ATLANTIS PRESS

Advances in Intelligent Systems Research (AISR), volume 157

Mathematics, Informatics, Science, and Education International Conference (MISEIC 2018)

The Use of Design Pattern on Informatics Engineering Students Thesis

Alifah Diantebes Aindra,
Oktarica Pratiwi Suryoningtyas,
and Aji Prasetya Wibawa
Department of Vocational Education
Universitas Negeri Malang
Malang, Indonesia
diantebes.aindra@gmail.com,
octa.riecha@gmail.com,
aji.prasetya.ft@um.ac.id

Aim Abdulkarim
Department of Civil Education
Universitas Pendidikan Indonesia
Bandung, Indonesia
aim.abdulkarim@upi.edu

Eki Nugraha
Department of Computer Science
Universitas Pendidikan Indonesia
Bandung, Indonesia
ekinug85@yahoo.com

***Abstract*— University students should write a thesis to get their undergraduate (Bachelor) degree is. Most Informatics engineering students of the Electrical Engineering Department in Universitas Negeri Malang use research and development method for their thesis. There are many solutions or methods in software engineering which aim to develop a good quality software. One of these solutions is using design patterns. Several benefits of using the design patterns are; decreasing time of production, having a reusable and flexible code, and making people easier to understand. The purposes of the study were investigating whether or not the informatics engineering students use the design pattern for their thesis and what kind of design pattern that they use. The data of the study were obtained through distributing questionnaires to the informatics engineering students. The questionnaire being used is a close-ended questionnaire that provides options to be chosen. The percentage descriptive analysis was used to analyze the data. Therefore, the results showed that in fact, most of the informatics engineering students have already used the design patterns to develop their thesis although they do not realize on what kind of the design patterns they actually used in the study.**

Keywords*— *Design patterns; informatics student

I. INTRODUCTION

Each education level has to conduct a final examination as a graduation's requirement for students to be able to continue to higher education level. At the university level, students have to write a thesis or a final assignment to graduate from a particular university. They start to finish the final assignment at the beginning of the 8th semester, which means they only have a semester or approximately 5 months to complete it. Some informatics engineering students at Universitas Negeri Malang use developmental research design for their thesis. They developed a software that was later used to solve a problem they had previously discovered. Many solutions can be made during the process of software development to reduce the development process so that students can complete their thesis quickly. By reducing the development time of a software, it can also reduce the costs during the production process [1]. One of these solutions is to use design patterns for the development. The use of a design pattern can reduce the development time, facilitate communication between the development team, written source code becomes more flexible and easy to reuse [2].

Unlike the above mentioned, to implement or adapt the design pattern, a proper and good preparation from the programmer are needed as to clearly understand how to use the design pattern, understand the object-oriented programming and choose the right type of design pattern. Therefore the informatics engineering student must know and understand very well about the design pattern , the types of design patterns and how to implement them because using a design pattern will also affect the quality of the developed software [3] or if the design pattern is not implemented correctly, they will actually be losing the source code or software [2]. There are several types of design patterns and not all of them can be used in any software development, so choosing the right design pattern should be done well [4].

This article aims to investigate the use of a design pattern for informatics engineering students thesis and observe what kind of design pattern do they use. This article consists of the definition and many kinds of design pattern, method, result and discussion and, the last, conclusion.

II. DESIGN PATTERN

Design pattern defined by Gamma et al. [5] as "a general reusable solution to the commonly occurring problem in software design". A design pattern is a description or template for how to solve a problem that can be used in many different situations. It describes the core of the solution to the problem in such a way this solution can be used many times over, without doing it in the same way twice [5]. Design patterns improve software documentation, speed up the development process, enable large-scale reuse of software architectures, capture expert

knowledge, capture design trade-offs and help re-structuring systems [6]. There are three basic classes of design patterns [2], the scheme can be seen in Figure 1.

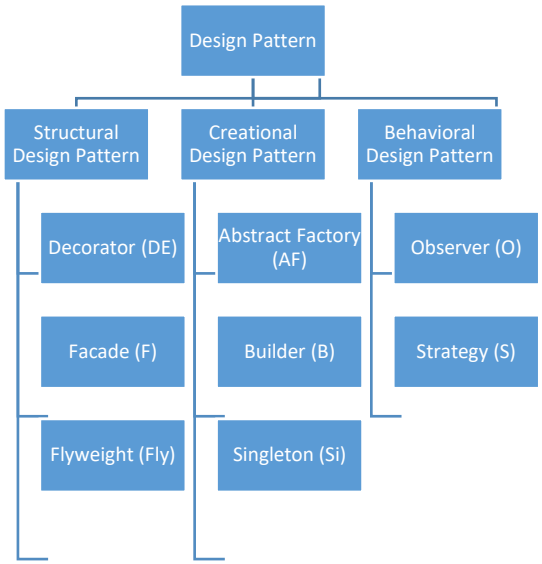

Fig. 1. Scheme of Design Pattern Classes

The first class is Structural design patterns, this group of design patterns eases software design by identifying a simple way to realize relationships between entities. Such patterns are all about Class and Object composition. Structural class-creation patterns use inheritance to compose interfaces while structural object-patterns define ways to compose objects to obtain new functionality. Examples of structural design patterns include: the first is Decorator (DE), Attach additional responsibilities to an object dynamically and provide a flexible alternative to subclassing for extending functionality [7]. Second is Façade (F), provide a unified interface to a set of interfaces in a subsystem. Façade defines a higher-level interface that makes the subsystem easier to use [8]. Third is Flyweight (Fly). Flyweight use sharing to support large numbers of fine-grained objects efficiently. Fourth is Proxy (P). It provides a surrogate or placeholder for another object to control access to it, etc [5].

The second Class is Creational design patterns. These design patterns are basically concerned with class instantiation and composed of two dominant ideas. One is encapsulating knowledge about which concrete classes the system uses and the other hides how instances of these concrete classes are created and combined [5]. Creational design patterns are further categorized into Object-creational patterns and Class-creational patterns, where Object-creational patterns deal with Object creation and Class-creational patterns deal with Class-instantiation. The examples of creational design patterns are: (1) Abstract Factory (AF) that provides an interface for creating families of related or dependent objects without specifying their concrete classes. (2) Builder (B) which separates the construction of a complex object from its representation so that the same construction process can create different representations, and (3) Singleton (Si) that ensures a class only has one instance, and provide a global point of access to it [5].

The third Class is Behavioral design patterns, that identify common communication patterns between objects and realize the assignment of responsibilities between objects. By doing so, these patterns increase flexibility in carrying out this communication because they shift focus away from the flow of control and concentrate just on the way objects are interconnected. The examples of behavioral design patterns are: (1) Observer (O) that define a one-to-many dependency between objects so that when one object changes state, all its dependents are notified and updated automatically and (2) Strategy(S) which defines a family of algorithms, encapsulate each one, and make them interchangeable. Strategy lets the algorithm vary independently from clients that use it [5].

III. METHOD

This study used survey methodology based on Cresswell [9]. The data was collected through distributing questionnaires to 100 informatics engineering students at Universitas Negeri Malang. The validated questionnaire contains questions that identify each characteristic on the design pattern types. By answering the question based on the source code they have written, the researcher can find out what design pattern they are using. The maximum value of each characteristic on the types of design patterns resulted from questionnaire can be seen in Table 1.

TABLE 1. SELECTED DESIGN PATTERN

Name of Design Pattern	**Max Value of Characteristics of Design Pattern**
Creational Design Pattern	
Singleton	3
Abctract Factory	3
Builder Pattern	4
Structural Design Pattern	
Flyweight Pattern	2
Proxy Pattern	2
Facade Pattern	2
Decorator Pattern	5
Behavioral Pattern	
Strategy Pattern	3
Observer Pattern	3

The questionnaire used is a close-ended questionnaire because the questionnaire has provided a choice of answers to choose (Yes or No). Therefore, the nominal data included in the discrete data will be obtained. In order to analyze the data obtained from the questionnaire, the percentage descriptive analysis was used and types of pattern design were grouped. The researchers also conducted interviews to find out whether using a design pattern assisted the student in developing a software, and if it is not, what are the causes for addition information. In this study, researchers focused on a well-known design pattern, so it was assumed that more programmers would use it [3]. The researchers start to collect the data at 26th of March 2018 until 14th of April 2018.

IV. RESULT AND DISCUSSION

After distributing questionnaires for three weeks and analyzing the data, the results of questionnaire revealed that not all informatics engineering students know about the design pattern. It can be seen in figure 2, most of students do not know what is design pattern. Most of the students who do not know about the design pattern are called group 1 and the rest know

about it is called group 2. Based on the interview results, most of students do not know about the design pattern since it is not taught in class[10], while students who know about it get to know the design pattern from the internet and learn from it.

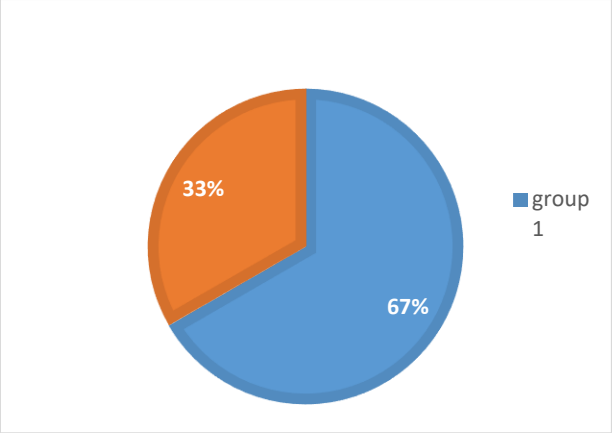

Fig. 2. Informatic students understanding of the design pattern

However, based on the results of the following questionnaire, some informatics engineering students who basically do not know the concept of design pattern, had inadvertently applied or used the design pattern in their thesis as shown in figure 3.

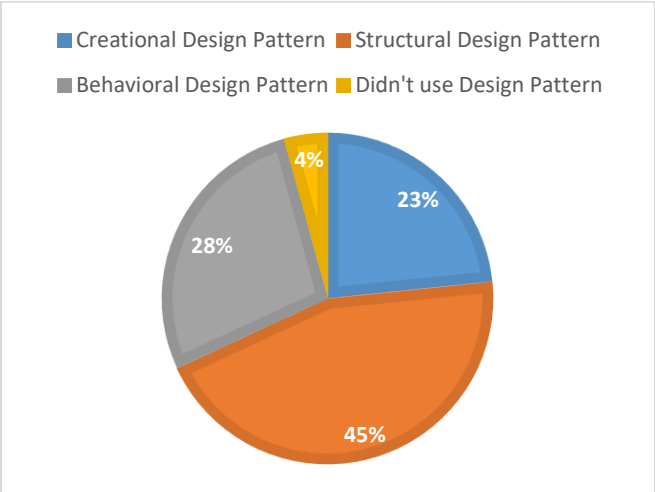

Figure 3. Percentage of informatic students who use design pattern

From the Figure 3, it can be seen that most students (about 96%) applied the design pattern for their thesis, while in figure 2, 67% of students do not know about design pattern before. The deeper analysis about the intersection of the previous two charts, it can be seen on Table 2.

TABLE 2. INTERSECTION OF CHARTS

	Knowing	**Do Not Know**
Apply	33.33 %	53.33 %
Not Apply	0	13.33 %

As shown in figure 4, there are flyweight, proxy, facade and decorator patterns which belong to the type of structural design pattern. In addition to these four types, there are other types of design patterns, but only the four types mentioned are the most widely used structural types. As seen in figure 4, the proxy pattern is the most widely used by informatics engineering students and followed by façade, flyweight and decorator.

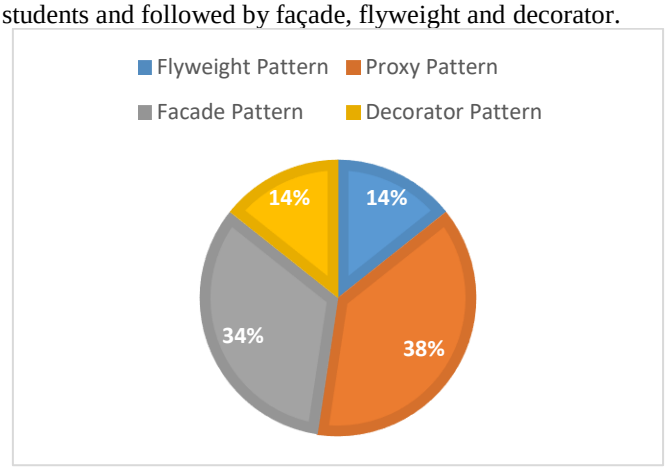

Fig. 4. Percentage of informatic students who use structural design pattern

As seen in figure 5, the Observer pattern is the most common behavioral design pattern used by informatics engineering students. They say that the Observer pattern is like a model-view-controller concept and is usually implemented in multiple frameworks. By using the Observer pattern, informatics engineering students will be easier to sort the parts between the front and rear ends and to track errors.

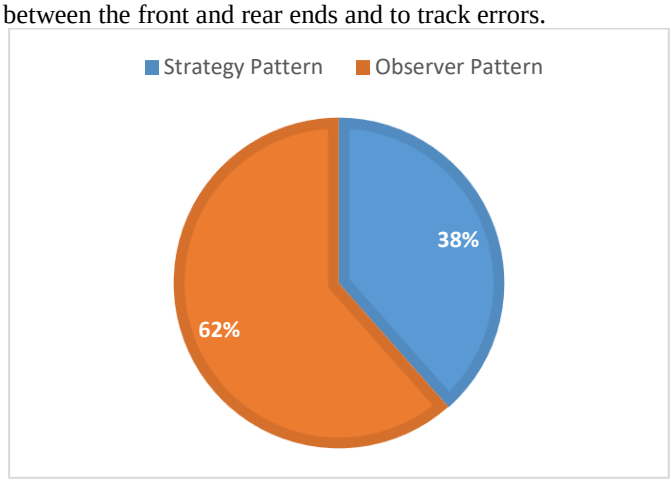

Fig. 5. Percentage of informatic students who use behavioral design pattern

There are several kinds of creational design patterns. In this study, the researchers only take three kinds of the most famous design patterns which are singleton, abstract factory and builder pattern. As shown in figure 6, the creational design pattern mostly used by informatics engineering students is the singleton pattern (55%), the second is abstract factory (36%) and the last is Builder (9%). Abstract factory is a design pattern that has high complexity, but this complexity can be reduced by limiting the number of abstract products. Therefore, the abstract factory is still widely used [11].

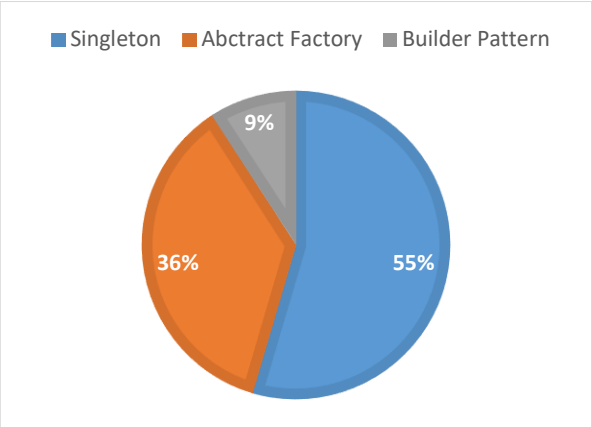

Fig. 6. Percentage of informatic student who use creational design pattern

Based on the results of several charts, it can be seen that most informatics engineering students use design patterns to develop their thesis regardless whether they know about the previous design pattern or not. Singleton patterns, proxies and observers are design patterns for each classification which are most widely used by informatics engineering students. Based on the results of interviews, the average time required by informatics students for developing software can be compared as in the Table 3.

TABLE 3. COMPARISON TIME IN DEVELOPING SOFTWARE

Informatic students	**Time in Month**
With the design pattern	4,3
Without the design pattern	6,5

From data in Table 3, informatics student who apply design pattern is faster in developing software than informatics students who do not apply them. It means that by apply the design pattern can reduce development time[1][2][3]. For more detail about the average time that used by informatics students for each design pattern can be seen in Table 4. Students that used singleton is finish their thesis faster than the student who apply the other design pattern.

TABLE 4. TIME THAT USED FOR EACH DESIGN PATTERN

Design Pattern	**Time in Month**
Singleton	3.08
Abstract Factory	4.625
Builder Pattern	3.5
Flyweight Pattern	5
Proxy Pattern	4.06
Facade Pattern	4.07
Decorator Pattern	5
Strategy Pattern	3.3
Observer Pattern	4.44

Furthermore, informatics engineering students who do not know about design patterns and do not apply them say that they have not heard before or did not study it in college as a course. Then, after knowing this design pattern, they want to learn and apply design patterns because of the advantages. The researchers said that design patterns will make their source code easier to reuse and developed quickly. For informatics engineering students who do not know about design patterns but have applied them said that they have written their thesis code like the characteristics in the questionnaire but they do not know whether the code they write has fully implemented the design pattern properly and correctly.

V. CONCLUSION

From the above explanation, it can be concluded that most students do not know about the design pattern but many of them have inadvertently applied it. From a variety of design patterns at least there are students who use. Based on the in-depth data analysis, there are also some students who apply more than 1 design pattern. This could be the assumption that the use of design patterns has been done by students to address the various problems that exist in their thesis. Because this study only focuses on whether students have used design patterns, this research can only be done as a preliminary study to see the impact of design pattern design implementation for their thesis.

However, due to students' ignorance in applying design patterns, the researchers could not conclude that the implementation of design patterns has been done correctly. Since the design pattern is an abstract concept and has some disadvantages if it is not implemented properly [3], so it is necessary to verify the implementation of design pattern [12]. Design pattern is a complex concept, so when implementing it sometimes requires high complexity, but there are several ways to reduce complexity in accordance with the programmer's wishes so that implementation can be done easily [13]. The suggestion that can be given for college design pattern is that it should be taught as a course because based on internet observations there are some universities that have taught design patterns such as Boston University [14], Oxford University [15], Anadolu University [16], and Open University of UK [17]. Design pattern can also be taught in primary and secondary school as the introduction of programming problem solving [18]. For further research, the quality assessment of the design pattern implementation is needed [19] to see if the implementation of the design pattern is correct and can have a positive impact on thesis to the fullest.

REFERENCES

[1] R. Wojszczyk, "The Experiment with Quality Assessment Method Based on Strategy Design Pattern Example," Inf. Syst. Archit. Technol. Proc. 38th Int. Conf. Inf. Syst. Archit. Technol. – ISAT 2017 Part II, vol. 656, pp. 103–112, 2017.

[2] R. Subburaj, G. Jekese, and C. Hwata, "Impact of Object Oriented Design Patterns on Software Development," Int. J. Sci. Eng. Res., vol. 6, no. 2, pp. 961–967, 2015.

[3] F. Khomh and Y. Guéhéneuc, "Do design patterns impact software quality positively?Hsueh, N. L., Chu, P. H., & Chu, W. (2008). A quantitative approach for evaluating the qualHsueh, N. L., Chu, P. H., & Chu, W. (2008). A quantitative approach for evaluating the quality of design patte," Softw. Maint. Reengineering, 2008. CSMR 2008. 12th Eur. Conf., pp. 1–5, 2008.

[4] A. Ilja, "Use of design patterns for mobile game development," Umeå University, Sweden, 2012.

[5] E. Gamma, R. Helm, J. Ralph, and J. Vlissides, Design Patterns : Element of Reusable Object Oriented Software. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc, 1995.

[6] M. G. Al-obeidallah, "A Survey on Design Pattern Detection Approaches," Int. J. Softw. Eng., vol. 7, no. 3, pp. 41–59, 2016.

[7] Sourcemaking, "Design Pattern," 2018. [Online]. Available: https://sourcemaking.com/.

[8] H. Sihui, "Design Patern : Facade," 2018. [Online]. Available: http://www.sihui.io/design-pattern-facade/.

[9] J. W. Creswell, Educational research: Planning, conducting, and

evaluating quantitative and qualitative research, vol. 4. 2012.
[10] Universitas Negeri Malang, "Kurikulum Program Studi S1 Teknik Informatika," Malang, Indonesia.
[11] A. Bulajic and S. Jovanovic, "An Approach to Reducing Complexity in Abstract Factory Design Pattern," J. Emerg. Trends Comput. Inf. Sci., vol. 3, no. 10, pp. 1411–1418, 2012.
[12] Rafał Wojszczyk and Wlodzimierz Khadzhynov, "The Process of Verifying the Implementation of Design Patterns - Used Data Models," Inf. Syst. Archit. Technol. Proc. 37th Int. Conf. Inf. Syst. Archit. Technol. – ISAT 2016 Part II, vol. 521, pp. 103–116, 2016.
[13] R. Batdalov and O. Nikiforova, "Towards Easier Implementation of Design Patterns," Elev. Int. Conf. Softw. Eng. Adv., no. c, pp. 123–128, 2016.
[14] D. V. Shtern, "Metropolitan College MET CS665 Design Patterns and Component Software Course Syllabus," 2011.
[15] Oxford University, "Course in Software Engineering : Design Pattern," Oxford University, 2018. [Online]. Available: http://www.cs.ox.ac.uk/softeng/subjects/DPA.html. [Accessed: 24-Jun-2018].
[16] Anadolu University, "Design Patterns," Anadolu University. [Online]. Available: http://www.anadolu.edu.tr/en/academics/faculties/course/84892/design-patterns/content. [Accessed: 24-Jun-2018].
[17] N. Schadewitz, "Design Patterns for Cross-cultural Collaboration," Int. J. Des., vol. 3, no. 3, 2009.
[18] R. Fojtik, "Design Patterns in the Teaching of Programming," Procedia - Soc. Behav. Sci., vol. 143, pp. 352–357, 2014.
[19] R. Wojszczyk, "The Model of Quality Assessment of Implementation of Design Patterns," Distrib. Comput. Artif. Intell. 13th Int. Conf., vol. 474, pp. 515–524, 2016.