International Journal of Science, Strategic Management and Technology
Volume **02** Issue **02** February-2026 | ISSN: **3108-1762 (Online)** | Impact Factor: 3.8
An International, Peer-Reviewed, Open Access Scholarly Journal Indexed in recognized academic databases

OPEN ACCESS

Current Trends of Object Oriented Programming Paradigm in Software Development

Dr. Rupali Pravinkumar Pawar
Assistant Professor,
Zeal Institute of Business Administration, Computer Application & Research, Pune

https://doi.org/10.55041/ijsmt.v2i2.134

Cite this Article: Pawar, R. P. (2026). Current Trends of Object Oriented Programming Paradigm in Software Development. International Journal of Science, Strategic Management and Technology, *Volume 10*(01). https://doi.org/10.55041/ijsmt.v2i2.134

ABSTRACT

Object oriented programming is the most popular programming paradigm in the current Information technology era. This paper focuses on in-depth Object-Oriented Programming paradigm, OOP modelling and design, features of OOP, benefits and challenges of OOP, and languages which follow OOP in recent times.

This paper analyzes some of the most important technological innovations in object-oriented software engineering in recent times. One of the most challenging jobs of software developers is to fit the development process in a specific architecture or framework or programming paradigms. This paper focus on advancements in Object technologies that include object cloning, class evolution, introspection and reflection, interfaces, global software development contexts, namespaces, query-able source codes, meta-model for generating design alternatives, magic methods, design pattern detection, auto-active functional verification.

This paper discusses current challenges and issues of implementing object-oriented paradigm in software development.

KEYWORDS: Objects, Class, Object Oriented Programming, Software Engineering, Software Development.

INTRODUCTION

Programming paradigms are many styles or ways of structuring a certain programming language or program. Some programming languages have been developed with features and a particular paradigm that make this type of programming easier than others. Functions or procedures are the main emphasis of procedure-oriented programming. Data was not the primary focus of these programming paradigms. Thus, there was a good chance that the issue would not be resolved effectively.

Data security was readily compromised and data was almost ignored in procedure-oriented programming. bject-oriented programming was able to overcome all of the disadvantages of procedural programming.

The two primary ideas of the OOP programming paradigm are "classes and objects." Data is the main focus, followed by approaches. A computer programming paradigm known as object-oriented programming (OOP) centers software design around data, or objects, as opposed to logic and functions. Put more emphasis on objects in this programming than on logic or data. An object-oriented program's structure also makes the approach advantageous for group projects and collaborative development.

Binding data and functions that use it together such that only those functions can access the data is the main objective of object-oriented programming (OOP). OOP also offers the advantages of efficiency, scalability, and code reuse.

OOP Model:

The main objective of object-oriented modelling and design is to learn how to apply object-oriented concepts to all the stages of the software development life cycle. Now a day, most of the programming languages follow the Object

Oriented Programming paradigm. Object-oriented modelling and design provide a way of thinking about problems using models organized around real-world

concepts. Object is the foundation of OOP, which combines both data and behaviour.

The OOP model visualizes the elements in a software application in terms of objects.

Purpose of Models:

1. Before construction, a physical object must be tested.
2. Communication with customers
3. To improve the visualization
4. To reduce complexity
5.

Types of Models:

There are 3 types of models in object-oriented modelling and design: Class Model, State Model, and Interaction Model. These are explained below.

1. Class Model:

The class model lists all the classes present in the system. The class model shows the details of the attributes and the behaviour associated with the objects. The class diagram is used to show the class model. The class diagram having three compartments, first is the class name followed by the attributes followed by the functions or the methods that are associated with the object of the class. The main goal in constructing a class model is to capture those concepts which are important for application from the real world.

2. State Model:

State model shows those aspects of objects concerned with time and the sequencing of operations – events that mark changes, states that define the context for events, and the organization of events and states. State diagram focuses on actions and events which become operations on objects in the class model. State diagram shows the state model of an application.

3. Interaction Model:

Interaction model is used to show the various interactions between objects. It describes how the objects collaborate to achieve the behaviour of the system as a whole. The following diagrams are used to show the interaction model:

- **Use Case Diagram**
- **Sequence Diagram**
- **Activity Diagram Programming Language:**

Programming language is a system of writing a computer program, it consists of syntax and semantics. It is a set of instructions to communicate with Computer. There are a number of ways to classify programming languages.

1. **Procedural Programming Language**

A procedural language follows a sequence of statements or instructions in order to achieve a desired output. Each procedure consists of a series of instructions or steps. E. g. C, C++, Pascal, etc. 2. Functional Programming Language Functional languages focus on the output of mathematical functions and evaluations, rather than execution of statements. Each function–a reusable module of code–performs a specific task and returns a result. E. g. Scala, Haskel, etc.

2. **Scripting Language**

Scripting languages are used to automate repetitive tasks, manage dynamic web content, or support processes. E. g. Ruby, Python, Node Js, etc.

3. **Logic Programming Language**

A logic programming language expresses a series of facts and rules to instruct the computer on how to make decisions. E. g. Prolog, Absys, Datalog, etc.

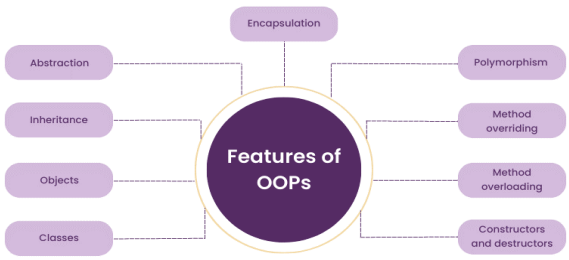
4. Object Oriented Programming Language

In OOP language, everything is designed around class and object format. Object is used to bind data and methods together. It is an instant of class data type. Objects can be reused in the same program or in another program. OOP is used to implement real world entities by using OOP concepts like inheritance, polymorphism, abstraction, etc.

E. g. C++, Java, Python, Ruby, etc.

Features of OOP:

The key features of Object-Oriented Programming (OOP) are encapsulation, inheritance, polymorphism, and abstraction, which promote code reusability, modularity, and maintainability



Classes & Objects

A class is a logical entity. It can also be defined as a blueprint of an object. It is a bunch of methods and

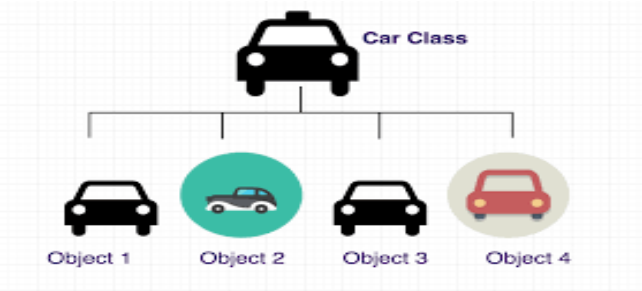
International Journal of Science, Strategic Management and Technology
Volume **02** Issue **02** February-2026 | ISSN: **3108-1762 (Online)** | Impact Factor: 3.8
An International, Peer-Reviewed, Open Access Scholarly Journal Indexed in recognized academic databases

variables. It is the basis of OOP. A class contains a skeleton of the object and does not take any space in the memory. Class determines how an object will behave and what the object will contain.

An object is a basic unit in object-oriented programming. An object

contains data and methods or functions that operate on that data. Objects take up space in memory.

Thus, instead of defining these similar data and functions in different objects, we define a blueprint of these objects which is a class.



contains data and methods or functions that operate on that data. Objects take up space in memory.

Thus, instead of defining these similar data and functions in different objects, we define a blueprint of these objects which is a class.

1. Abstraction

Abstraction is the process of hiding irrelevant information from the user. It is a technique that relies on the separation of interface and implementation.

Using abstraction in programming, we can hide the low-level details of the programming constructs that define the underlying logic, which in turn simplifies and streamlines the development process. By using abstraction in our application, the

2. Encapsulation

Encapsulation is the process using which data and the methods or functions operating on them are bundled together. By doing this, data is not easily accessible to the outside world. In OOP we achieve encapsulation by making data members private and having public functions to access these data members.

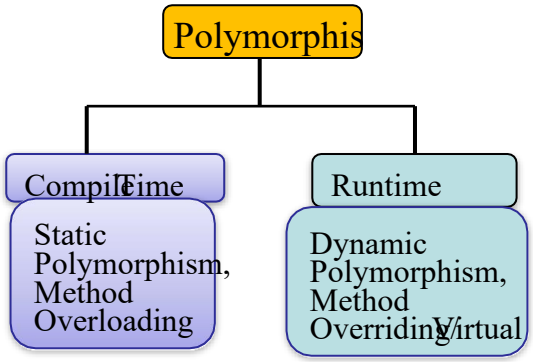


Fig. No.3

3. Inheritance

Using inheritance objects of one class can inherit or acquire the properties of the object of another class. Inheritance provides reusability of code.

As such we can design a new class by acquiring the properties and functionality of another class and in this process, we need not modify the functionality of the parent class. We only add new functionality to the class.

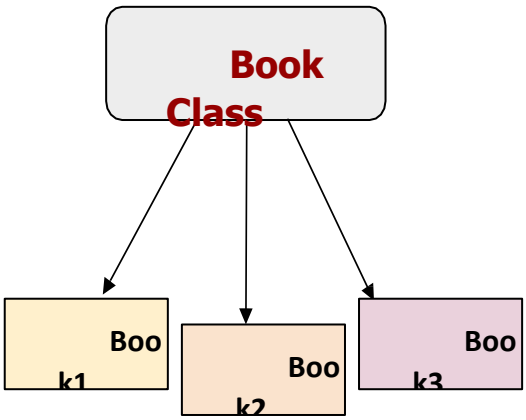


Fig. No.3

4. Polymorphism

Polymorphism means many forms. Polymorphism is an important feature of OOP and is usually implemented as operator overloading or function overloading. Operator overloading is a process in which an operator behaves differently in different situations. Similarly, in function overloading, the same function behaves differently in different situations. OOP supports dynamic binding in which a function call is resolved at runtime. Dynamic binding is implemented by using method overriding or virtual function. In Java Programming, Runtime polymorphism is achieved by using the concept of

method overriding.

5. Message Passing

In OOP, objects communicate with each other using messages. When objects communicate, information is passed back and forth between the objects. A message generally consists of the object name, method name and actual data that is to be sent to another object.

Languages which follow OOP:

1. **Complete/ Pure Object-Oriented Language:**
Pure or complete Object-Oriented Language are fully supported or have features which treat everything inside a program as objects. It doesn't support primitive data types (like int, char, float, bool, etc.). Programming languages are considered as pure when these languages treat everything as class and object format.
E.g. Simula (Simulation Language) is credited as being the first object-oriented programming language. But some programming language concepts are paired with OOP.

2. Partially Object-Oriented Language:

Programming languages support for procedure oriented as well as object-oriented programming. These are known as Partially object-oriented programming languages. Some OOP languages support the use of built-in primitive data types, so we can call these as partially OOP languages.

Benefits of OOP:

- OOP is faster and easier to execute
- OOP provides a clear structure for the programs.
- OOP helps to do programming on the basis of concept DRY ("Don't Repeat Yourself"), and it makes the code easier to maintain, modify and debug
- OOP makes it possible to create full reusable applications with less code and shorter development time
- Reduce the number of codes a programmer needs to write
- Reduce the cost of software development
- Improve the maintainability
- It provides the security

Current advancements in OOP:
The advancements in Object technologies that have been analyzed in this paper in following table,

Sr. No.	Advancement of OOP	Description
1.	Object Cloning	It is a process to copy one object to another. It creates exact copies of objects.
2.	Class Evolution	Execution of classes in OOP.
3.	Introspection And Reflection	Reflection is the ability of a program to manipulate as data something representing the state of the program during its own execution.
4.	Interfaces	It is a reference type in Java. used for implementation of multiple inheritance.
5.	Global Software Development Contexts	Global software development (GSD) contains different context setting dimensions, which are essential for effective teamwork and success of projects.
6.	Namespaces	Namespaces help you to more effectively manage your code base by compartmentalizing various libraries and classes according to context. [15]
7.	Query-Able Source Codes	It is an advancement in object-oriented software development processes where source code encodes a query-able and modifiable model of the applications and is utilizable by IDEs in these aspects. [16]
8.	Meta-Model for Generating Design Alternatives	A meta-model is an explicit model of the constructs and rules needed to build specific models within a domain

		of interest.
9.	Dunder or Magic Methods	Dunder or magic methods used in Python, the methods having two prefix and suffix underscores in the method name. Dunder here means "Double Under (Underscores)". These are commonly used for operator overloading.
10.	Design Pattern Detection	Detecting well-known design patterns in object-oriented program source code can help maintainers understand the design of a program.
11.	Auto-Active Functional Verification.	Auto-active Functional verifiers are OOP concepts that provide an intermediate level of computerization(automation) between fully automatic systems and the interactive users supply code with annotations as input while benefiting from a high level of automation in the back-end.[17]

Challenges and Issues of OOP:

Software development is a complex process that can often be unpredictable. For every project, the software development team faces new problems and solves that. Software developers are immune to software development challenges. Today the software development team develops software for new problems. They started developing software from scratch. Different libraries and frameworks are chosen for different parts of the system.

- One of the most challenging jobs of software developers is to fit the development process in a specific architecture or framework.
- Software development teams face new problems, so it is very complicated to fit the software development process in a specific framework or architecture.
- To implement the specific programming paradigm.

This paper focuses on current challenges of implementing object-oriented paradigms in software development.

1. Object-oriented languages create an implicit environment that they carry around with them.

2. The diamond problem

Inheritance is a very famous concept in OOP, where we can take properties of one class and transfer it to others. But the problem is mixing the properties of two different classes.

3. The hierarchy problem

We don't think of object-oriented programming without hierarchy. The most difficult task is to establish proper Actually, you're dealing with a bunch of similar properties.

4. The reference problem

We could use clusters of properties, and inherit, extend, or override properties as needed. By clustering all these things, that would be a bit messy, and we need an accurate representation of the problem at hand.

CONCLUSION

This research focused on basic concepts of Object-Oriented Programming. This paper starts from procedure oriented and object-oriented programming concepts, after those covers object-oriented modelling concepts. We have covered the OOP features such as Classes, Objects, Encapsulation, Polymorphism, Inheritance and Abstraction. By the use of that OOP features projects can be developed in an efficient and better way.

This paper covered benefits and challenges of implementation of Object-Oriented Programming in software development. The current trend of software development is to meld the process of software development as per the customer requirements. Feature of this research is to overcome the challenges and issues of OOP in software development.

REFERENCES

[1] Amit Verma, Navdeep Kaur Gill, "Image Processing and Watermark", International Journal of Computer Science Technology (JCST), Vol. 7, Issue 1, pp. 143-147, Jan - March 2016.

[2] Procedure Oriented Programming (POP) vs Object Oriented Programming (OOP). 2011.

[3] Rajesh Babu Nagineni, A Research on Object Oriented Programming and Its Concepts, International Journal of Advanced Trends in Computer Science and Engineering, 10(2), March -April 2021, 746 - 749.

[4] NAYER A, F F WU and K IMHOT, Object Oriented Programming for Flexible Software Example of a Load Flow Power Systems IEEE Transactions on 1990 5(3): Page 689-696.

[5] Seed, Graham M, An Introduction to Object-Oriented Programming in C++ with Applications in Computer Graphics, Second EdSpringer-Verlag, 2001.ISBN 1- 85233-450-9.

[6] Svenk, Goran, Object-Oriented Programming: Using C++ for Engineering and Technology Delmar, 2003. ISBN 0-7668-3894-3.

[7] kaur, l., kaur, n., ummat, a., kaur, j., & kaur, n. (2016). research paper on object-oriented software engineering. international journal of computer science and technology, 36-38.

[8] Kak, Avinash C. Programming with Objects, A Comparative Presentation of Object-Oriented Programming with C++ and Java, John Wiley, 2003. ISBN 0-471-26852-6.

[9] Yevick, David, A First Course in Computational Physics and Object-Oriented Programming, Cambridge University Press, 2005. ISBN 0-521-82778-7.

[10] Onu F. U., (2016) "Comparative Study of Structured and OOP Paradigms", International Journal of Advanced Research in Computer Science and Software Engineering vol. 6, No 8, pp 30-39.

[11] Kiczales G., John L., Anurag M., Maeda C., Cristina V., Jean-Marc L., and Irwin J., (1997), Aspect Oriented Programming, proceedings of the European Conference on Object-Oriented Programming (ECOOP), Finland. Springer-Verlag LNCS 1241.

[12] Aniche, M., Yoder, J., & Kon, F. (Accepted/In press). Current Challenges in Practical Object- Oriented Software Design. In 41st ACM/IEEE

[13] Schuster F., Tendyck T., Liebchen C., and Davi L., (2015), Counterfeit Object-oriented Programming: On the Difficulty of Preventing Code Reuse Attacks in C++ Applications, 2015 IEEE Symposium on Security and Privacy, pp 745- 762.

[14] Corral R., Morgado-Estevez A., Jimenez G., and Civit A., (2014), A game-based approach to the teaching of Object-oriented programming languages, Computer and Education, Vol 73., pp [15] Gilmore W. J. (2008), Advanced OOP Features. In: Beginning PHP and MySQL, Apress, pp 193-194.

[15] Biswajit S, and Debaprasad M, (2017), Analysis of Applications of Object Orientation to Software Engineering, Data Warehousing and Teaching Methodologies, International Journal of Computer Sciences and Engineering, Vol. 5(9), 244-248.

[16] Julian T, Carlo A. F, Martin N, and Nadia P, (2015), Auto Proof: Auto-active Functional Verification of Object-oriented Programs, Proceedings of the 21st International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS). Lecture Notes in Computer Science, 9035:566--580, Springer

[17] International Conference on Software Engineering: New Ideas and Emerging Results (NIER) IEEE

[18] Andrew P. Black, Object-oriented programming: Some history, and challenges for the next fifty years, Information and Computation, Volume 231, 2013, Pages 3-20, ISSN 0890-5401, https://doi.org/10.1016/j.ic.2013.08.002. (https://www.sciencedirect.com/science/article/pii/S0890540113000795)

[19] A. Snyder, The essence of objects: Common concepts and terminology Technical Report HPL- 91-50, Hewlett Packard Laboratories (1991).

[20] Skelly J. F., (1997), Application of Object Oriented Programming Techniques in Front End Computers, Paper submitted to: QCALEPCS ' 97 Beijing, China, pp 3-5.