WS1-4, Question 5, Object Oriented Design

Author: Thomas Brereton
ID number: 1708846

In this question we are asked to explain "the rationale for the use of object oriented programming" This question can be broken down into the following points.

- What problems does object oriented programming solve
- How does it solve these problems
- To what degree does it solve these problems

Object oriented programming (OOP) is an evolution in programming and attempts to provide an effective style of code. Object oriented design consists of the principles; Classes, Objects, Encapsulation and Abstraction of Information, Inheritance, and Polymorphism. These principles try to address known software engineering problems by enabling the key features[1]:

- Modular code
- Code that is easy to understand
- Code that is easy to correct and modify
- Increase in team productivity

Objects are essentially re-usable software components and are constructed from Classes. An analogy would be, a building is an object and its engineering drawings are the class. In other words, a class is how an object works. Following from this analogy, we could build many buildings with the same engineering drawings. This re usability of code is a key aspect of OOP, and removes redundant code often found in procedural programming. Thus, this solves the problem of repeated code by allowing users to create many objects of a class.

Another problem is programmers often need the code within a class in addition to some new code. Writing a completely new class just to include the new code would bring back the problem of repeated code. Therefore, OOP introduces the principle of inheritance and polymorphism. Inheritance allows programmers to create a new class which inherits (extends) code from the parent class. This means the child class only needs to include the different code and nothing which belongs to the parent class. Therefore, this solves the problem of repeated code when users want to create slightly different classes to existing.

Another factor is polymorphism which allows creating objects or lists of parent classes at compile time but passing in child classes at runtime. The benefits of this may not seem obvious at first but it further reduces code redundancy and increases productivity of software engineering team working together. For example, say a list contains objects of some parent class, p1, and some time later a programmer creates a child class of p1, called c1. It is still possible to use this list with the child class, c1. This is polymorphism and this principle only holds if c1 is a child of p1. This is because c1 contains all the attributes of p1 due to inheritance. The benefits of polymorphism allow software engineering teams to work well together when they adhere to the OOP principles.

A major problem in software engineering is the maintenance of legacy code. Previously, in procedural programming, correcting a small portion of code also involves looking through every line to ensure this small portion is not repeated. Inheritance avoids this tedious task by allowing programmers to modify code in the parent class, which is reflected in every child class. In other words, programmers go to one spot (class) to modify or correct code, and every child class of this parent class has it code automatically updated. Therefore, this solves the problem of maintaining legacy code by using the principles of inheritance.

When programmers are new to a large team, program, or code in general, it takes extensive periods of time before he or she understands it. To avoid programmers from spending too much time reading the code to better understand it OOP uses the principle of encapsulation and abstraction of data. It is the principle of hiding the complex mechanics behind classes and methods (functions). The programmer only needs to know what the classes and methods do, not how they do it. This reduces the time spent understanding code as the underlying mechanics are often not important, just what it does. Therefore, this solves the problem of productivity of dealing with new code and software engineering teams.

Whilst object oriented design addresses the issues outlined previously, it does not completely solve them, and there are still more. A few issues that OOP creates is a large overhead for writing a program. The size, speed, and cost increases for OOP[2]. The size because more code is generally needed for small programs. Speed is reduced as more code is generally needed, which relates to cost, as to keep the same productivity more programmers are needed. However, these costs are reduced significantly with today's increased computing power and intelligent integrated development environments. Moreover, the benefits of OOP often outweigh its costs when designing large or important programs.

References

[1] Harvey M. Deitel 1945. *Java : how to program : early objects / Paul Deitel (Deitel & Associates, Inc.), Harvey Deitel (Deitel & Associates, Inc.)* ID: 44BIR_ALEPH_DS003304362. 2015.

[2] *Object Oriented Programming.* `https://www.cs.drexel.edu/~introcs/Fa15/notes/06.1_OOP/Disadvantages.html?CurrentSlide=2`. (Accessed on 11/16/2016).