Investigation of Software Design Patterns
Dustin Waldron
Department of Computer Science and Engineering, Taylor University, Upland, IN, 46989

What Are Design Patterns?
In software engineering, a design pattern is a general repeatable solution to a commonly occurring problem in software design. A design pattern isn’t a finished design that can be transformed directly into code. It is a description or template for how to solve a problem that can be used in many different situations.

3 types of Design Patterns

- **Creational Design Patterns** pertain to class instantiation. These patterns can be further classified into class-creation patterns, which use inheritance effectively in the instantiation process, and object-creation patterns, which use delegation of various classes during instantiation of a class.
- **Structural Design Patterns** are a classification of design patterns that relate to the composition of a class. Structural object-patterns define various practices to compose objects to gain new functionality.
- **Behavioral Design Patterns** are design patterns that identify common communication patterns between objects. Behavioral Design Patterns increase flexibility in carrying out communication among class objects.

Why Design Patterns are Important
Design Patterns can be applied to numerous types of problems in software engineering. Design Patterns can decrease time required in the software development process. Reusing Design Patterns can prevent subtle issues that may cause major problems later in development. Design Patterns also improve code readability, as well as provide a language of communication for software engineers.

Why Design Patterns are Ignored
Some software engineers believe that Design Patterns target the wrong problem. The need for patterns is a result of using computer programming techniques with insufficient abstraction ability. With ideal factoring, a concept should not be copied, but referenced. For instance, Peter Norvig has demonstrated that 16 design patterns can be simplified or eliminated in the Lisp or Dylan programming languages.

Furthermore, some software engineers believe that Design Patterns lack formal foundations and are ad hoc. Other critics argue that Design Patterns do not differ significantly from other forms of abstraction and implementing new terminology in the field of programming is unnecessary, such as the Model-View-Controller paradigm.

Observer
The Observer Design Pattern is a pattern used to define one to many dependencies so that when one independent object changes the other dependent objects automatically change. Multiple observer objects register with a subject for change notifications. The independent object is called the subject and the dependent objects are called observers. When the state of the subject object changes, the subject object sends notifications to observer objects. The observer objects then perform necessary actions based on the notification sent. The subject object provides an interface for observers to register. The subject knows subscribed observer objects. The Observer Design Pattern is used heavily in views of Model-View-Controller applications.

Singleton
The Singleton Design Pattern is a design pattern used to ensure a class has only one instance. A Singleton object is encapsulated by “just-in-time initialization.” There is need for one and only one Singleton object for lazy initialization and global access to object data and functionality. A Singleton is a set containing one element. Singleton classes contain static attributes and accessor methods. Implementing the Singleton Design Pattern is better than global variables because the number of instances of a particular object are absolute, and definitely known in the Singleton object. The Singleton Design Pattern is necessary for an object to share data with other objects in a program.

Decorator
The Decorator Design Pattern is a pattern used to extend or modify the behavior of an instance of a class at runtime. A Decorator class changes the “skin” of an object. Unlike inheritance, which extends the abilities of a class, only a single instance of a class can have it’s behavior modified leaving other instances of the same class unmodified. The use of a Decorator Class removes the necessity of subclasses for a particular class. Typically, the Decorator Design Pattern utilizes a wrapper class around the object to extend behavior. The Decorator Design Pattern is capable of implementing recursive composition of classes.

MVVM
The Model-View-ViewModel Design Pattern is an architectural design pattern. There are Three components to the MVVM Design Pattern:
- **Model** - represents a set of classes that describe the business logic and data of an application
- **View** - represents the User Interface components of an application that displays data.
- **ViewModel** - responsible for exposing methods, commands, and other properties that help to maintain state of the view, manipulate models, and trigger events in the view.

The MVVM Design Pattern supports two-way data binding between the View and ViewModel of an application. MVVM is effective for separating data, behavior, and presentation of an application. Separation of data, logic, and presentation provides maintainable and extensible code, manageable development process, and easy testing with unit tests in layers of an application.

Why Design Patterns are Ignored

Some software engineers believe that Design Patterns lack formal foundations and are ad hoc. Other critics argue that Design Patterns do not differ significantly from other forms of abstraction and implementing new terminology in the field of programming is unnecessary, such as the Model-View-Controller paradigm.

Factory Method
The Factory Method Design Pattern is a design pattern used to instantiate an object from a set of classes based on a set logic rules. Based on arguments passed, the factory method object creates the appropriate subclass. The Factory Method can be seen as a “virtual” constructor. A factory method is preferable over using the operator because of order of invoked class constructors and memory space allocated during instantiation. Unlike a normal constructor, a Factory Method object can be more descriptive and return a subclass. Factory Method classes are helpful when the inheritance hierarchy exercises polymorphism in the creation of objects.

Singleton
The Singleton Design Pattern is a design pattern used to ensure a class has only one instance and provide a global joint to access it. A Singleton object is encapsulated by “just-in-time initialization.” There is need for one and only one Singleton object for lazy initialization and global access to object data and functionality. A Singleton is a set containing one element. Singleton classes contain static attributes and accessor methods. Implementing the Singleton Design Pattern is better than global variables because the number of instances of a particular object are absolute, and definitely known in the Singleton object. The Singleton Design Pattern is necessary for an object to share data with other objects in a program.

Literature cited

Acknowledgments
Dr. Tom Nuttall, Taylor University
Mr. David Colgan, Alumnus, Taylor University
Mr. Andrew Frenz, Alumnus, Taylor University
Mr. Campbell Prince, Wycliffe Bible Translation Software Studio Team
Department of Computer Science and Engineering Taylor University Center of Mission Computing