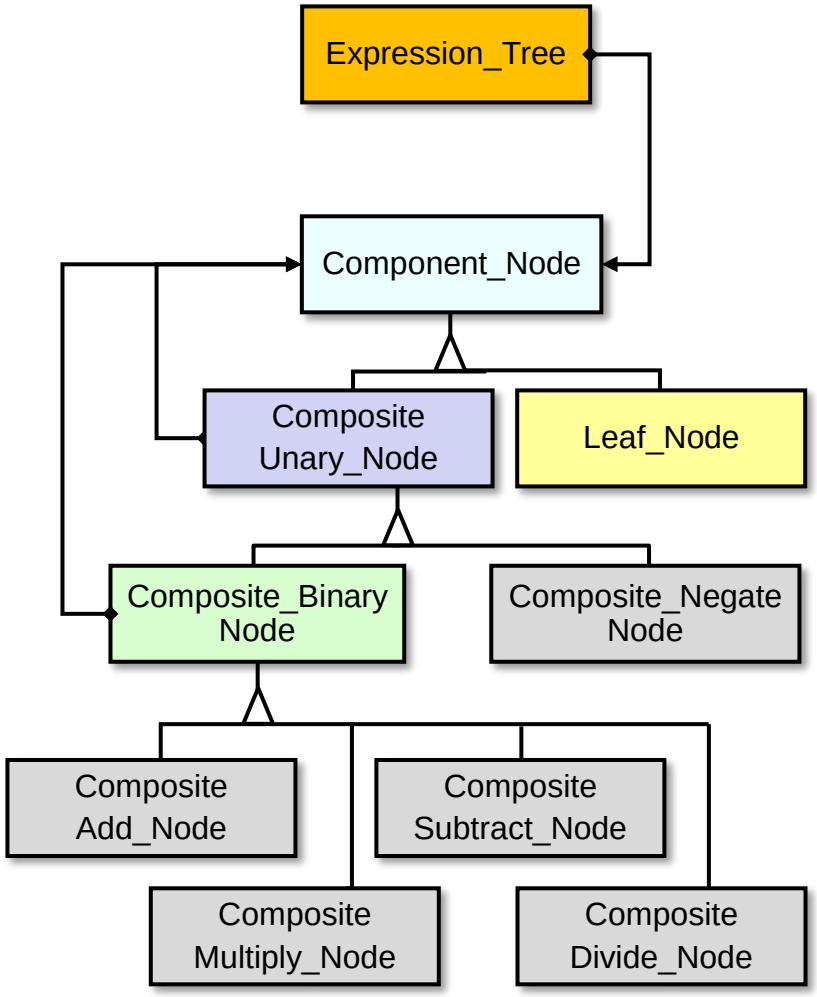
The Object-Oriented Design of the Expression Tree Processing App

Douglas C. Schmidt

Learning Objectives in This Lesson

- Understand the OO design of the expression tree processing app.



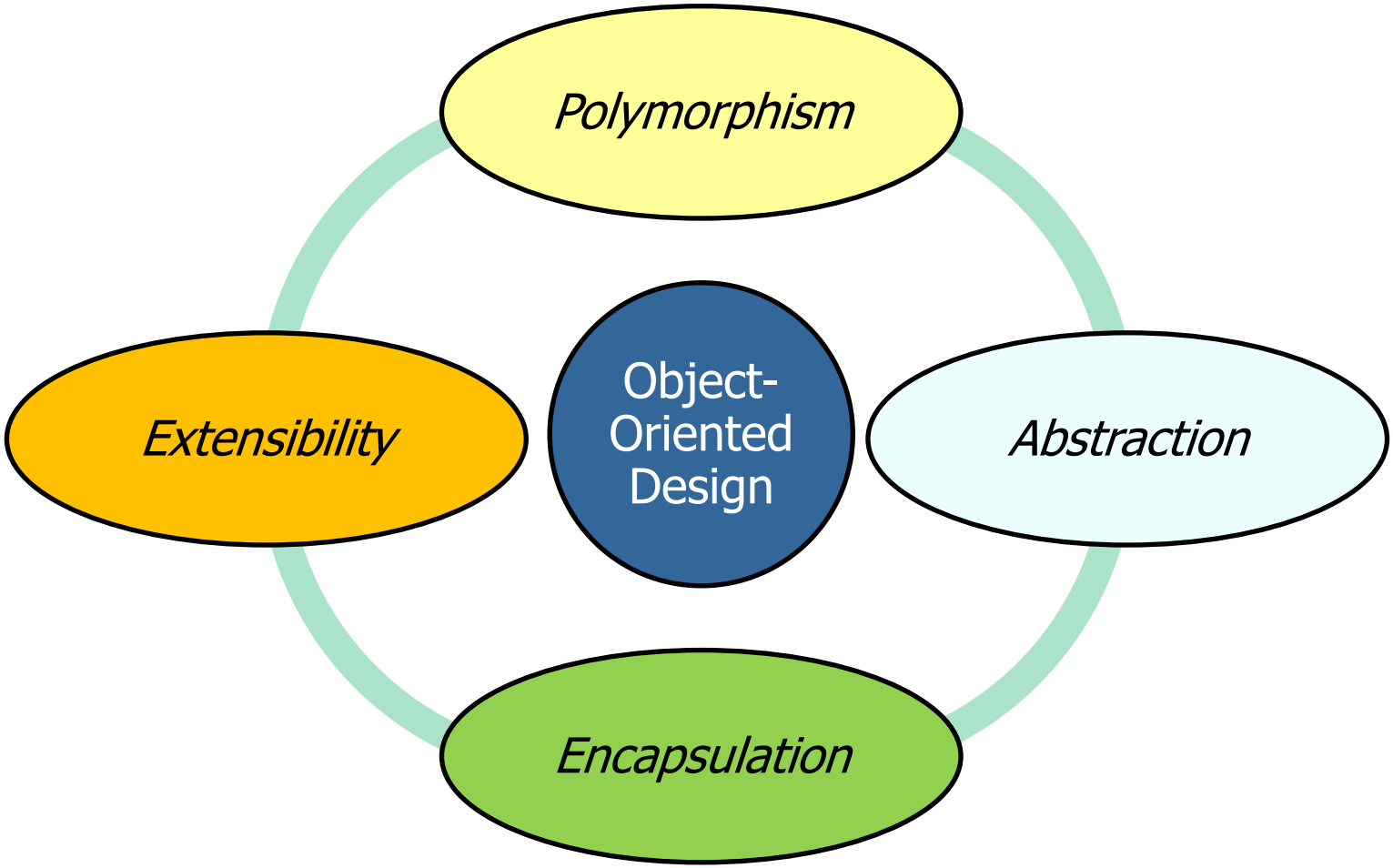
en.wikipedia.org/wiki/Unified_Modeling_Language has more on OOD notations.

Douglas C. Schmidt

Lesson Introduction

Lesson Introduction

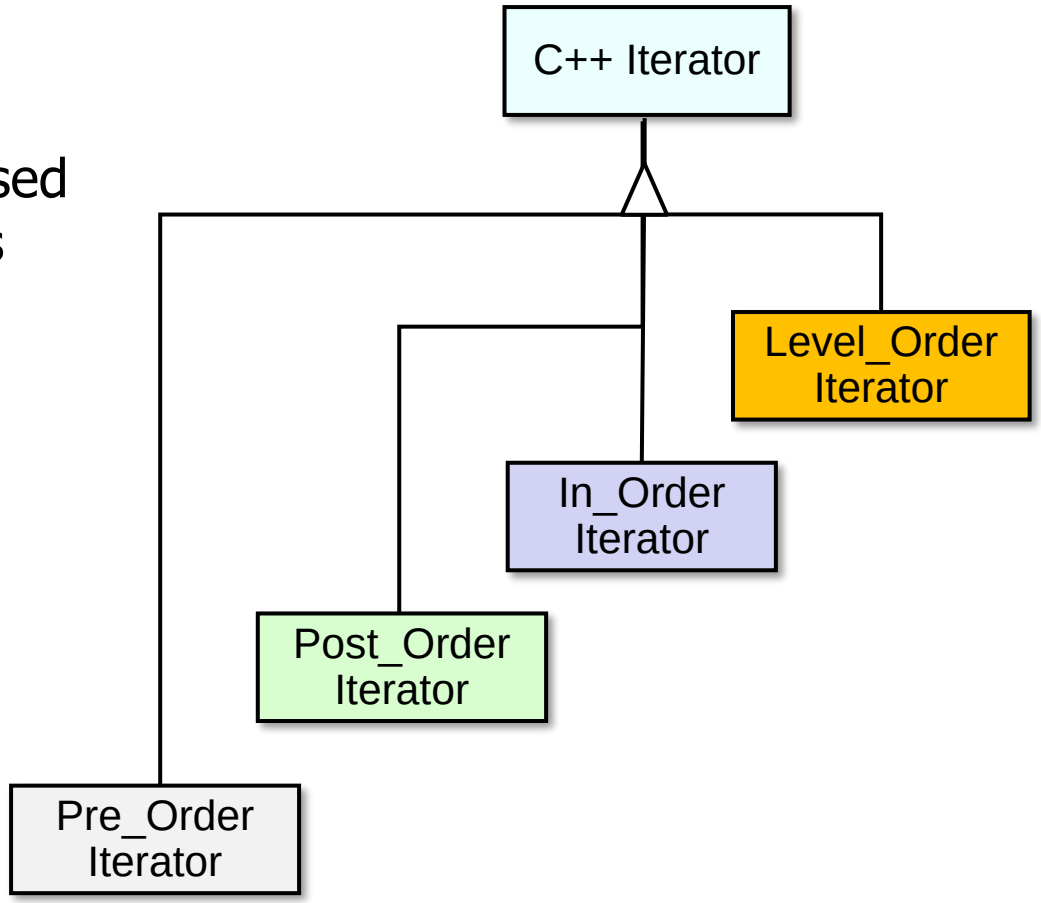
- Object-oriented design (OOD) is a method of planning a system of interacting objects to solve software problem(s).



en.wikipedia.org/wiki/Object-oriented_design has more information on OO design.

Lesson Introduction

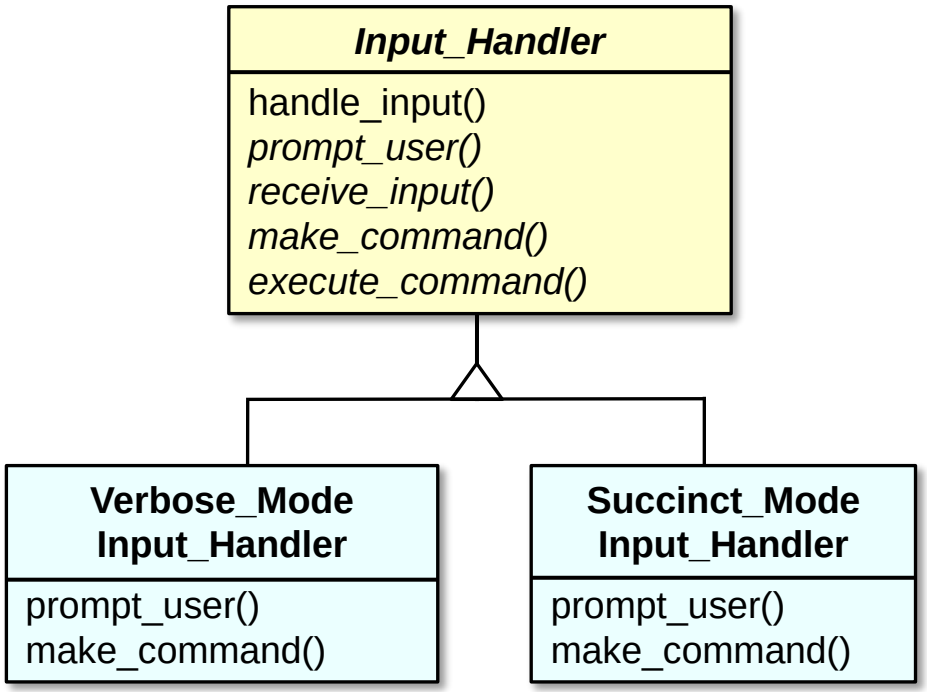
- Object-oriented design (OOD) is a method of planning a system of interacting objects to solve software problem(s).
- OOD employs "hierarchical data abstraction."
  - Components are designed based on stable *class* & *object* roles & relationships
    - Rather than functions corresponding to actions



en.wikipedia.org/wiki/Liskov_substitution_principle has more information.

Lesson Introduction

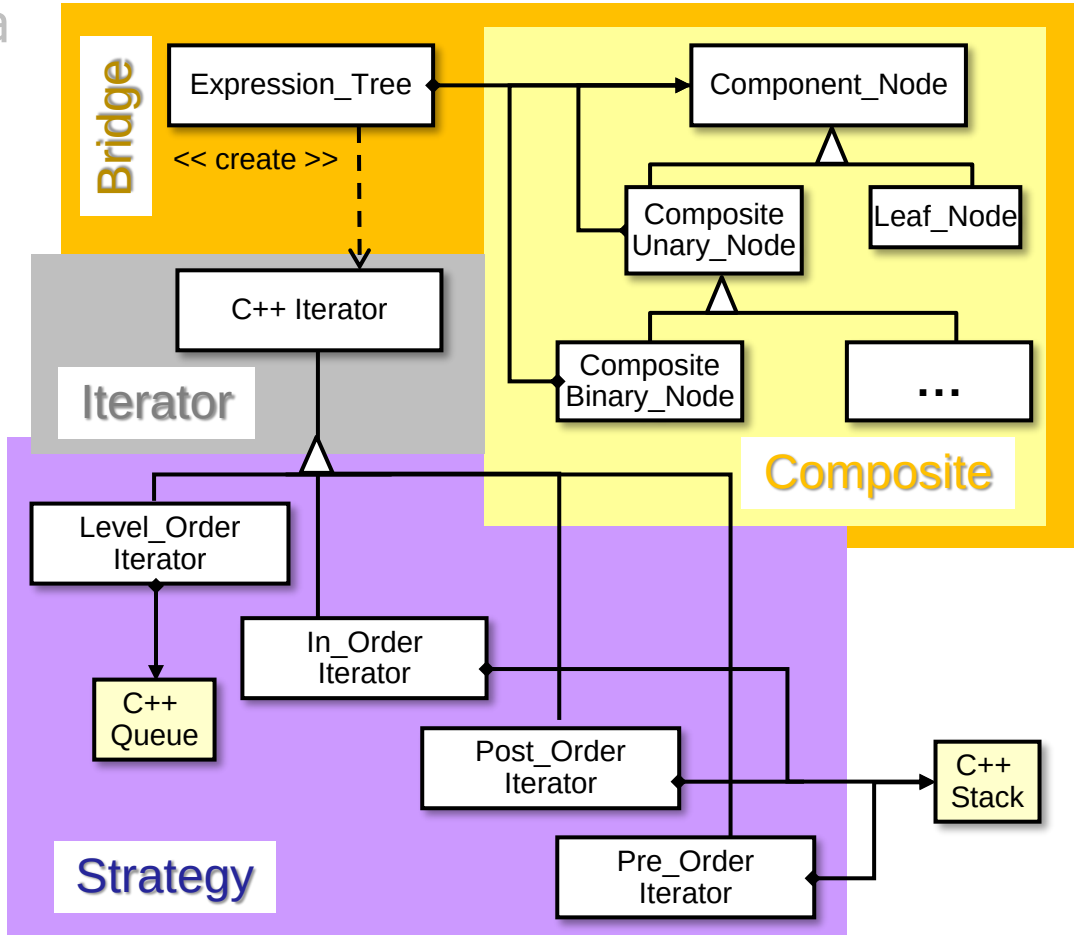
- Object-oriented design (OOD) is a method of planning a system of interacting objects to solve software problem(s).
- OOD employs "hierarchical data abstraction."
- It also associates actions with specific objects and/or classes of objects.
  - Emphasize *high cohesion* & *low coupling*



en.wikipedia.org/wiki/Low-Coupling_/_High-Cohesion_pattern has more information.

Lesson Introduction

- Object-oriented design (OOD) is a method of planning a system of interacting objects to solve software problem(s).
- OOD employs "hierarchical data abstraction."
- It also associates actions with specific objects and/or classes of objects.
- Well-designed OO programs group classes & objects via *patterns* & combine them to form *frameworks.*



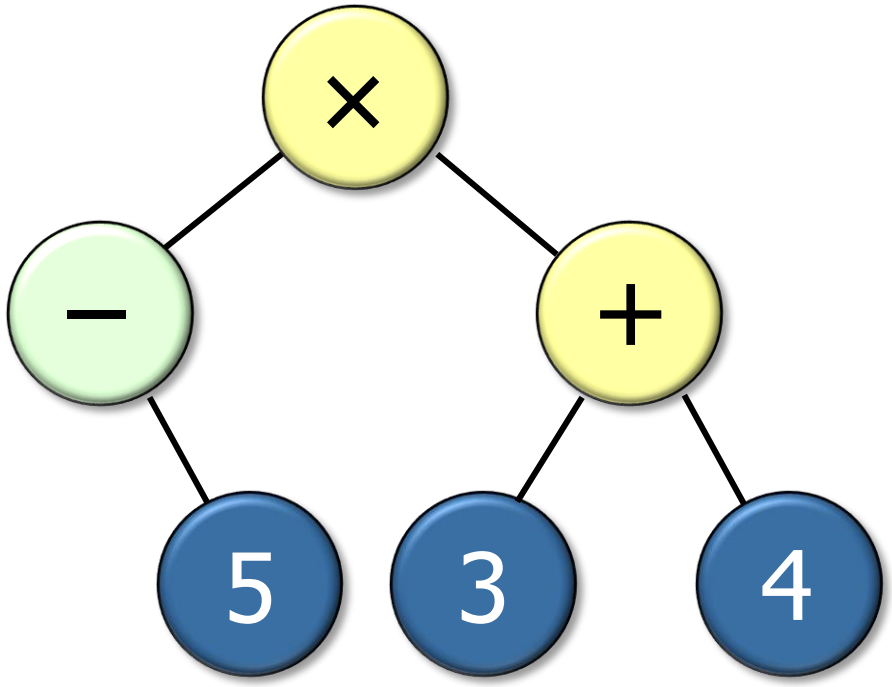
www.dre.vanderbilt.edu/~schmidt/patterns-frameworks.html has more information.

Douglas C. Schmidt

OO Design of Expression Tree Processing App

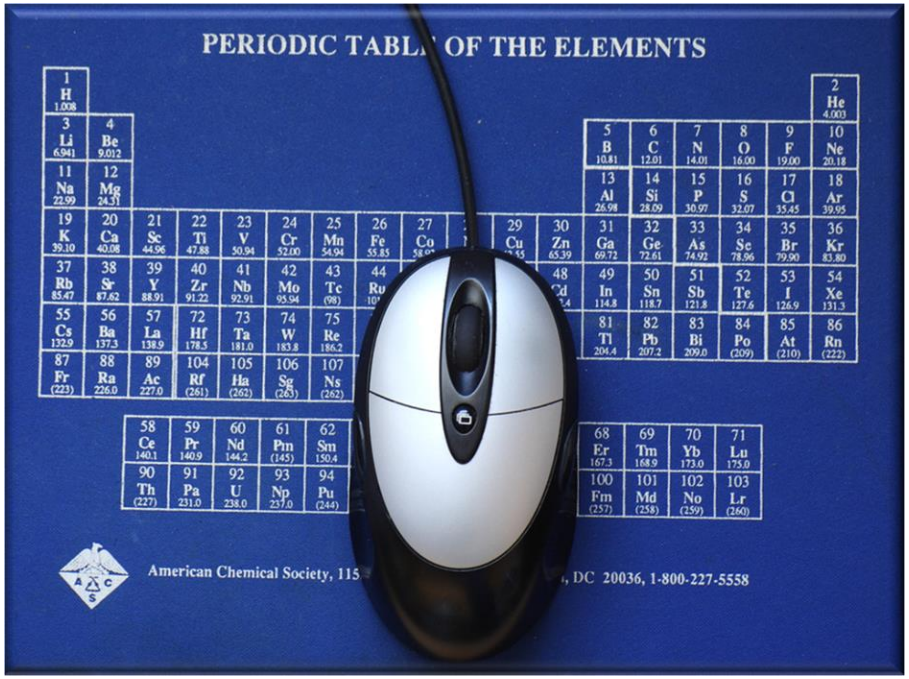
OO Design of Expression Tree Processing App

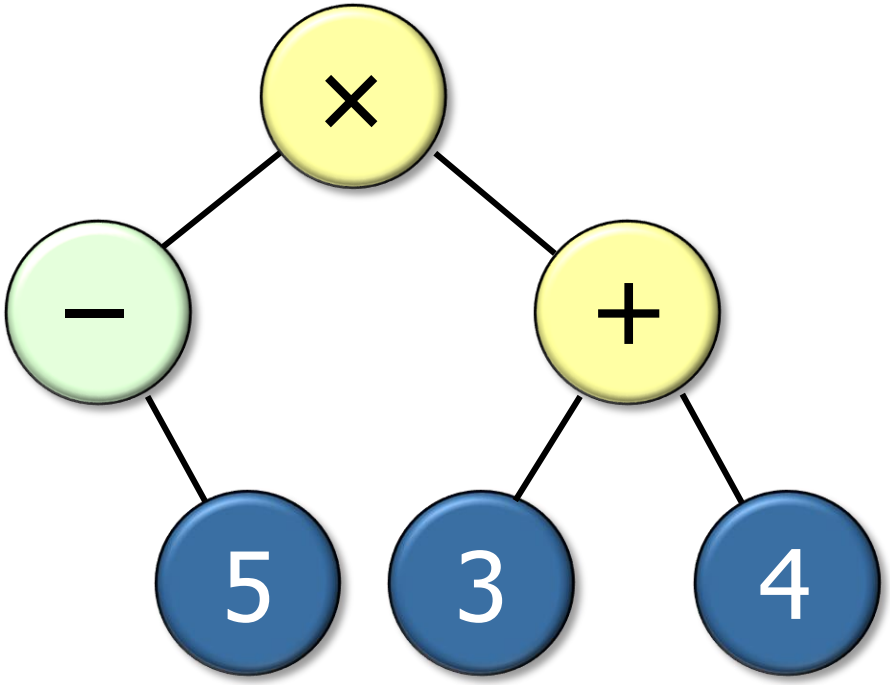
- Create an OO design based on modeling classes & objects in "expression tree" domain.



OO Design of Expression Tree Processing App

- Conduct *scope, commonality, & variability* analysis to determine stable APIs & variable extension points.

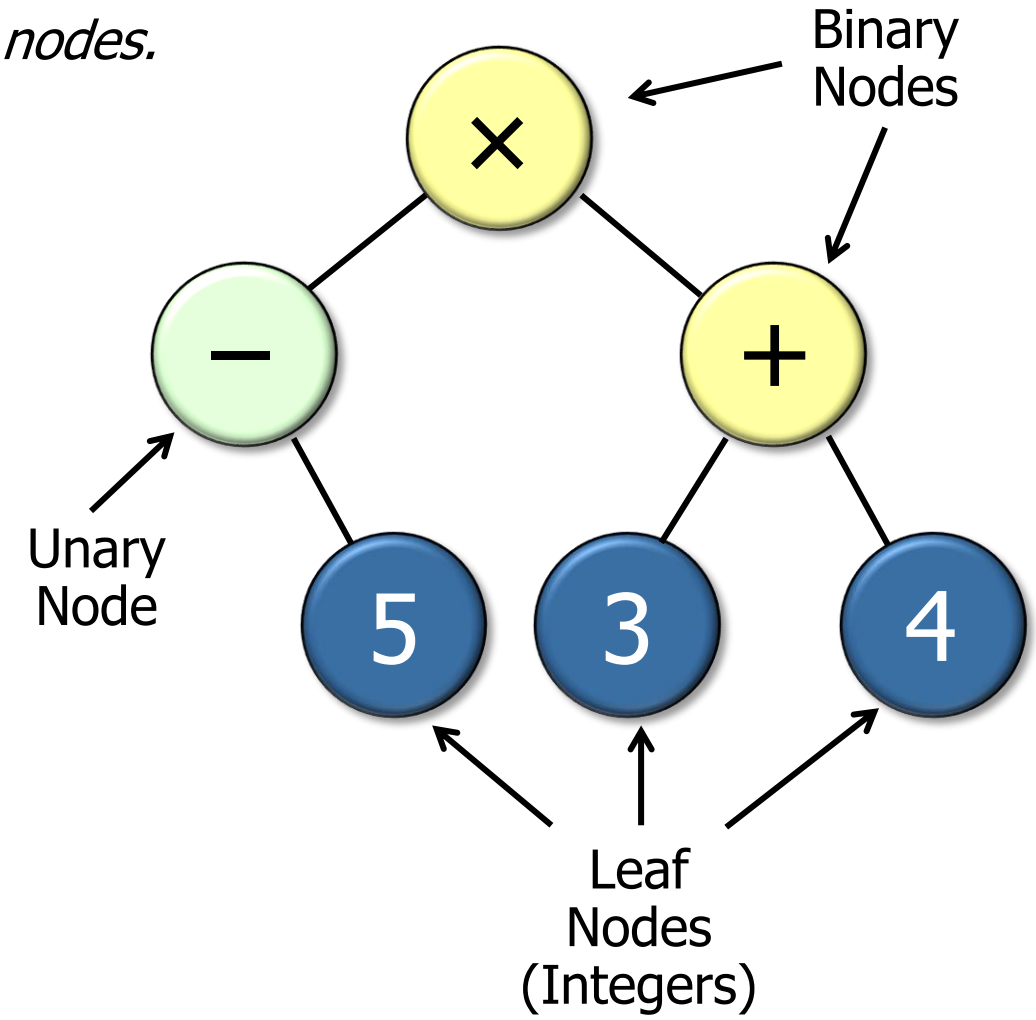




See www.dre.Vanderbilt.edu/~schmidt/PDF/Commonality_Variability.pdf

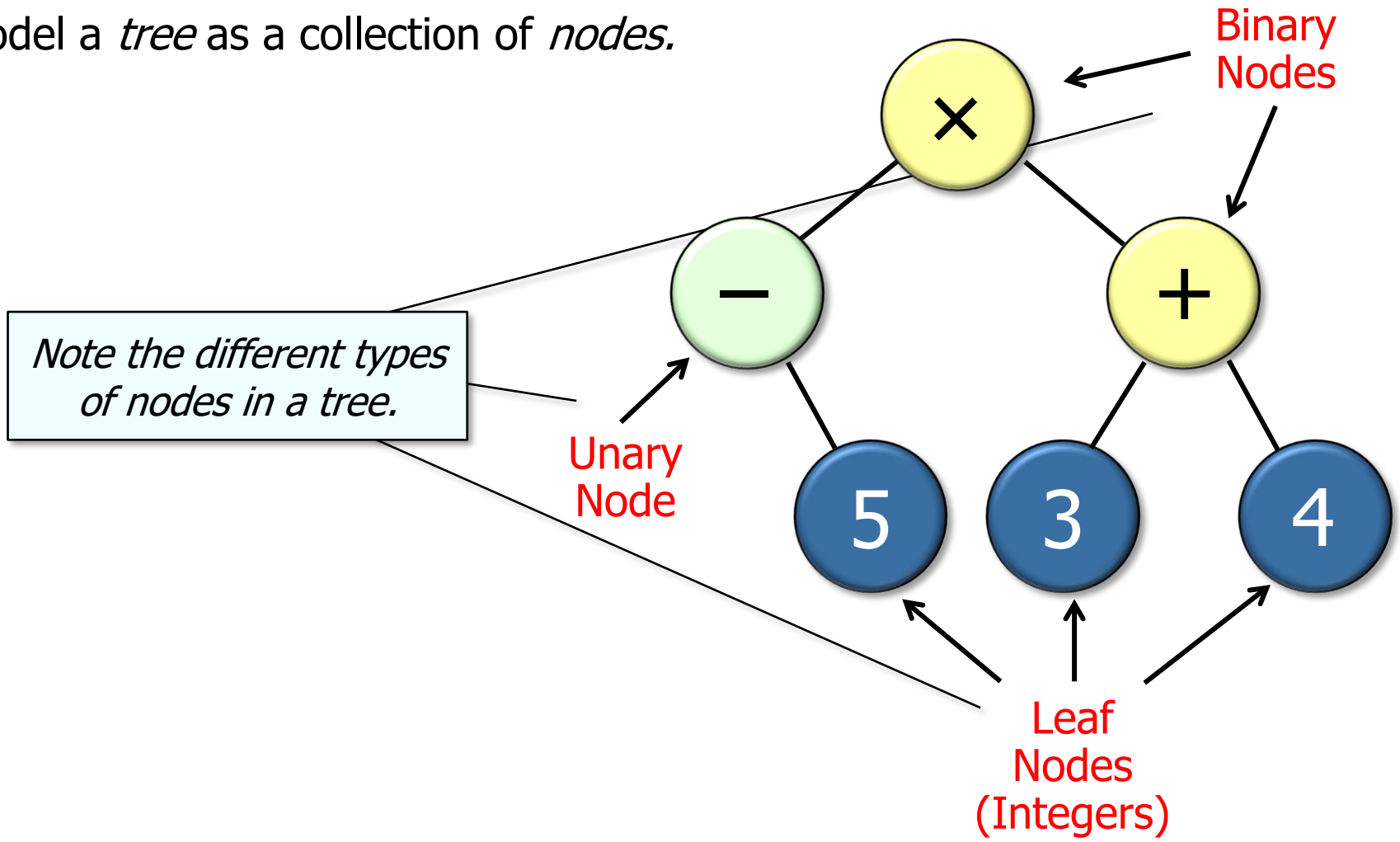
OO Design of Expression Tree Processing App

- Conduct *scope, commonality, & variability* analysis to determine stable APIs & variable extension points.
  - Model a *tree* as a collection of *nodes.*



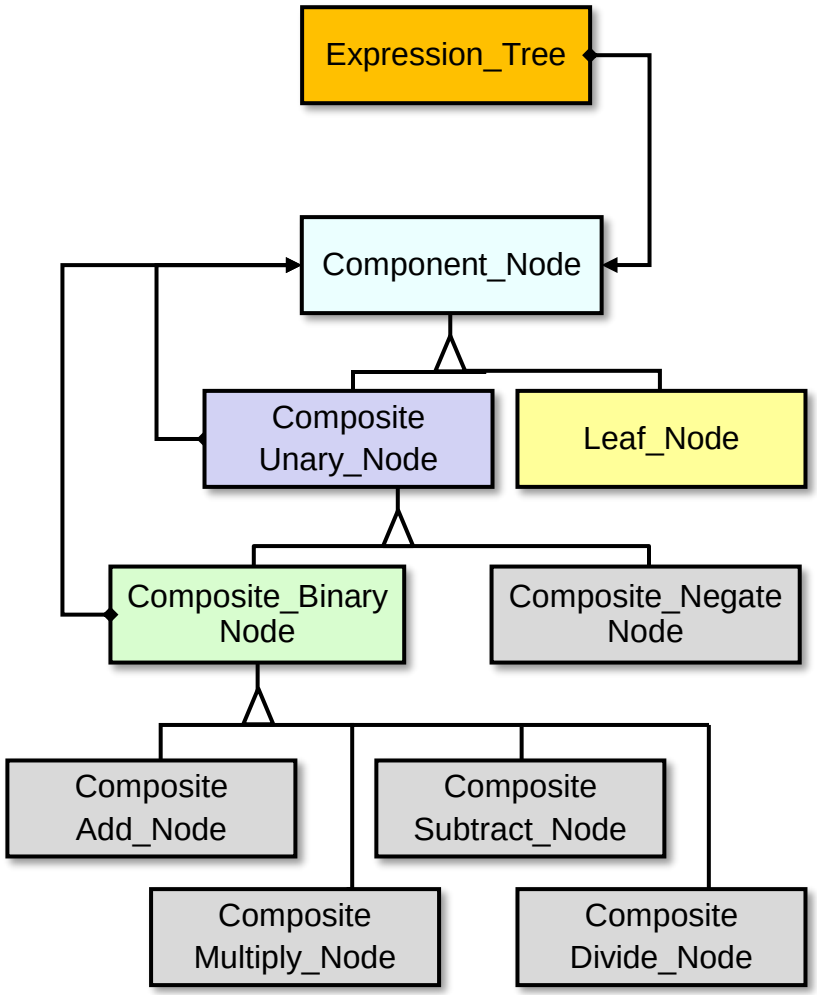
OO Design of Expression Tree Processing App

- Conduct *scope, commonality, & variability* analysis to determine stable APIs & variable extension points.
  - Model a *tree* as a collection of *nodes.*



OO Design of Expression Tree Processing App

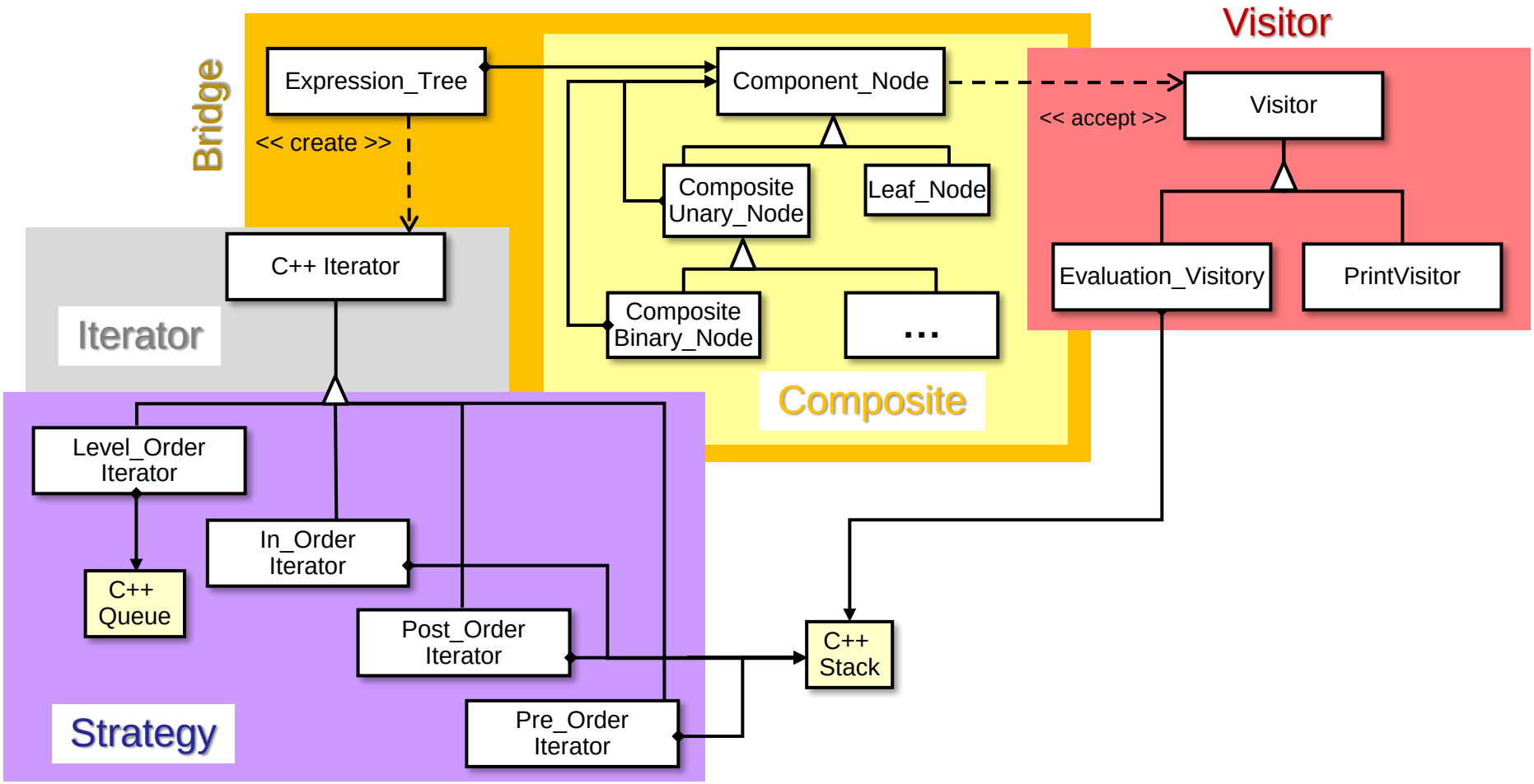
- Conduct *scope, commonality, & variability* analysis to determine stable APIs & variable extension points.
  - Model a *tree* as a collection of *nodes.*
  - Represent *nodes* as class hierarchy, capturing properties of each node.
    - e.g., the "arities" (binary & unary nodes)



See en.wikipedia.org/wiki/Arity

OO Design of Expression Tree Processing App

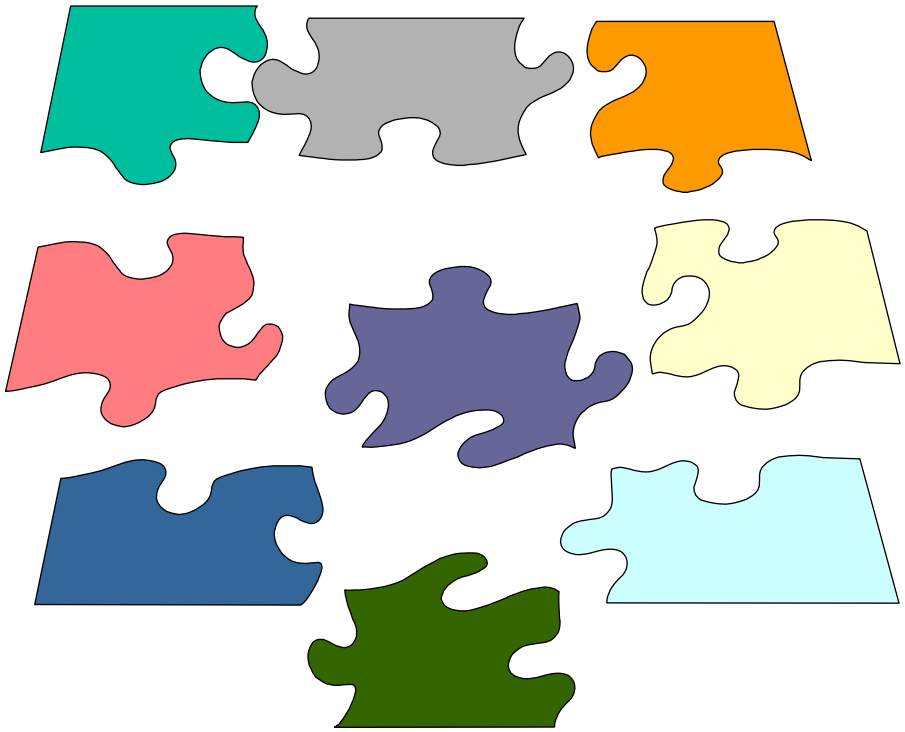
- Apply "Gang of Four" (GoF) patterns to guide the development of a framework of extensible classes.



en.wikipedia.org/wiki/Design_Patterns has information on the "Gang of Four" (GoF) book.

OO Design of Expression Tree Processing App

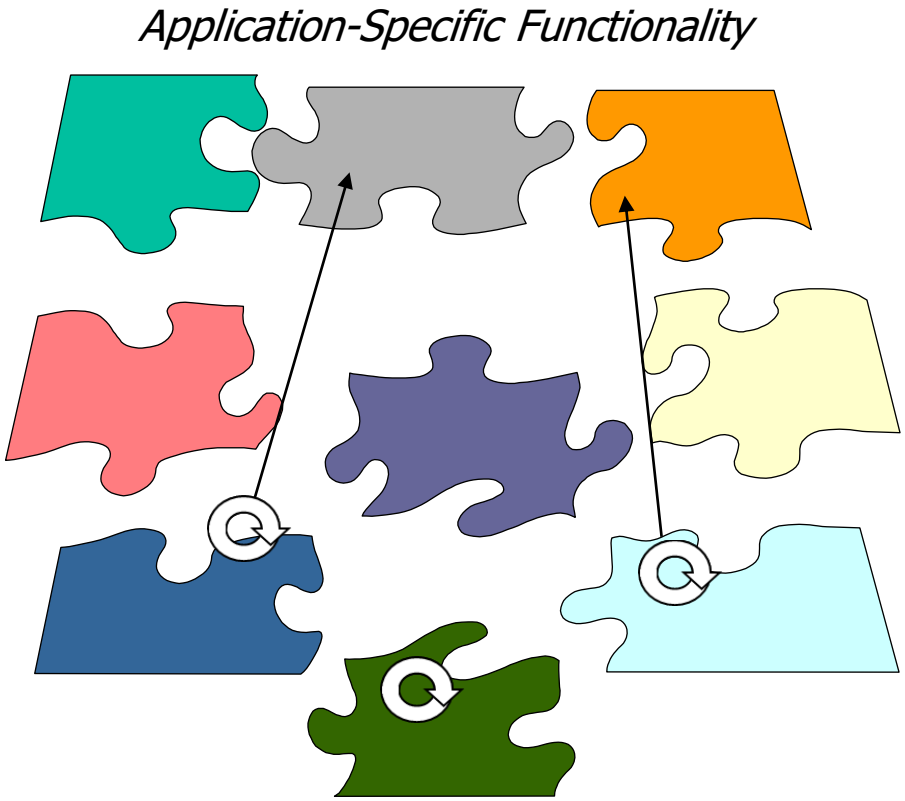
- Apply "Gang of Four" (GoF) patterns to guide the development of a framework of extensible classes.
  - A *framework* is an integrated set of software components that collaborate to provide a reusable architecture for a family of related applications.



See www.dre.vanderbilt.edu/~schmidt/frameworks.html

OO Design of Expression Tree Processing App

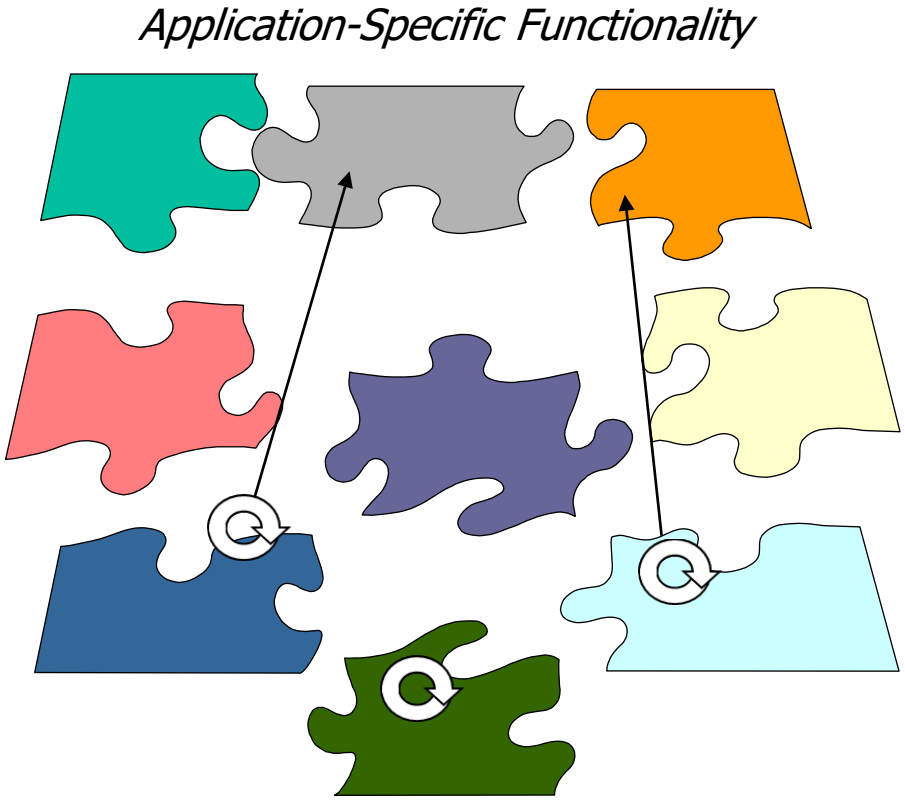
- Apply "Gang of Four" (GoF) patterns to guide the development of a framework of extensible classes.
  - A *framework* is an integrated set of software components that collaborate to provide a reusable architecture for a family of related applications.
  - Frameworks exhibit three characteristics that differentiate them from other forms of systematic reuse.



www.dre.vanderbilt.edu/~schmidt/reuse-lessons.html has info on systematic reuse.

OO Design of Expression Tree Processing App

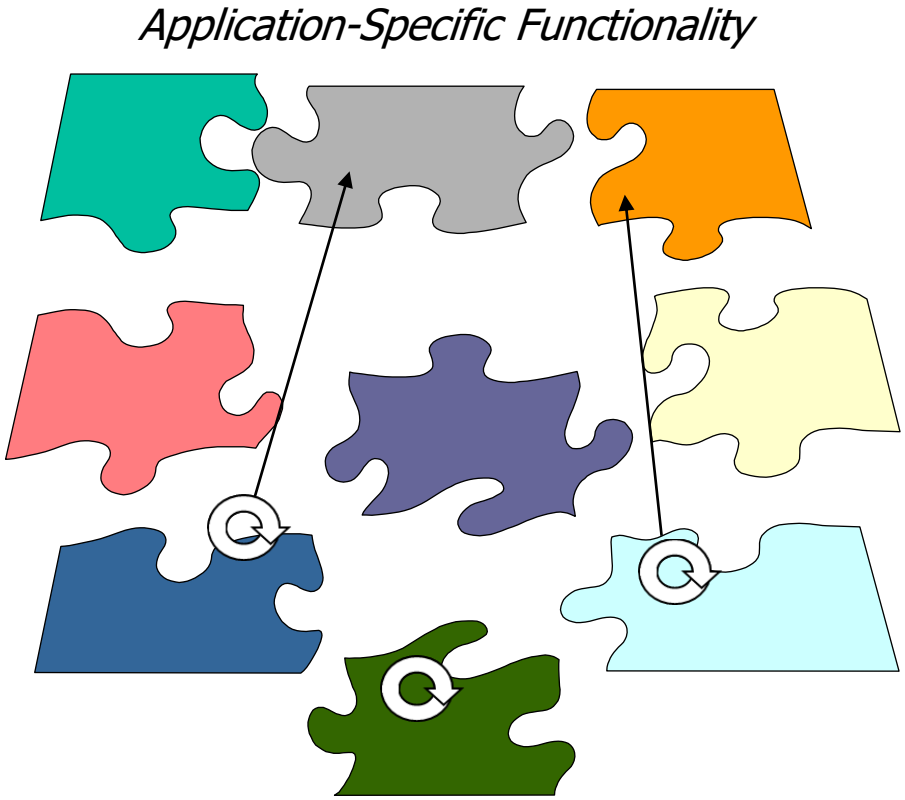
- Apply "Gang of Four" (GoF) patterns to guide the development of a framework of extensible classes.
  - A *framework* is an integrated set of software components that collaborate to provide a reusable architecture for a family of related applications.
  - Frameworks exhibit three characteristics that differentiate them from other forms of systematic reuse.
    1. Inversion of control (IoC)



See en.wikipedia.org/wiki/Inversion_of_control

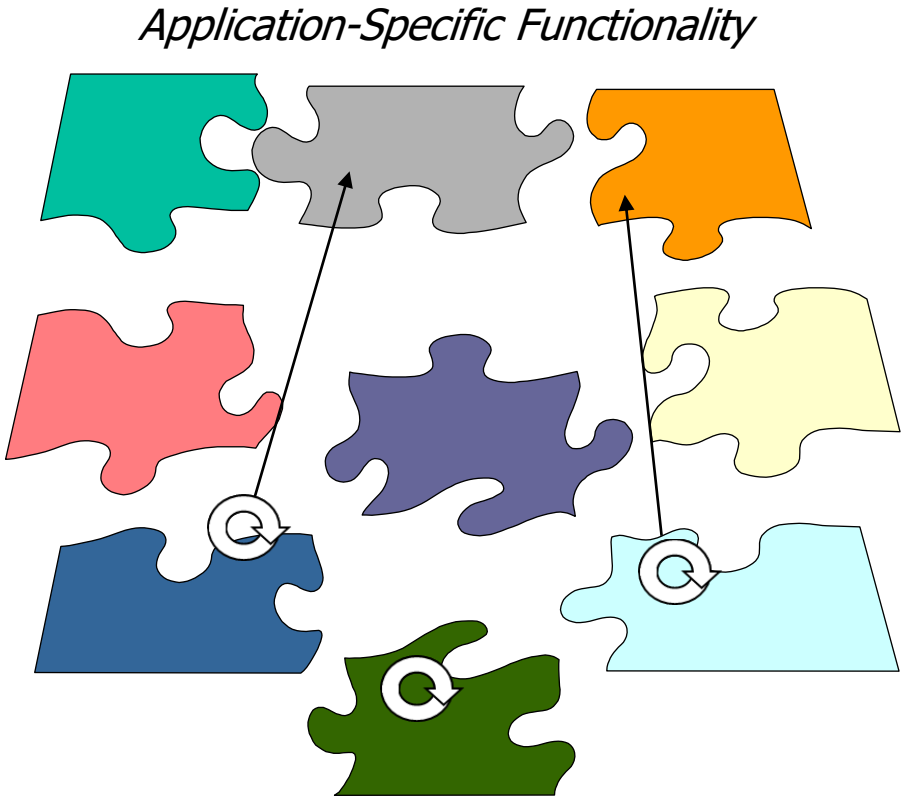
OO Design of Expression Tree Processing App

- Apply "Gang of Four" (GoF) patterns to guide the development of a framework of extensible classes.
  - A *framework* is an integrated set of software components that collaborate to provide a reusable architecture for a family of related applications.
  - Frameworks exhibit three characteristics that differentiate them from other forms of systematic reuse.
    1. Inversion of control (IoC)
       - The framework controls the main execution thread



OO Design of Expression Tree Processing App

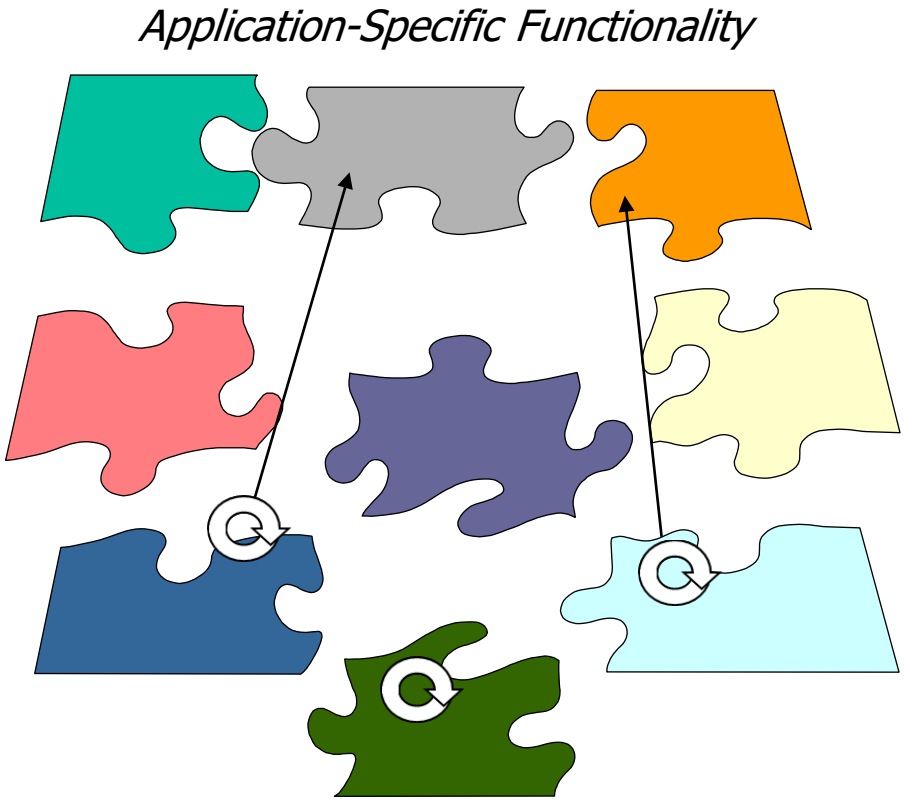
- Apply "Gang of Four" (GoF) patterns to guide the development of a framework of extensible classes.
  - A *framework* is an integrated set of software components that collaborate to provide a reusable architecture for a family of related applications.
  - Frameworks exhibit three characteristics that differentiate them from other forms of systematic reuse.
    1. Inversion of control (IoC)
       - The framework controls the main execution thread
       - Decides how/when to run app code via callbacks



See en.wikipedia.org/wiki/Callback_(computer_programming)

OO Design of Expression Tree Processing App

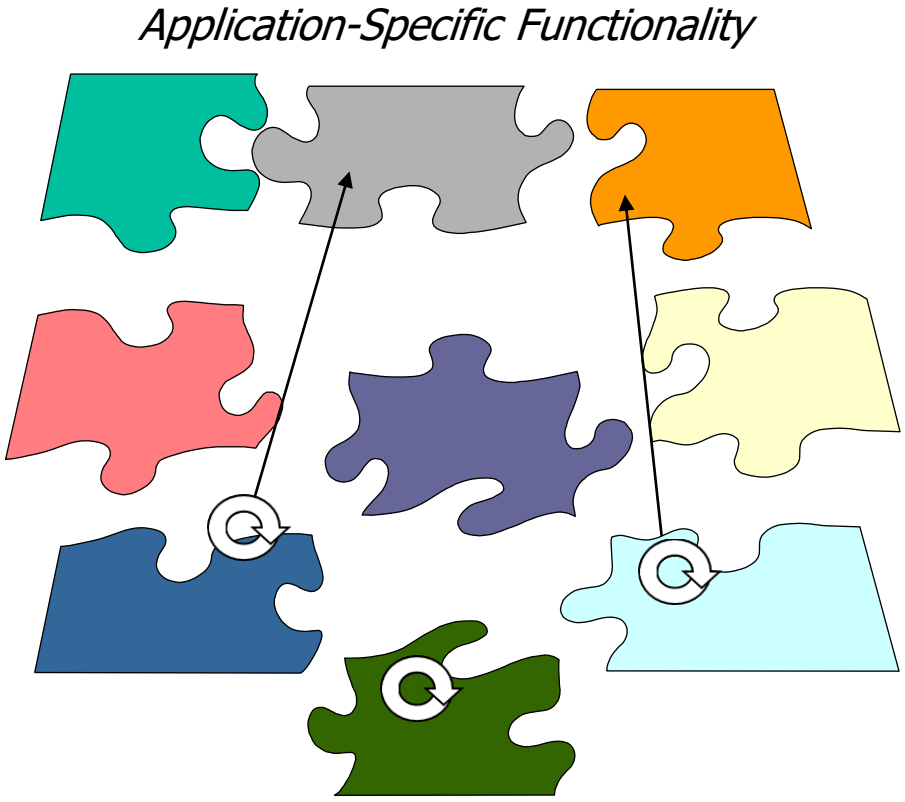
- Apply "Gang of Four" (GoF) patterns to guide the development of a framework of extensible classes.
  - A *framework* is an integrated set of software components that collaborate to provide a reusable architecture for a family of related applications.
  - Frameworks exhibit three characteristics that differentiate them from other forms of systematic reuse.
    1. Inversion of control (IoC)
       - The framework controls the main execution thread
       - Decides how/when to run app code via callbacks
         - e.g., an Android looper dispatches a handler, which then dispatches a runnable



See blog.mindorks.com/android-core-looper-handler-and-handlerthread-bd54d69fe91a

OO Design of Expression Tree Processing App

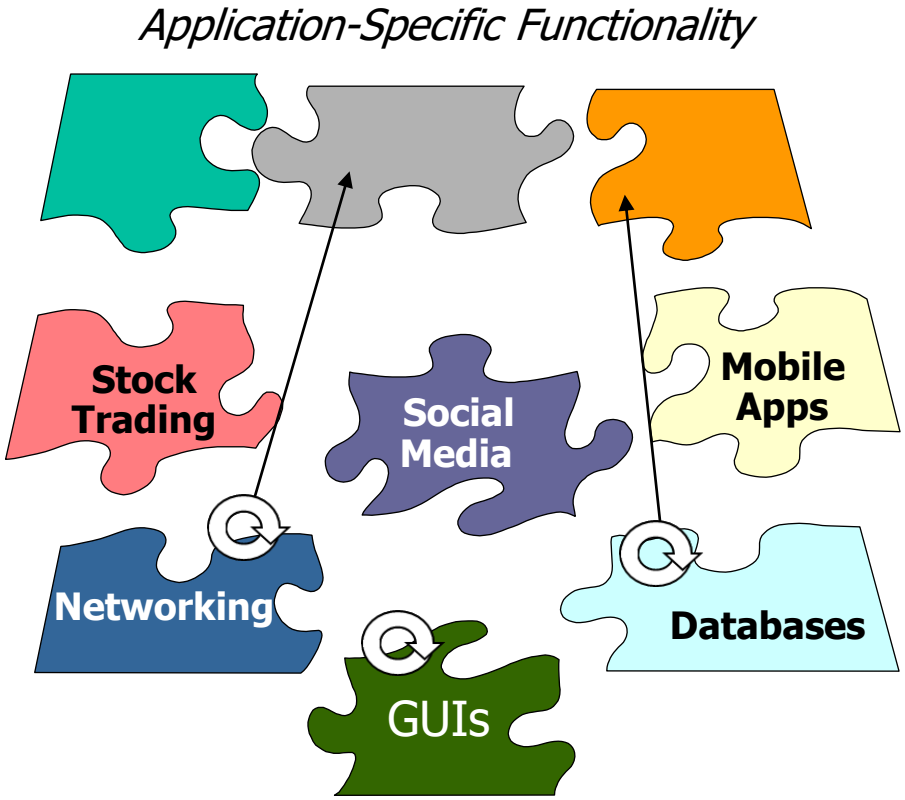
- Apply "Gang of Four" (GoF) patterns to guide the development of a framework of extensible classes.
  - A *framework* is an integrated set of software components that collaborate to provide a reusable architecture for a family of related applications.
  - Frameworks exhibit three characteristics that differentiate them from other forms of systematic reuse.
    1. Inversion of control (IoC)
       - The framework controls the main execution thread
       - Decides how/when to run app code via callbacks
       - IoC is often called "The Hollywood Principle"



See www.dre.vanderbilt.edu/~schmidt/Coursera/articles/hollywood-principle.txt

OO Design of Expression Tree Processing App

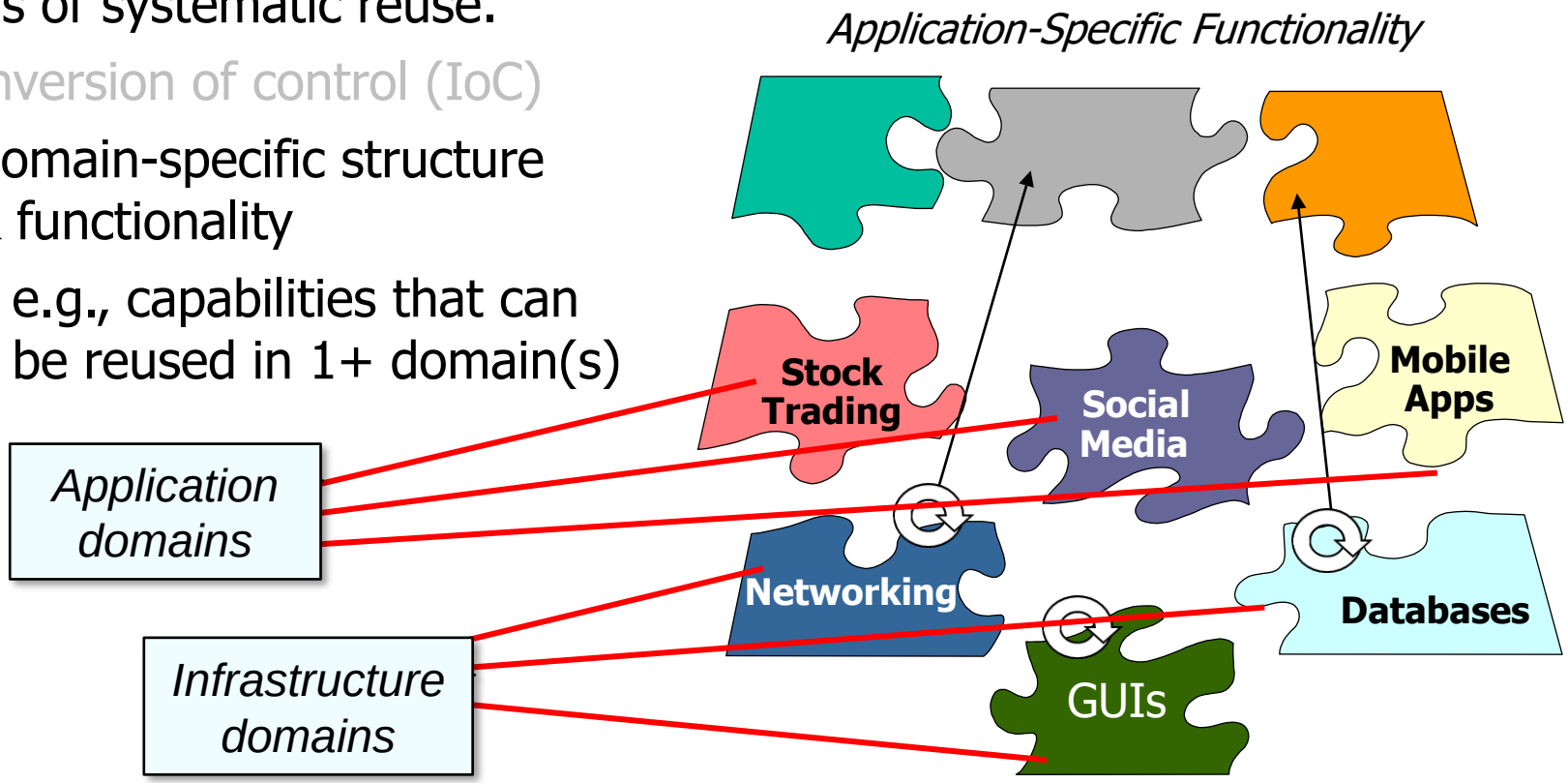
- Apply "Gang of Four" (GoF) patterns to guide the development of a framework of extensible classes.
  - A *framework* is an integrated set of software components that collaborate to provide a reusable architecture for a family of related applications.
  - Frameworks exhibit three characteristics that differentiate them from other forms of systematic reuse.
    1. Inversion of control (IoC)
    2. Domain-specific structure & functionality



See en.wikipedia.org/wiki/Domain-driven_design

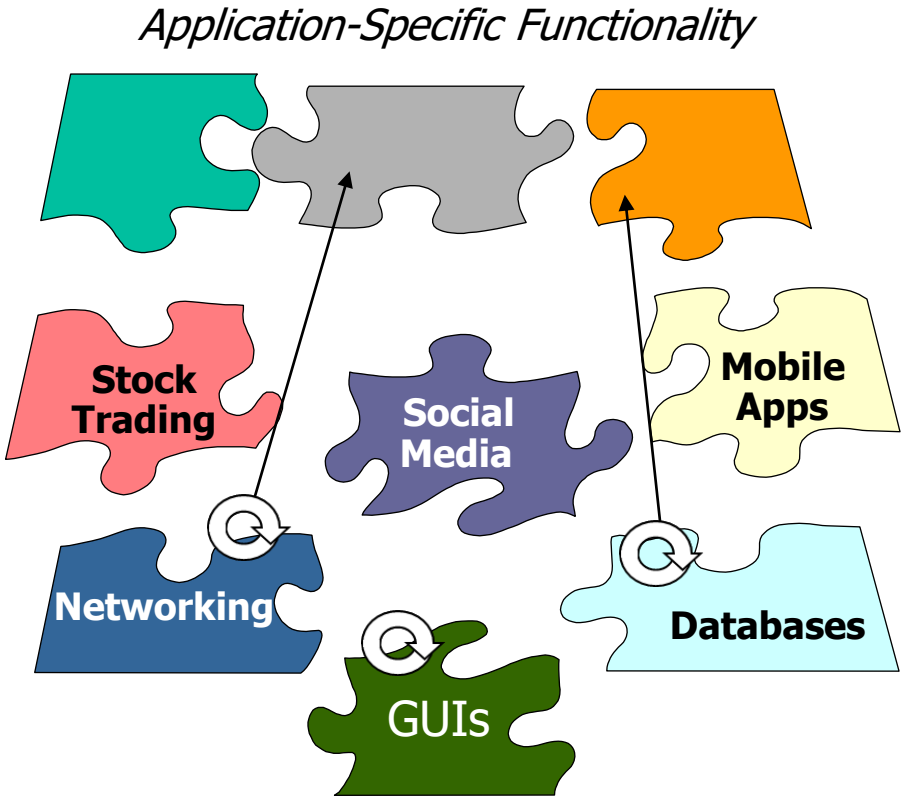
OO Design of Expression Tree Processing App

- Apply "Gang of Four" (GoF) patterns to guide the development of a framework of extensible classes.
  - A *framework* is an integrated set of software components that collaborate to provide a reusable architecture for a family of related applications.
  - Frameworks exhibit three characteristics that differentiate them from other forms of systematic reuse.
    1. Inversion of control (IoC)
    2. Domain-specific structure & functionality
       - e.g., capabilities that can be reused in 1+ domain(s)



OO Design of Expression Tree Processing App

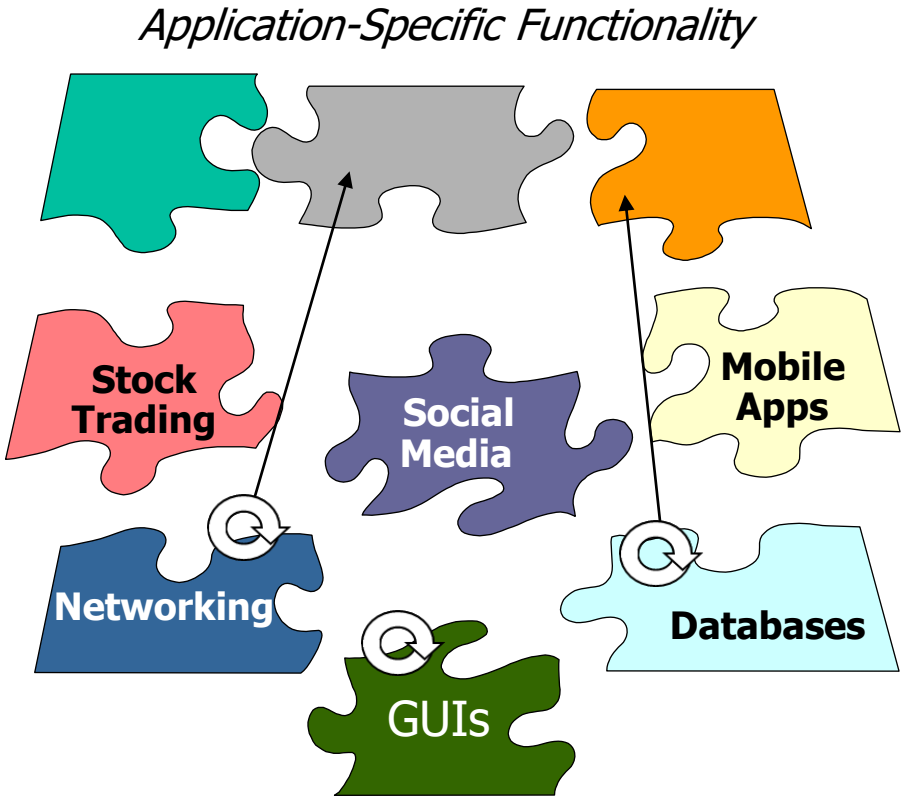
- Apply "Gang of Four" (GoF) patterns to guide the development of a framework of extensible classes.
  - A *framework* is an integrated set of software components that collaborate to provide a reusable architecture for a family of related applications.
  - Frameworks exhibit three characteristics that differentiate them from other forms of systematic reuse.
    1. Inversion of control (IoC)
    2. Domain-specific structure & functionality
       - e.g., capabilities that can be reused in 1+ domain(s)



Application-specific functionality can systematically reuse framework components.

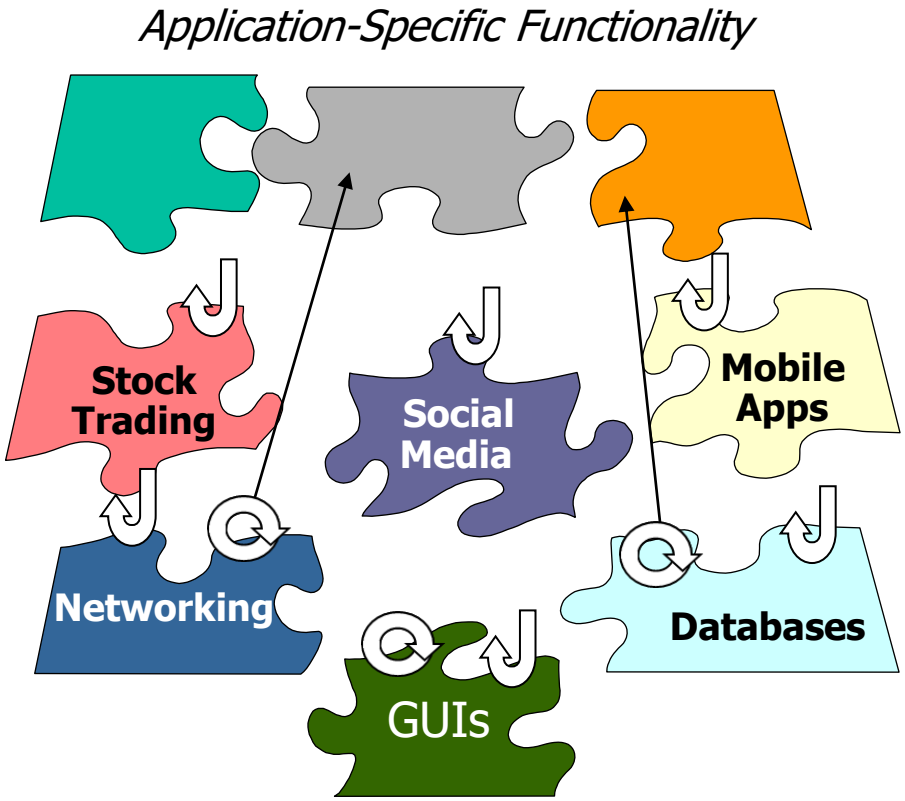
OO Design of Expression Tree Processing App

- Apply "Gang of Four" (GoF) patterns to guide the development of a framework of extensible classes.
  - A *framework* is an integrated set of software components that collaborate to provide a reusable architecture for a family of related applications.
  - Frameworks exhibit three characteristics that differentiate them from other forms of systematic reuse.
    1. Inversion of control (IoC)
    2. Domain-specific structure & functionality
    3. Semi-complete applications



OO Design of Expression Tree Processing App

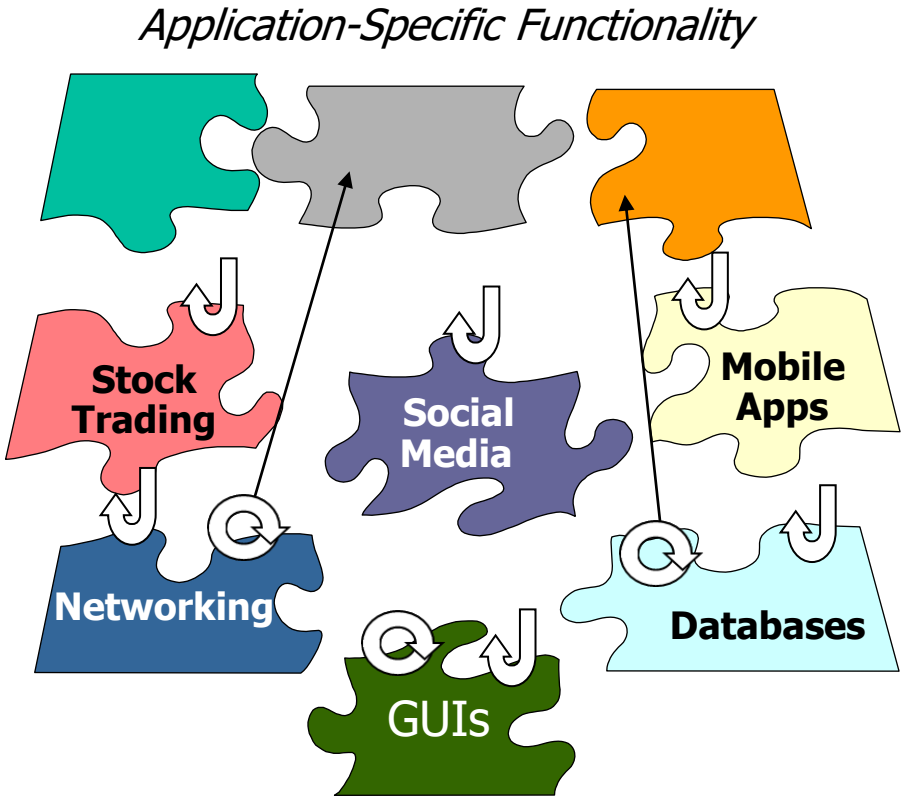
- Apply "Gang of Four" (GoF) patterns to guide the development of a framework of extensible classes.
  - A *framework* is an integrated set of software components that collaborate to provide a reusable architecture for a family of related applications.
  - Frameworks exhibit three characteristics that differentiate them from other forms of systematic reuse.
    1. Inversion of control (IoC)
    2. Domain-specific structure & functionality
    3. Semi-complete applications
       - *Hook methods* plug app logic into the framework



See codebetter.com/davelaribee/2008/06/16/hook-methods

OO Design of Expression Tree Processing App

- Apply "Gang of Four" (GoF) patterns to guide the development of a framework of extensible classes.
  - A *framework* is an integrated set of software components that collaborate to provide a reusable architecture for a family of related applications.
  - Frameworks exhibit three characteristics that differentiate them from other forms of systematic reuse.
    1. Inversion of control (IoC)
    2. Domain-specific structure & functionality
    3. Semi-complete applications
       - *Hook methods* plug app logic into the framework
       - Mediate interactions among *common* abstract & *variant* concrete classes/interfaces



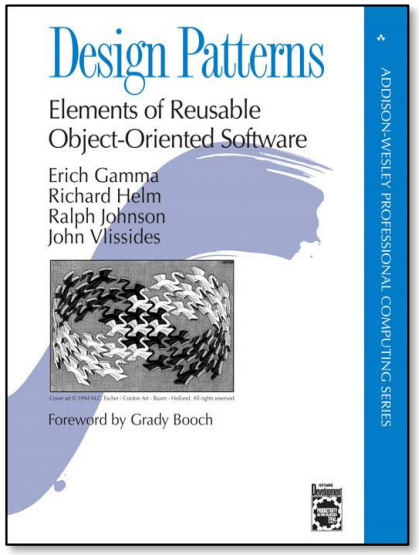
e.g., Java Runnable is an abstract interface providing basis for concrete variants.

OO Design of Expression Tree Processing App

- Integrate pattern-oriented language & library features with frameworks.
  - Both an app-specific framework…

```
Expression_Tree tree = ...;
Visitor print_visitor = ...;

for (auto iter = tree.begin(order);
     iter != tree.end(order);
     ++iter)
   (*iter).accept(print_visitor);
```

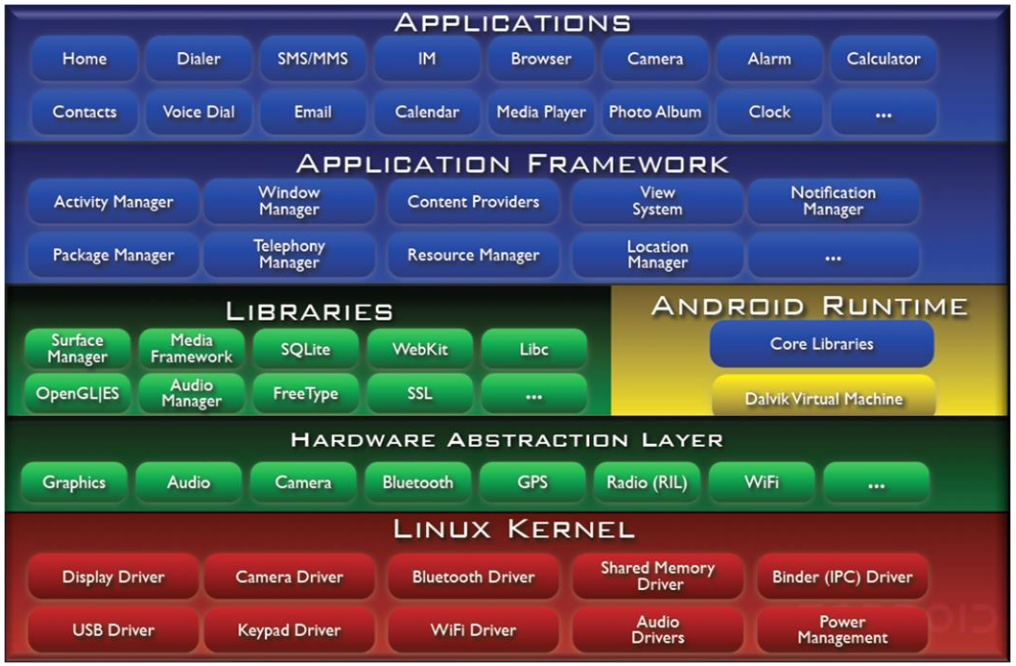


***Factory Method*, *Bridge*, *Composite*, *Iterator*, *Strategy*, & *Visitor* patterns**

This app-specific framework exhibits high pattern density!

OO Design of Expression Tree Processing App

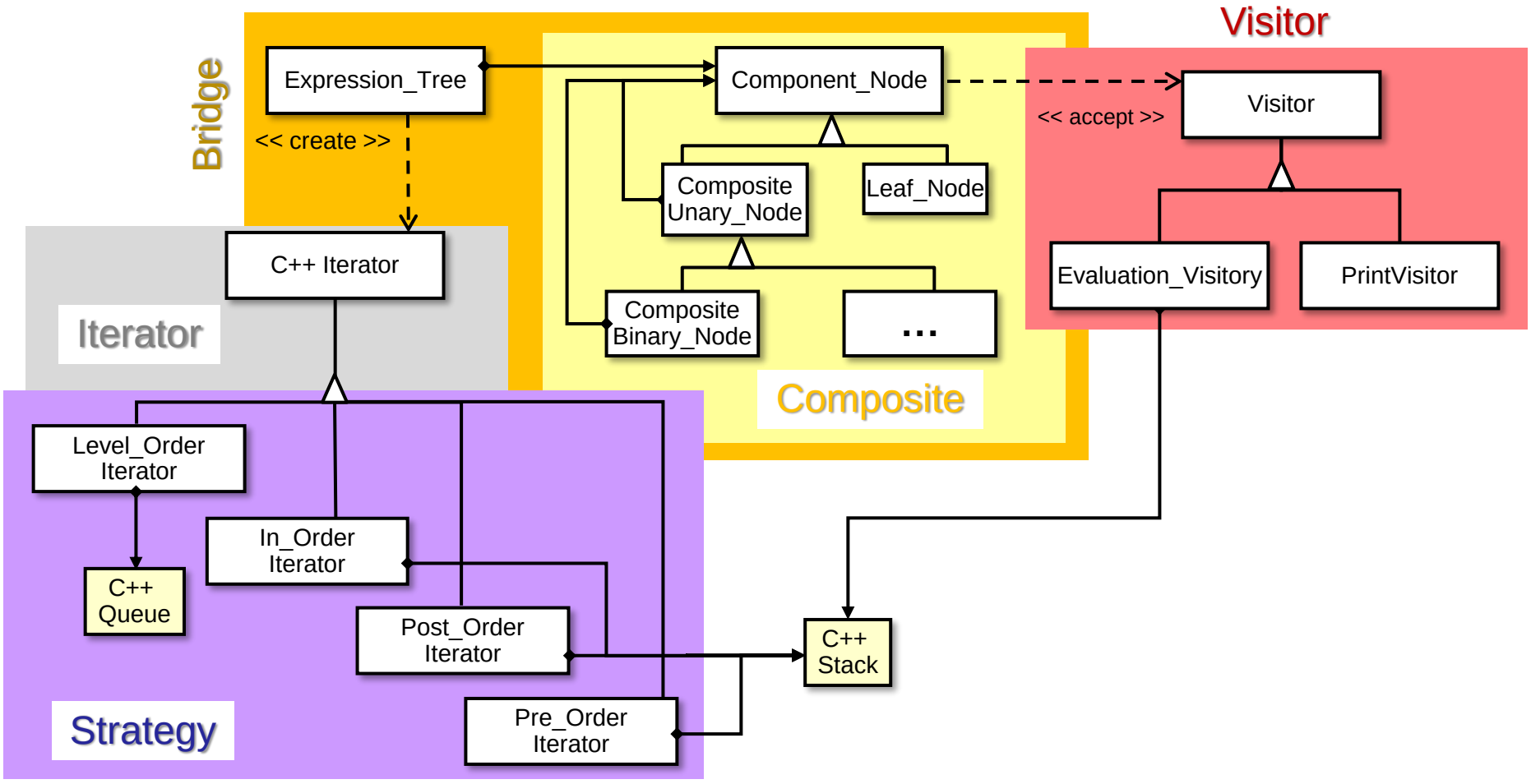
- Integrate pattern-oriented language & library features with frameworks.
  - Both an app-specific framework… & off-the-shelf frameworks…



See developer.android.com & en.wikipedia.org/wiki/Standard_Template_Library

OO Design of Expression Tree Processing App

- Complexity resides in (stable) structure & APIs, rather than (variable) algorithms.



See www.drdobbs.com/windows/software-complexity-bringing-order-to-ch/199901062