Aroofa Mohammad

CS 4376.001 - Object-Oriented Design

Professor Mark Paulk

8/30/2022

The Advantage of Object-Oriented Paradigm vs “Traditional” Paradigm

There are many differences between the object-oriented paradigms and the “traditional” design paradigms. The object-oriented paradigm is based on objects, which are usually instances of classes allowing objects to have both data and methods for modularity and reusability (*Ooad - Object Oriented Paradigm*). In contrast, traditional programming paradigms are based on procedural programming, which uses procedure calls, where each function or subroutine is a set of computational steps to follow (*Differences between procedural and Object Oriented Programming*). A noticeable difference is that object-oriented programming (OOP) uses a bottom-up approach, while traditional design paradigms use a top-down approach.

Object-oriented analysis (OOA) is the first step in OOP. In OOA, the system can interact with real-world objects. OOA determines the needs in creating software specifications in the context of an object model. The key steps of OOA are to identify objects, classify those objects using model diagrams, define internals and behavior, and document how objects interact. The primary distinction between object-oriented analysis and other types of analysis is the requirements and organization around objects, which contain data and function. Furthermore, the analysis model produced by OOA is turned into a design model by object-oriented design (OOD). This serves as a blueprint for developing programs. The analysis model turns into designing classes, constraints, and interfaces. OOD creates a design from the solved domain model. *(Upadhyay)*

The “traditional” paradigm focuses on how to solve the problem by breaking the problem into smaller sub-problems called procedures, functions, or methods. There are no clear rules for determining whether the functions are too basic, showing that decompositions are relatively arbitrary. Unlike object-oriented paradigms, “traditional” paradigms are not able to interact with the real world. The “Traditional” paradigm uses the concept called functional decomposition, which is the process of breaking complicated problems or operations into smaller more manageable components. This concept follows a top-down approach which is normally depicted as a hierarchy/tree. Meaning that the programmer starts at the root then goes down to the subsystem’s smallest component until the big picture is formed (Jurison).

Object orientation has four key concepts inheritance, encapsulation, abstraction, and polymorphism. In the concept of inheritance, a class can take on the traits and operations of another class by developing child classes that are specialized versions of the parent class. Inheritance is important because it allows the programmer to reuse code for maintainability and efficiency. Encapsulation is a concept in which data and methods are packaged and concealed within an object; as a result, they are inaccessible from outside the object. An example of this is a class in a high-level programming language called Java or Python. Encapsulation makes sure the programmers’ code is secure and safe. Abstraction is the practice of hiding an object's technical details so that it may be utilized without being aware of how it operates. This allows programmers to create code that is easy to understand. An object's capacity to assume several forms is known as polymorphism. The most frequent instance of polymorphism in OOP is when an object from a child class is referenced using a parent class. Polymorphism makes it possible for code to be more flexible and adaptable. Before the creation of the first object-oriented language, all of these ideas were there but unprotected. To finish them, careful and deliberate usage of structures and pointers was necessary. With the inclusion of OOP, code can be more efficient, user-friendly, and secure (Dorman)

.           As seen object-oriented paradigms differ from “traditional” paradigms in multiple ways. “Traditional” paradigms are created using procedural programming while object-oriented using OOP. Unlike the “Traditional” paradigm which uses simple processes, the object-oriented paradigm uses UML notations for creating diagrams. In addition, object-oriented has four main concepts, inheritance, encapsulation, abstraction, and polymorphism, which allow programmers to create safe and more efficient programs.

References:

“Ooad - Object Oriented Paradigm.” *Tutorials Point*, https://www.tutorialspoint.com/object\_oriented\_analysis\_design/ooad\_object\_oriented\_paradigm.htm.

Pankaj. “Differences between Procedural and Object Oriented Programming.” *GeeksforGeeks*, 28 June 2022, https://www.geeksforgeeks.org/differences-between-procedural-and-object-oriented-programming/#:~:text=In%20procedural%20programming%2C%20the%20program,follows%20a%20bottom%2Dup%20approach.

Upadhyay, Rajkumar. “Object Oriented Analysis and Design.” *GeeksforGeeks*, 5 July 2022, https://www.geeksforgeeks.org/object-oriented-analysis-and-design/.

“Traditional vs. Object-Oriented Approaches: Comparison between Traditional Approach and Object-Oriented Approach to Development in Software Engineering.” *Saylor Academy*, https://learn.saylor.org/mod/book/view.php?id=26802&chapterid=2497.

Dorman, Tate. “Functional Programming vs OOP: Which Paradigm to Use.” *Educative*, Educative, 17 June 2022, https://www.educative.io/blog/functional-programming-vs-oop.

Jurison, J. (1999). Software project management: The manager's view. *Communications of the Association for Information Systems*, 2, Article 17, 1-57.