Merits (Advantages) of Object-Oriented Methodology:

1. Modularity:

Merit: By providing an application with a class with both data and methods, OOP fosters modularity. It is easier to understand, develop and maintain such programming.

2. Code Reusability:

Merit: To boost code reuse, object-oriented programming provides inheritance. This can lead to developing new classes from existing ones; thus creating an efficient and cost effective way of software development.

3. Flexibility and Extensibility:

Merit: With the aid of inheritance and polymorphism, a flexible and extensible system can be realized. Modifications in one part of the system do not affect other parts, while new features can be added without changing the program.

4. Abstraction:

Merit: To model classes on their essential attributes or behaviors, abstraction is used when designing complex systems. Therefore developers are able to focus on details that matter while ignoring those that are irrelevant.

5. Encapsulation:

Merit: Encapsulation allows for hiding internal details of an object while exposing a well-defined interface. Besides safety precautions since it prevents unauthorized access as well as manipulation of data.

6. Maintenance and Debugging:

Merit: Modularity eases debugging through narrowing down changes specific classes cause to code (Gogoi & Bhattacharyya 2003). In addition, OOP makes maintenance easy since the structure of code is modularized.

7. Natural Modeling:

Merit: The modelling in OOP is closer to entities in real life so it becomes easier for developers to create systems based on the actual structure and behavior of a problem domain.

8. Parallel Development:

Merit: The use of different programmers to work on various classes or objects allows for parallel development with component integration taking place later on.

Demerits (Disadvantages) of Object-Oriented Methodology:

1. Learning Curve:

Demerit: For beginners especially those coming from procedural background may find OOP difficult to learn. Such concepts as polymorphism or inheritance can be challenging for the beginners.

2. Performance Overhead:

Demerit: Some OOP features are associated with performance overhead. For instance, the use of dynamic binding and message passing has runtime overhead when compared to the more direct procedural code.

3. Increased Complexity:

Demerit: Improperly implemented or designed, OOP can increase complexity in some cases. Class hierarchy, which is poorly designed, can create a system that is complex and difficult to maintain.

4. Overhead of Abstraction:

Demerit: Though useful, abstraction has certain disadvantages. For example, it does not show anything important for programmers about optimizing or customizing a program.

6. Not Suitable for All Types of Problems:

Demerit: Object oriented programming may not always be suitable for solving all problems. Sometimes problems require more natural solutions present in procedural or functional programming paradigms.

7. Limited Support for Parallel Programming:

Demerit: Traditional OOP models may have limitations in dealing with certain aspects of parallel programming. In parallelization and distribution systems however actor based models are sometimes preferred over these newer paradigms.

8. Potential for Overuse of Inheritance:

Demerit: Too much inheritance leads to rigid class hierarchies that cannot change flexibly. Hence, multiple inheritances tend to create ambiguity and complexity which is often called “the diamond problem”.

Related posts:

1. Abstraction and encapsulation
2. Object Oriented Programming & Methodolog Viva Voce
3. How to install compiler for code blocks
4. Object Oriented Programming
5. Differences between Procedural and Object Oriented Programming
6. Features of Object Oriented Paradigm
7. Inheritance in Object Oriented Programming
8. Object Oriented Programming
9. Introduction to Object Oriented Thinking & Object Oriented Programming
10. Difference Between Object-Oriented Programming (OOP) and Procedural Programming
11. features of Object oriented paradigm
12. Concept of Objects: State, Behavior & Identity of an object
13. Access modifiers
14. Static members of a Class
15. Instances in OOP
16. Message Passing in OOP
17. Construction and destruction of Objects