Afropolitan Journals

ISSN: 2762-4585X www.afropolitanjournals.com

Object Oriented Programming Approach: A Panacea for Effective Software Development

Agu S. C.[1] and Elugwu F.[2]

[1]Department of Computer Science, Madonna University Nigeria. [2]Department of Computer Science, Delta State Polytechnic, Otefe, Nigeria.

Abstract

This paper studied object-oriented programming (OOP) as an effective solution approach to software development. It reviewed the Object Oriented Software Engineering (OOSE) Model and also the basic concepts of OOP which characterized its impacts and pointed to its goals towards software engineering. In addition, it examined the advantages and disadvantages of OOP. If further discussed how OOP is an improvement to software development, the reasons for object cloning, late static binding and how it is deployed and the basics of Model-View-Controller (MVC) Architecture as a good way to develop clean, scalable, powerful and fast applications in the least amount of time and with the least effort.

Keywords: Object Oriented Programming, Software Development, Object Cloning, Static Binding, Model-View-Controller Architecture.

Introduction

By analytically considering the history and evolution of programming techniques, Object oriented Programming (OOP) has introduced tremendous changes into the programming world with numerous comparative advantages of OOP over a few earlier programming techniques (Onu *et al*, 2015), thus stating that stakeholders in software development industry have felt the remarkable impact of OOP on modern software development process.

An Object- oriented software (OOS) is software designed using object oriented approach and implemented using an object oriented language (Raju *et al*, 2011).

Procedural software approaches emphasis is on what procedure is followed for the program execution and which function is used. It includes number of steps or procedures that the system must follow and also it organizes these procedures into several groups known as functions. The functions created in the program share the global data and these functions transfer data from one to another. It focuses on following top-down approach of design (Ashwin, 2016).

They were not best fit to adapt new requirement. Most of the performance failure is due to the issues in the development process and in the architecture design phase (Iqbaldeep *et al*, 2016). However, OOS helps us to design higher level of abstraction. It helps to perform manipulation by the programmer into the system. It is acceptable for designing and implementing software system in area range from client server to real time creating the trends which make the development of the software fast and with low costs. Object

Oriented Software Engineering (OOSE) could be described as a combination of activities and object oriented methods on development of the software. OOSE makes a structure in which the methods, processes and tools are combined for development of software. The key point in OOSE is the design and analysis phases which play the role and the relation dependency.

According to (Raju *et al*, 2011), most of the software developed nowadays follows the steps of Software Development Life Cycle (SDLC). Hence, Unified Modelling Language (UML) specifications become handy. UML has become the de- facto standard for analysis and design of OOS.

There is a massive movement from the procedural style of software development to an object-oriented style in recent years by large software development industries. Although safety-critical software developers have largely resisted this trend because of concerns about verifiability of object-oriented systems, there are considerable enhancements and benefits offered by object technology and the key features of the object-oriented approach from a user's perspective. There are main issues affecting safety which a paradigm was proposed - Verified Design-by-Contract – that uses formal methods to facilitate the safe use of inheritance, polymorphism, dynamic binding and other features of the object-oriented approach (David, 2005).

The Object Oriented Software Engineering (OOSE) is a combination of Object Oriented Analysis (OOA) models, Object Oriented Design (OOD) and the Object Oriented Programming (OOP) which provide a powerful way for development of the software, covering the general terms and issues which effects software development in industry (Iqbaldeep *et al*, 2016).

Basic Concepts of Object Oriented Programming (OOP)

OOP changed the utilization of procedural oriented programming, where the attention was on the methodology of execution into the way objects collaborate to convey and share the data (Ashwin, 2016). A language is defined as being an object-oriented language if it supports the following basic concepts which characterized the impact of OOP to software development.

(a) The Object - Objects are the basic run-time entities in an object-oriented system. Programming problem is analyzed in terms of objects and nature of communication and transfer of information between them (Onu *et al*, 2015). They are the basic building block of OOP representing the way anything can be described in the programming world (Ashwin, 2016). They are autonomous entities that respond to *messages* (Mehmet & Lodewijk, 2013). An object has the following characteristics. * *State:* For recording the history of an object. * *Behaviour:* The observable effects based on its state and the relations with other objects. * *Identity:* As known by other objects, either by name or by reference. * *Relations:* between objects are expressed by interactions in the form of message passing (Iqbaldeep *et al*, 2016).

(b) The Class - a class is a collection of objects of similar type. Once a class is defined, any number of objects can be created which belong to that class (Onu *et al*, 2015). It abstracts and groups objects according to their common behavior. These individual objects, also called *instances*, can only maintain specific properties in the form of unique values of their local variables (Mehmet & Lodewijk, 2013). It is the principal body of any system. The classes frame the fundamental improvement unit of any system (Ashwin, 2016).

(c) Inheritance - is the process by which objects can acquire the properties of objects of other class. It provides reusability that is, creating additional features to an existing class without modifying the class which is achieved by deriving a new class from the existing one where the new class will have combined features of both the classes with the net benefit of avoiding redundancy in our code and increasing the extensibility factor of our class (Onu *et al*, 2015). A new class known as derived class can be created from the parent class which helps in reducing the coding time and the derived class is error free (Ashwin, 2016). It is a structural organization of classes, whereby a class may inherit operations and/or local variables from its *super classes*, or may have its operations and/or local variables inherited by its *subclasses*. This structural inheritance relation provides for the organization of classes so that they can be systematically reused. Classes may also inherit from multiple parents, providing additional reusability (Mehmet & Lodewijk, 2013). It is a powerful feature of object oriented language like java. It is an operation in which an object can get traits from another object. So, using the inheritance we can inherit traits which are needed from the super class and inherit down to the base class (Iqbaldeep *et al*, 2016).

(d) The Polymorphism – the ability to take more than one form. An operation may exhibit different behaviors in different instances (Onu *et al*, 2015). The capacity to ask for that the same operations be performed by an extensive variety of diverse sorts of things (Ashwin, 2016). Polymorphism allows different objects to respond to the same message. The message passing semantics enable each object to respond to the same message in a way appropriate to the object (Mehmet & Lodewijk, 2013). It means defining more than one same name functions with different structure. It makes a function to get implemented in different ways in classes and sub classes and get different forms. It creates a common interface for different implementation of a function for the programmer which operates different for different objects (Iqbaldeep *et al*, 2016).

(e) Data Encapsulation – Once an Object is created, knowledge of its implementation is not necessary for its use thus, preventing programmers from tampering with values they should not and also allowing object controls how one interacts with it thereby preventing other kinds of errors. For example, a programmer (or another program) cannot set the width of a window (Onu *et al*, 2015). It hides the internal data structures defined by *local variables* within an object. The local variables of an object are only accessible through its operations (Mehmet & Lodewijk, 2013).

(f) Generalization - describes a relationship between a general kind of thing and a more specific kind of thing. This type of relationship is often described as an "is a" relationship.

For example, a car is a vehicle and a truck is a vehicle. In this case, vehicle is the general thing, whereas car and truck are the more specific things. Generalization relationships are used for modeling class inheritance and specialization. A general class is sometimes called a superclass, base class, or parent class; a specialized class is called a subclass, derived class, or child class (w3Computing, 2016).

The Object Oriented Software Engineering Models

OOSE is composed of Object Oriented Analysis (OOA) and Object Oriented Design (OOD) models which create a framework for software development. The goal of OOA is the development of a model which describes the requirements of the users. It studies the feasibilities, validates the specifications and manages the requirements. It identifies all classes and the relationships between the classes and their behaviors. OOD is the next step to OOA. It focuses on organizing the objects in the classes and all methods and the functions in a class are defined in OOD phase. It is a process in which the user requirements are transformed to a design for software creation (Iqbaldeep *et al*, 2016). Figure 1 shows mapping view of analysis and design model.

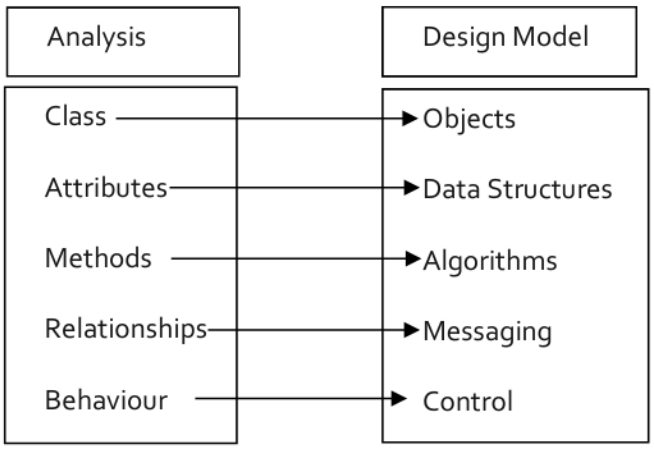

Figure 1: Mapping View of Analysis and Design Model (Iqbaldeep *et al*, 2016)

Software Engineering Goals

According to Ashwin (2016), the three basic goals of software engineering are reusability, extensibility and flexibility. Reusability reduces the time for producing a code. Once code is developed, it becomes easy and efficient to reuse it many times. It also increased the reliability of the code as the previously existing codes are used for development. Extensibility can be related to the concept of inheritance in OOP. The attribute or behavior that is set for the base class will be extended to its derived class automatically and hence the same attribute can be used to refer the derived class variables. OOP is very flexible in terms of software development. Its flexibility can be understood by polymorphism, where we can add more variations and do modifications using the same function name but with different variables. Figure 2 shows the *software engineering goals.*

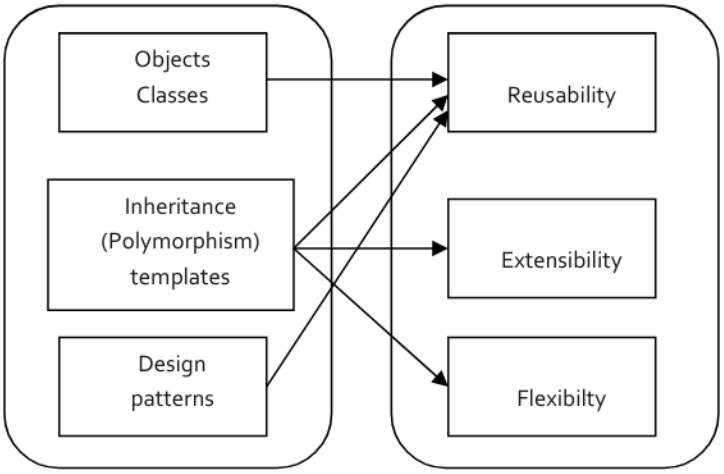

Figure 2. Software Engineering Goals (Ashwin, 2016).

Literature Review

Notable scholars spread the software crisis as a conventional struggle between management and software workers, with both new management technologies and new software technologies (software engineering, structured programming, object-oriented programming) developed as solutions, primarily pushed by management, to solve the supposed crisis. Yet, programmers and analysts have advocated object oriented programming for its technical advantages over previous methods that could solve the software crisis (Hansen, 2013). Noting that proper object-oriented design methodology cannot be imposed by the technology (the language) itself, but that the designer must first transform his/her way of thinking over to the new paradigm. This was to buttress the point on the controversy over dynamic typing and binding in light of the histories of Smalltalk and C++ as possible counter examples to the thesis that management favors higher abstractions to manage complexity.

In a bid to maintain IEEE standard glossary of software engineering in object oriented design software, software metrics are deployed for measuring the software complexity, estimating size, quality and project efforts using various approaches through which the software cost and predicates on various kinds of deliverable items can be estimated (Sanjeev, 2016). Consequently, different metrics like software quality metrics, object oriented metrics, size metrics, CK metrics, QMOOD metrics, GQM metrics, MOOSE metrics and EMOOSE metrics proposed from early 90's were classified. Comparison tables that analyze the difference between all object oriented metrics easily were also maintained.

Although a considerable number of object-oriented software development methods have been introduced to produce extensible, reusable, and robust software, a number of important obstacles that are not addressed by current methods have been summarized and evaluated with respect to pilot applications (Mehmet & Lodewijk, 2013).

Combining object-orientation and parallelism paradigms have attracted tremendous benefits from numerous research projects and an increasing number of practical

applications. It has been realized that parallelism is an inherently needed enhancement for the traditional object-oriented programming (OOP) paradigm, and that object orientation can add significant flexibility to the parallel programming paradigm (Rajesh, 2015).

Aspect Oriented Programming (AOP) is a technique used to enhance the separation of concern in software design and implementation. Aspect Oriented Design language (AODL) is a design notation build on the existing design language UML. It is a notation used to show the interaction between traditional UML models of base (Object Oriented) code and Aspect extensions, such as point cuts and join points. It aims at investigating how to identify Aspects in the design stage, then how they can be applied in the implementation process in software development. This will lead to improve modularity and reusability of the software (Mazen & Gary, 2015).

The concepts of Object-Oriented Programming Languages (OOPLs) can be deployed in modeling Geographic Information System (GIS) pointing out some desirable features of GIS software with a view to determining the relevance of OOP in GIS modeling. Current GIS Modeling software features imbibe several OOP techniques and concepts. For instance, a GIS software, ArcGIS reveal that GIS software models have close affinity with OOP implementation languages like C++, Java, C# (Onu *et al*, 2016).

Object-oriented analysis (OOA) makes compatible with newly develop or other existing business computing application in a better way where different objects classes are used in modeling the exact procedure or near to the exact procedure within its application domain. Objects are basically structured into different classes of objects which are generally related to behaviours and characteristics (Momin, 2016). These methodologies may use different generalization, classification, and different aggregation as a structure object assemblies for the target actions like services or activities which are related to the objects. OOA represent different advantages and various application of the UML.

Testing of object oriented software is different from testing software using procedural language. Several new challenges are posed. Hence, new technology provides features for testing at unit (class) level as well as integrating level. A maintenance level component has also been incorporated. According to Raju *et al*, (2011), results of applying this tool have been incorporated, and have been found to be satisfactory. Testing is conducted at 3 levels: Unit, integration and system. At the system level there is no difference between the testing techniques used for OO software and other software created using a procedural language where it might take more than one test case to determine the true functionality of the application being tested. However, software development methodologies like Rational Unified Process (RUP) recommend creating at least two test cases for each requirement or objective; one for performing testing through positive perspective and the other through negative perspective.

Very handy in object oriented software is the knowledge of the fundamental principles, thus mitigating common faults by beginning programmers such as (a) non-referenced class fault – knowing a class to model a concept about but cannot figure out how to integrate the class

into their designs. (b) Misrepresentations - failure to achieve high cohesion and coupling in class design at an early stage instead of postponed it until a discussion of metrics in advanced level courses and (c) misconceptions - such as object/variable and object/class conflation and identity/attribute confusion and failure to use collection were much evidenced (Benjy *et al*, 2006).

Advantage and Disadvantage of OOP

The advantages and disadvantages of object-oriented programming is presented, which is a well-adopted programming style that uses interacting objects to model and solve complex programming tasks (Saylor, 2018).

Some of the advantages of object-oriented programming include:

1. Improved software-development productivity: Object-oriented programming is modular, as it provides separation of duties in object-based program development. It is also extensible, as objects can be extended to include new attributes and behaviors. Objects can also be reused within and across applications. Because of these three factors – modularity, extensibility, and reusability – object-oriented programming provides improved software-development productivity over traditional procedure-based programming techniques.
2. Improved software maintainability: For the reasons mentioned above, object oriented software is also easier to maintain. Since the design is modular, part of the system can be updated in case of issues without a need to make large-scale changes.
3. Faster development: Reuse enables faster development. Object-oriented programming languages come with rich libraries of objects, and code developed during projects is also reusable in future projects.
4. Lower cost of development: The reuse of software also lowers the cost of development. Typically, more effort is put into the object-oriented analysis and design, which lowers the overall cost of development.
5. Higher-quality software: Faster development of software and lower cost of development allows more time and resources to be used in the verification of the software. Although quality is dependent upon the experience of the teams, object oriented programming tends to result in higher-quality software.

According to Saylor (2018), some of the disadvantages of object-oriented programming include:

1. Steep learning curve: The thought process involved in object-oriented programming may not be natural for some people, and it can take time to get used to it. It is complex to create programs based on interaction of objects. Some of the key programming techniques, such as inheritance and polymorphism, can be challenging to comprehend initially.
2. Larger program size: Object-oriented programs typically involve more lines of code than procedural programs.

3. Slower programs: Object-oriented programs are typically slower than procedure based programs, as they typically require more instructions to be executed.
4. Not suitable for all types of problems: There are problems that lend themselves well to functional-programming style, logic-programming style, or procedure-based programming style, and applying object-oriented programming in those situations will not result in efficient programs.

Discussion and Findings

How OOP is an Improvement to Software Development

OOP approach has improved software development (Skoglund, 2008).

- Grouping codes together: all the codes that relate to a class can be grouped together. A typical database class with methods are presented below:

```
Class MySQLDatabase {
        Private $connection;
        Function __construct() {
        Public function open_connection() {
        Public function close_connection() {
        Public function query($sql) {
        Public function escape_value($value) {
        // "database neutral" methods
        Public function fetch_array($result_set) {
        Public function num_rows($result_set) {
        ....
        ....
}
```

- Modularization: we can swap from any class to another if there is need. For instance, swapping from any class to MYSQLDatabase class is achieved as follows:

```
$database = new MySQLDatabase();
$db = $database;
```

- Methods: are written in the classes enabling codes to be class neutral.
- Consistency: improvements made to the objects are available on every page where the object is instantiated or called.

Object Cloning

Cloning an object simply means copying an object. One pertinent question has been what is the purpose of copying an object in OOP? The practical examples of why it should be done, *a* real life example demonstrating the usefulness of it.

There are a number of situations that require cloning of an object (Mike, 2015). First, reducing extra effort in initialization; when an object is created, it is usually initialized, either in the form of calling methods to set values, or in the form of passing parameters to the constructor. Most often, this initialization demands extra effort. For instance, if a new object Y is needed which differs from another object Z by a single value, and then it may be simpler to clone Z as Y and alter a single value of Y instead of creating Y.

Second, repetition of an algorithm; For instance, when chess playing algorithm wants to make next move, it internally makes many copies of the current board, modify each one of the boards with one of many possible moves that it may make, evaluate each one of the new boards using some heuristics, pick the best, and then use that as the new current board. (Copying the best board to the current board is more likely by setting a 'current board' reference to point to the best board)

Third, the concept of a defensive copy; Assume a Person object is required to provide its DateOfBirth, it may not necessarily return a reference to its own DateOfBirth, because this object may be altered, thus changing the Person object DateOfBirth. As a result, the Person object is probably to return a *defensive copy* of its DateOfBirth. A similar concept is the concept of a *snapshot copy*. An event that is needed to pass a list of event handlers is expected to make a (shallow) snapshot copy of the list before invoking the handlers because some of the event may be removed from the list while the list is been processed. Hence, failure to make snapshot copies may have negative consequences. This explains the need for avoiding ConcurrentModificationException.

Late Static Binding

Ordinarily, inheritance allows a subclass to inherit a super class which enables the subclass to use the objects and methods of the super class. This is what is obtainable in OOP basics and it is referred to as *call-forwarding* (PHP, 2018). This actually creates a problem because when the OOP compiler loads, it binds every occurrence of the super class to its methods and object even before the sub classes are defined. This happens early when the object are created, it does not happen later when the objects are called. This is also referred to *early static binding*.

In complex situations, many subclasses inherit a super class. In this scenario, the super class will be inherited but the methods and the objects of the super class would not be directly worked with or used. Rather the methods and the objects of the inherited sub classes would be required. But early static binding technique poses a big problem to this advanced OOP concept. To avoid this problem, three possible techniques are deployed

- Avoid static method: static method is avoided while instance methods are used (Skoglund, 2008). For instance

```
$user = $User::find_by_id(1);
$user = new User();
$user = $User->find_by_id(1);
```

- Deduce the class name (Skoglund, 2008): by using get_class() method. Unfortunately, this approach will return the super class object. Specialized skills beyond the scope of this paper are needed to use this method. This approach is not recommended
- Late Static Binding: a new feature of OOP (JAVA, PHP, C# etc) which can be adopted any of the followings using PHP (Skoglund, 2008).
  i) Get_called_class(): used instead of get_class method to find out what class actually called the static method in question. This will allow us to determine the class name, and then we can create a new instant of the class.

```
Private static function instantiate() {
$class_name = get_called_class();
$object = new $class_name;
```

  ii) Static scope resolution operator: instead of self scope resolution operator

Code 1:

```
Public static function find_by_id($id=0) {
        $result_array = self::find_by_sql("SELECT * FROM" . self::$table_name.
 "WHERE id = {$id} LIMIT 1);
}
```

Code 2:

```
Public static function find_by_id($id=0) {
        $result_array = static::find_by_sql("SELECT * FROM" .
static::$table_name. "WHERE id = {$id} LIMIT 1);
        }
```

The problem in code 1 is the keyword *self:: scope resolution operator*. Any time *self* is used in static method, it refers to the actual method where it resides instead of the method of the class inheriting it. Code 2 rectified the problem by replacing self with *static scope resolution operator* which enables late static binding, a binding that happens at run-time.

The Basics of MVC Architecture

MVC, which stands for Model-View-Controller, is a really good way to develop clean, scalable, powerful and fast applications in the least amount of time and with the least effort (Gaurav, 2015). The Model-View-Controller concept involved in software development evolved in the late 1980s. It is a software architecture built on the idea that the logic of an application should be separated from its presentation. A system developed on the MVC architecture should allow a front-end developer and a back-end developer to work on the same system without interfering with each other.

Model: is the name given to the component that will communicate with the database to manipulate the data. It acts as a bridge between the View component and the Controller component in the overall architecture. It doesn't matter to the Model component what happens to the data when it is passed to the View or Controller components. The code snippet for running *first_model.php* is:

```
<?php
class Model
{
public $string;
public function __construct()
{
$this->string = "Let's start php with MVC";
}
}
?>
```

(Gaurav, 2015).

View: The View requests for data from the Model component and then its final output is determined. View interacts with the user, and then transfers the user's reaction to the Controller component to respond accordingly. An example of this is a link generated by the View component, when a user clicks and an action gets triggered in the Controller. To run *first_view.php*, type:

```
<?php
class View
{
private $model;
private $controller;
public function __construct($controller,$model)
{
$this->controller = $controller;
$this->model = $model;
}
public function output()
{
return "<p>" . $this->model->string . "</p>";
}
}
?>
```

(Gaurav, 2015).

Controller: The Controller's job is to handle data that the user inputs or submits through the forms, and then Model updates this accordingly in the database. The Controller is nothing without the user's interactions, which happen through the View component. The code snippet for running *first_controller.php* is:

```
<?php
class Controller
{
private $model;
public function __construct($model)
{
$this->model = $model;
}
}
?>
```
(Gaurav, 2015).

A simple way to understand how MVC works is given below.

1. A user interacts with View.
2. The Controller handles the user input, and sends the information to the model.
3. Then the Model receives the information and manipulates it (either saving it or updating it by communicating with the database).
4. The View checks the state of the Model and responds accordingly (lists updated information).

The following code snippet, *first_example.php* is the MVC architecture work with PHP.

```
<?php
$model = new Model();
$controller = new Controller($model);
$view = new View($controller, $model);
echo $view->output();
?>
```
(Gaurav, 2015).

Conclusion

An object-oriented software is software designed using object oriented software engineering model approach and implemented using an object oriented language to enhance the procedural and other former approaches used in software development considering its basic concepts of classes, objects, inheritance, encapsulation, and polymorphism that promote reusability, extensibility and flexibility.

Object Cloning became very handy in software development as it reduces the extra effort in initialization of object, repetition of algorithm and introduced the concept of defensive copy, snapshot copy that mitigates negative consequences that could occur in the process of cloning thus avoiding ConcurrentModificationException.

Late Static Binding, a binding that happens at run-time allows many subclasses inherit a super class, enabling the methods and the objects of the inherited sub classes to be used while preventing the methods and the objects of the super class from being used thus eliminating the problem (not adopting proper inheritance) that is created by early static binding, a binding that happens early when the object are created.

Model-View-Controller software architecture is built on the idea and logic that an application should be separated from its presentation enabling a system to have a front-end developer and a back-end developer to work on the same system without interfering with each other. The architecture introduced Model - as the component that will communicate with the database to manipulate the data, View - requests for data from the Model component and then its final output is determined, and the Controller's job - is to handle data that the user inputs or submits through the forms, and then Model updates this accordingly in the database.

References

Ashwin, U. (2016). Object-Oriented Programming and its Concepts, *International Journal of Innovation and Scientific Research*. *26(1), pp 1-6.*

Benjy, T., Mark, R., & Lynda, T. (2006). *Identifying Novice Difficulties in Object Oriented Design*, Accessed August 1, 2018 http://users.aber.ac.uk/ltt/RESEARCH/fp099-thomasson.pdf

David C., (2005), *Safe Object-Oriented Software: The Verified Design-By-Contract Paradigm*. Accessed August 1, 2018 https://www.eschertech.com/papers/safe_oo_software.pdf

Gaurav, K. (2015). *The Basics of MVC Architecture in PHP*, Accessed August 15, 2018 https://opensourceforu.com/2015/12/the-basics-of-mvc-architecture-in-php/

Hansen, H. (2013). *Connections between the Software Crisis and Object-Oriented Programming.* Cornell University, Science & Technology Studies *SIGCIS History of Computing Workshop in Memory of Michael S. Mahoney.* Accessed August 1, 2018 https://www.sigcis.org/files/Hsu%20--%20Software%20Crisis%20and%20OOP.pdf

Iqbaldeep, K., Navneet, K., Amandeep, U., Jaspreet, K. & Navjot, K. (2016). *Object Oriented Software Engineering*, Accessed August 1, 2018 http://www.ijcst.com/vol74/1/8-iqbaldeep-kaur.pdf.

Mazen, I. G., & Gary, A. (2015). *Improving the Design and Implementation of Software Systems uses Aspect Oriented Programming.* Accessed August 1, 2018 http://eprints.hud.ac.uk/id/eprint/24048/3/AllenInformation.pdf

Mehmet, A., & Lodewijk, B. (2013). *Obstacles in Object-Oriented Software Development*. University of Twente Faculty of Computer Science Enschede, The Netherlands. Accessed August 1, 2018 http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.106.1584&rep=rep1&type=pdf

Mike, N. (2015). *What is the purpose of copying an object in object oriented programming?* Accessed August 15, 2018 https://stackoverflow.com/questions/28410866/what-is-the-purpose-of-copying-an-object-in-object-oriented-programming.

Momin, M. (2016). Object-Oriented Analysis and Design. *International Journal of Advanced Engineering and Management, Technical and Scientific Publisher*. 1(1), pp 18-24.

Onu, F. U., Okorie, K., & Ugwa, C. U. (2015). Effects of Object-Oriented Programming on Modern Software Development. *International Journal of Computer Applications Technology and Research*. *4(11), pp 829 – 837*.

Onu, F. U., Uche-Nwachi, E., Chigbundu, K. E. (2016). The Role of Object-Oriented Programming (OOP) in Modeling of Geographic Information Systems (GIS). *IOSR Journal of Mobile Computing & Application. 3(4), pp 40-46.*

PHP, (2018). *Late Static Binding*, Sunshine PHP 2019 CFP. Accesses August 14, 2018 http://php.net/lsb.

Rajesh, M. (2015). Object-Oriented Programming and Parallelism. *International ISTE 5(9).*

Raju, P.D.R., Cheekaty, S., & Kalidasu, H. (2011). *Object Oriented Software Testing*, *International Journal of Computer Science and Information Technologies*. 2(5), pp 2189-2192.

Sanjeev, K. P., Parveen, K., & Anuj, G. (2016). A Review of Software Quality Metrics for Object-Oriented Design. *International Journal of Advanced Research in Computer Science and Software Engineering. 6(8).*

Saylor, F. (2018). *Advantages and disadvantages of OOP*. Accessed August 14, 2018 https://www.saylor.org/site/wp-content/uploads/2013/02/CS101-2.1.2-AdvantagesDisadvantagesOfOOP-FINAL.pdf

Skoglund, K. (2008). *PHP with MySQL Beyond the Basics*. video tutorial, via Lynda.com

w3Computing, (2016), *Systems Analysis and Design (SAD) Tutorial.* Accessed August 13, 2018 http://www.w3computing.com/systemsanalysis/