Global Journal of Engineering, Design & Technology
ISSN: 2319-7293
OPEN ACCESS Freely available online
Opinion Article

The Implications of Object-Oriented Analysis and Design

Weifan Liu*

Department of Computer Science and Centre for Health Informatics, University of Manchester, Manchester, UK

DESCRIPTION

Object-oriented analysis and design (OOAD) is a technical method for assessing and planning an application, system, or company using object-oriented programming and visual modeling to direct stakeholder communication and product quality throughout the software development process.

In contemporary software engineering, OOAD is frequently carried out incrementally and iteratively. Analysis models and design models are the byproducts of OOAD processes. These are intended to be continuously improved upon and changed, with the help of important elements like risks and business value.

Object-oriented analysis

Any analytic activity in the software life cycle has the goal of modeling the functional needs of the system without regard to implementation limitations.

The primary distinction between object-oriented analysis and other types of analysis is how we structure requirements around objects, which include behaviors and states patterned after actual items that the system interacts with in the real world.

The two components, processes and data, are taken into account individually in other or traditional analysis approaches. For instance, Entity Relationship (ER) diagrams can be used to represent data, and flow charts or structure diagrams can be used to model behaviors.

Use cases and object models are typical OOA models. Use cases outline the situations for typical domain tasks that the system must carry out. Names, class relationships, operations, and properties of the primary objects are described in object models. Prototypes or mockups of the user interface can also be made to aid comprehension.

Object-oriented design

An implementation constraint is applied by a developer during object-oriented design (OOD) to the conceptual model created during object-oriented analysis. The platforms for the hardware and software, the demands for performance, the need for permanent storage and transactions, the usability of the system, and time and money limits are only a few examples of such restrictions.

The technology-independent concepts from the analytical model are translated onto implementing classes and interfaces to create a model of the solution domain, or a thorough description of how the system will be implemented using specific technologies. The application of architectural patterns and design patterns along with object-oriented design concepts is another important topic covered throughout OOD.

Object-oriented modeling

Modeling applications, systems, and business domains using the object-oriented paradigm across the course of the whole development life cycle is known as object-oriented modeling (OOM). In contemporary software engineering, OOM is a key approach that is heavily utilized by both OOD and OOA activities. The two main components of object-oriented modeling are the modeling of static structures, such as classes and components, and the modeling of dynamic behaviors, such as business processes and use cases. The two unique abstract levels during OOM are OOA and OOD. The two most well-known worldwide standard languages used for object-oriented modeling are the Unified Modeling Language (UML) and SysML.

The benefits of OOM are comprehensive manuals and well-written programming language codes are generally difficult for users to grasp. Users and stakeholders can provide developers with input on the proper requirements and system structure through the usage of visual model diagrams, which can be easier to understand.

Reducing the "semantic gap" between the system and the outside world and having the system built using vocabulary that is nearly identical to what the stakeholders use in daily business is one of the main objectives of the object-oriented approach. Object-oriented modeling is a crucial tool for making this possible.

Modeling helps coding. Most contemporary software methodologies

Correspondence to: Weifan Liu, Department of Computer Science and Centre for Health Informatics, University of Manchester, Manchester, UK, E-mail: Weifan999@yahoo.com

Received: 19-Oct-2022, Manuscript No. GJEDT-22-21571; **Editor assigned:** 25-Oct-2022, PreQC No. GJEDT-22-21571 (PQ); **Reviewed:** 09-Nov-2022, QC No. GJEDT-22-21571; **Revised:** 16-Nov-2022, Manuscript No. GJEDT-22-21571 (R); **Published:** 23-Nov-2022, DOI: 10.35248/2319-7293.22.11.157

Citation: Liu W (2022) Object-Oriented Analysis and Design and the Benefits. Global J Eng Des Technol.11:157

aim to answer "what" questions first, followed by "how" questions, i.e., first determine the functionality the system is to offer without taking implementation constraints into account, then think about how to make specific solutions to these abstract requirements, and refine them into detailed designs and codes by constraints like technology and budget. This is made possible by object-oriented modeling, which generates abstract and understandable descriptions of both system requirements and designs. Models that define a system's fundamental structures and behaviors, such as processes and objects, are significant and valuable development assets with higher abstraction levels than concrete and difficult source code.