**Overview and Related Terms & Concepts**Design Pattern  
Architectural Pattern  
Programming Idiom  
…  
**The Gang Of Four – GOF**Singleton  
Abstract Factory  
Composite  
Proxy  
...

**Overview and Related  
Terms & Concepts**

**Design Pattern – What is it?**

- A general reusable solution to a commonly occurring problem in

software design

- A description/ template for how to solve a problem that can be

used in many different situations

- Not a finished design that can be transformed directly into code

- Programming language independent

- OO design patterns typically show relationships and interactions

between classes/ objects, without specifying the final application

classes/ objects that are involved

**Design Pattern – History**

- Christopher Alexander (1936, Austria)

\_ An architect noted for his theories about design, and for more than 200 building projects

in California, Japan, Mexico and around the world

\_ Documented and published about his experience in solving design issues as they related

to buildings and towns: it occurred to Alexander that certain design constructs, when

used time and time again, lead to the desired effect

- Nearly two decades ago, software professionals began to

incorporate Alexander's principles into the creation of early design

pattern documentation as a guide to novice developers

- The Gang Of Four (GOF, 1995)

\_ Erich Gamma, Richard Helm, Ralph Johnson, John Vlissides

\_ Published the book “Design Patterns: Elements of Reusable ObjectOriented Software” which introduces 23 design patterns

- Since that time, many more patterns have been documented

and cataloged. However, these 23 GOF ones are probably the

best known and certainly the most popular

**Design Pattern – Structure**

|  |  |
| --- | --- |
| **Term** | **Description** |
| **Pattern Name** | Describes the essence of the pattern in a short, but expressive, name |
| **Intent** | Describes what the pattern does |
| Also Known As | List any synonyms for the pattern |
| **Motivation** | Provides an example of a problem and how the pattern solves that problem |
| **Applicability** | Lists the situations where the pattern is applicable |
| **Structure** | Set of diagrams of the classes and objects that depict the pattern |
| **Participants** | Describes the classes and objects that participate in the design pattern and their responsibilities |
| Collaborations | Describes how the participants collaborate to carry out their responsibilities |
| **Consequences** | Describes the forces that exist with the pattern and the benefits, trade-offs, and the variable that is isolated by the pattern |

**Algorithms & Anti–Patterns**

- Algorithms

\_ NOT thought of as design patterns

\_ Solve computational problems rather than design problems

- Anti–Patterns

\_ Design patterns that appear obvious but are ineffective/ far from optimal in practice

\_ Comprise the study/ specific repeated practices that appear initially to be beneficial, but ultimately result in bad consequences that outweigh the hoped-for advantages

**Benefit of using Patterns**

- They have been proven

\_ Provides a way to solve software development issues using a proven solution

\_ Facilitates the development of highly cohesive modules with minimal coupling

\_ Isolates the variability that may exist in the system requirements

\_ Making the overall system easier to understand and maintain

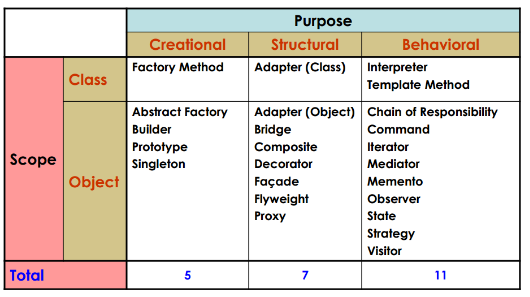
- They are expressive

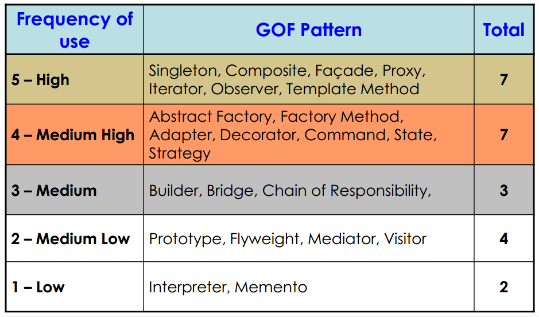
\_ Make communication between designers and developers more efficient

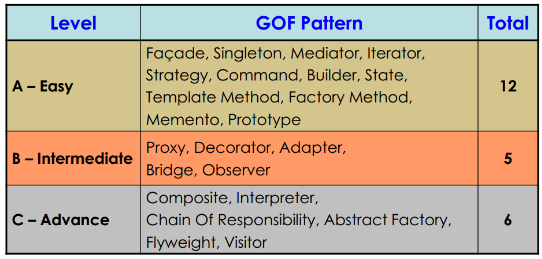
\_ Software professionals can immediately picture the high-level design in their heads when they refer the name of the pattern used to solve a particular issue when discussing system design

**The Gang Of Four – GOF**

**GOF – Catalogues**







**Design Pattern – Golden Rules**Client should always call the abstraction/ interface  
and not the exact implementation  
Future changes should not impact the existing system  
Change always what is changing  
Have loose coupling  
Inheritance (Very coupled)  
Composition  
Aggregation  
Association  
Dependency  
Realization (Least couple)