Transformation of Dependency and Association in UML Design Class

Ngamsantivong Thavatchai, Ratanavilisagul Chiabwoot

Abstract: This paper presents two new conceptual relationships between classes of software development known as dependency and association. The design between the two relationships could be interchangeable because it always takes place in real-life situations — for instance, the relationship from friends to husband-wife and vice versa. However, in terms of coding, the most important factor is system performance. That means the designer could write the code as dependency or association to provide the same result. To improve the efficiency of the program, the researcher writes the code in the C++ language to execute four types of variables named messages, strings, calculation, and sorting. The four types of the variable used to test the performance of aggregation, composition, dependency, and functional programming, the timestamp was used to measure the execution time before and after for each case 50 times. The F-test statistic was used to compare the mean difference of each type of variable. The researcher found that for the Message variable. The functional programming is the fastest, followed by aggregation, composition, and dependency, the average C.P.U. time are 13566.60, 17891.70, 18532.66 and 19336.76, at 0.0 level of significance. For the String variable found that functional programming is the fastest followed by aggregation, composition, and dependency, the average C.P.U. time are 26982.68, 29311.86, 29377.50 and 28478.24 and 28788.18, at 0.0 level of significance. For calculation found that functional programming is the fastest, followed by aggregation, composition, and dependency, the average C.P.U. time are 28478.24 and 28788.18, at 0.0 level of significance. For sorting found that functional programming is the fastest, followed by dependency, composition, and aggregation, the average C.P.U. time are 23785.88, 27449.76, 28478.24 and 28788.18, at 0.0 level of significance. For the Message variable. The functional programming is the fastest, followed by aggregation, composition, and dependency, the average C.P.U. time are 13566.60, 17891.70, 18532.66 and 19336.76, at 0.0 level of significance.

II. RELATED LITERATURE

A. UML

Unified Modeling Language (UML) is the language of identifying, specifying, and documenting system and software artifacts. Therefore, the class diagram and relationships are the main focus of this study.

B. Dependency

Webster’s Dictionary defines dependency as something dependent on something else. In UML, dependence represents a relationship between a client and a supplier. In which a change in the specifications of the supplier may affect the client. It represents a line of dashes (----), and the arrow provides the direction. For instance, if A is dependent on B (A --- > B) and then if B changes, A has to change too. In UML, Class A is dependent on Class B, as represented in Fig. I.

C. Association

Associations represent structural relationships between objects of different classes or information that must preserve for some duration of time. It is not only characterized by procedural dependency relationships. It represents a line ( |— | ), and the arrow indicates the direction.

II. RELATED LITERATURE

A. UML

Unified Modeling Language (UML) is the language of identifying, specifying, and documenting system and software artifacts. Therefore, the class diagram and relationships are the main focus of this study.

B. Dependency

Webster’s Dictionary defines dependency as something dependent on something else. In UML, dependence represents a relationship between a client and a supplier. In which a change in the specifications of the supplier may affect the client. It represents a line of dashes (----), and the arrow provides the direction. For instance, if A is dependent on B (A --- > B) and then if B changes, A has to change too.

In UML, Class A is dependent on Class B, as represented in Fig. I.

Fig. 1: Dependency relationship

C. Association

Associations represent structural relationships between objects of different classes or information that must preserve for some duration of time. It is not only characterized by procedural dependency relationships. It represents a line ( |— | ), and the arrow indicates the direction.

Fig. 2: Association relationship

One object has a permanent association with another object. There are two categories of associations: aggregation and composition. An aggregation is a stronger form of a relationship, in which a relationship exists between itself and its parts. The aggregate has an aggregation association with its constituent parts. A hollow diamond was attached to the end of an association path on the side of the aggregate (the whole ) to indicate aggregation. A composition is a form of aggregation with strong ownership and coincident lifetimes on the part of the aggregate. When the whole is removed, the part is also removed. The part may be removed (by the whole) before the whole is removed. A solid filled diamond ( ) was attached to the end of an association path (on the “whole side”) to indicate composition.
D. Interchangeable relationships in real-world situations

As previously mentioned, in real-world situations, connections can be interchangeable. For instance, A and B, have a friendly relationship. After a while, A and B develop a husband-wife relationship, as shown in Fig 3.

![Fig. 3: Interchangeable relationship](image)

E. Composition and Aggregation in Coding

In Object-Oriented Analysis and Design, an association relationship consists of composition and aggregation. The composition is a strong type of relationship. For instance, A composed of B, which means that A could not exist if B did not exist. In coding language, it is as follows:

```java
Class A {
    A () ;
    B _ref = new B ( );
    ...
}
```

In terms of aggregation, this relationship is not strong. Therefore Class A could exist even though Class B does not exist. The code should be:

```java
Class A {
    A () ;
    B _ref ;
}
```

F. Analysis of Variance

The ANOVA or Analysis of Variance is a statistical tool used to identify the mean and the differences between three or more sample groups. The ANOVA computed by using the formula below.

\[
F = \frac{\text{Mean Square Between}}{\text{Mean Square Within}} = \frac{MSG}{MSB}
\]

III. CASE STUDY

This research created four situations and the execution of each relationship; the first sent the message, the second showed the string, the third calculation, and the last was sorting. An example of the output of execution illustrated in Fig 4.

A. Dependency class diagram

The researcher created three classes; Basic, Dependency, and Code. The basic class is the main class and includes strings, messages, sorting, and calculation. The Dependency class demonstrated the relationship between Basic and Dependency. The Coding class was used to perform the execution of the program.

![Fig. 4: Execution output](image)

B. Aggregation class diagram

The result is similar to that found in Fig. 5. There were three classes created for the purpose of this research, known as Basic, Aggregation, and Code. The function and Code classes executed the same tasks as the participants in Fig. 5. The only change was the Aggregation class, which showed the relationship between the Aggregation class and functional programming.

![Fig. 5: Dependency class diagram](image)

C. Composition class diagram

In the previous example, the researcher replaced the Aggregation class with the Composition class. It changed the relationships in the Composition class while the rest remained the same.

![Fig. 6: Aggregation class diagram](image)
Fig. 7: Composition class diagram

D. Normal class diagram (Functional Program)

In the last case, the researcher moved all of the functionality in terms of dependency, aggregation, and composition into the main function and created none object-oriented programming.

Fig. 8: None objected oriented programming

IV. RESEARCH FINDINGS

From the executable program, the timestamps recorded the execution before and after of each variable type in all cases.

A. Message execution time

The computation of the messages indicated a relationship between the Aggregation, Composition, Dependency, and functional show in Table 1.

Table 1: Message execution time

<table>
<thead>
<tr>
<th>Descripition</th>
<th>N</th>
<th>Mean</th>
<th>Std. Deviation</th>
<th>Std. Error</th>
<th>Minimum</th>
<th>Maximum</th>
</tr>
</thead>
<tbody>
<tr>
<td>Aggregation</td>
<td>50</td>
<td>13881.70</td>
<td>11777.75</td>
<td>158.076</td>
<td>15000</td>
<td>20012</td>
</tr>
<tr>
<td>Composition</td>
<td>50</td>
<td>19522.60</td>
<td>1864.41</td>
<td>185.129</td>
<td>14737</td>
<td>21935</td>
</tr>
<tr>
<td>Dependency</td>
<td>50</td>
<td>19305.74</td>
<td>1211.33</td>
<td>171.317</td>
<td>18950</td>
<td>27244</td>
</tr>
<tr>
<td>Functional</td>
<td>50</td>
<td>13556.66</td>
<td>1254.36</td>
<td>177.358</td>
<td>13716</td>
<td>22052</td>
</tr>
<tr>
<td>Total</td>
<td>200</td>
<td>17311.80</td>
<td>2057.61</td>
<td>180.872</td>
<td>13176</td>
<td>27244</td>
</tr>
</tbody>
</table>

The results revealed that Aggregation was the fastest class (17891.70), followed by Composition (18532.66) and Dependency (19336.76). The functional program did not include because it was not relevant in terms of computation. In real-life situations, developers could not write all of the code, only in the main function.

B. String execution time

Concerning string execution times, the results demonstrated that Dependency was the fastest (27449.76), followed by Composition (28478.24) and Aggregation (28788.19). The dependency relationship was the fastest, perhaps because the other two relationships took time to create the object.

Table 2: String execution time

<table>
<thead>
<tr>
<th>Descripition</th>
<th>N</th>
<th>Mean</th>
<th>Std. Deviation</th>
<th>Std. Error</th>
<th>Minimum</th>
<th>Maximum</th>
</tr>
</thead>
<tbody>
<tr>
<td>Aggregation</td>
<td>50</td>
<td>22879.74</td>
<td>5411.47</td>
<td>765.338</td>
<td>21644</td>
<td>37070</td>
</tr>
<tr>
<td>Composition</td>
<td>50</td>
<td>28478.24</td>
<td>2506.578</td>
<td>250.844</td>
<td>24743</td>
<td>35354</td>
</tr>
<tr>
<td>Dependency</td>
<td>50</td>
<td>27449.76</td>
<td>3092.166</td>
<td>439.940</td>
<td>21197</td>
<td>29983</td>
</tr>
<tr>
<td>Functional</td>
<td>50</td>
<td>22825.88</td>
<td>1829.820</td>
<td>272.974</td>
<td>21359</td>
<td>39041</td>
</tr>
<tr>
<td>Total</td>
<td>200</td>
<td>27315.29</td>
<td>3935.955</td>
<td>278.396</td>
<td>21197</td>
<td>37070</td>
</tr>
</tbody>
</table>

C. Calculation execution time

Aggregation (29311.86) was the fastest class in terms of calculation, followed by Composition (29377.50), and finally, Dependency (29397.30). The results are shown below, as follows.

Table 3: Calculation execution time

<table>
<thead>
<tr>
<th>Descripition</th>
<th>N</th>
<th>Mean</th>
<th>Std. Deviation</th>
<th>Std. Error</th>
<th>Minimum</th>
<th>Maximum</th>
</tr>
</thead>
<tbody>
<tr>
<td>Aggregation</td>
<td>50</td>
<td>29311.86</td>
<td>973.713</td>
<td>137.704</td>
<td>26239</td>
<td>30137</td>
</tr>
<tr>
<td>Composition</td>
<td>50</td>
<td>29377.50</td>
<td>11777.717</td>
<td>168.053</td>
<td>24987</td>
<td>30715</td>
</tr>
<tr>
<td>Dependency</td>
<td>50</td>
<td>29397.30</td>
<td>986.654</td>
<td>139.120</td>
<td>26224</td>
<td>30225</td>
</tr>
<tr>
<td>Functional</td>
<td>50</td>
<td>29369.68</td>
<td>71.335</td>
<td>10.088</td>
<td>26902</td>
<td>37300</td>
</tr>
<tr>
<td>Total</td>
<td>200</td>
<td>29675.34</td>
<td>1370.818</td>
<td>96.931</td>
<td>24617</td>
<td>30715</td>
</tr>
</tbody>
</table>

D. Sort execution time

In terms of sorting, the aggregation (18408.36) relationship was the fastest way to execute, followed by composition (21641.68) and dependency (22861.14). The results show in Table 4.

Table 4: Sorting execution time

<table>
<thead>
<tr>
<th>Descripition</th>
<th>N</th>
<th>Mean</th>
<th>Std. Deviation</th>
<th>Std. Error</th>
<th>Minimum</th>
<th>Maximum</th>
</tr>
</thead>
<tbody>
<tr>
<td>Aggregation</td>
<td>50</td>
<td>18408.36</td>
<td>1276.899</td>
<td>185.595</td>
<td>10906</td>
<td>25067</td>
</tr>
<tr>
<td>Composition</td>
<td>50</td>
<td>21641.68</td>
<td>2497.214</td>
<td>353.198</td>
<td>18950</td>
<td>27244</td>
</tr>
<tr>
<td>Dependency</td>
<td>50</td>
<td>22984.14</td>
<td>1542.002</td>
<td>215.529</td>
<td>20514</td>
<td>36820</td>
</tr>
<tr>
<td>Functional</td>
<td>50</td>
<td>19725.20</td>
<td>1836.965</td>
<td>257.785</td>
<td>16317</td>
<td>26183</td>
</tr>
<tr>
<td>Total</td>
<td>200</td>
<td>20209.10</td>
<td>2782.520</td>
<td>195.625</td>
<td>16317</td>
<td>36820</td>
</tr>
</tbody>
</table>

V. CONCLUSION

An Analysis of Variance was employed to compare the average calculation times. The results are shown below, as follows:

Table 5: ANOVA of Aggregation, Composition, and Dependency

<table>
<thead>
<tr>
<th>ANOVA</th>
<th>df</th>
<th>Mean Square</th>
<th>F</th>
<th>Sig.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Aggregation Between Groups</td>
<td>3</td>
<td>332535665.4</td>
<td>21.4091</td>
<td>.000</td>
</tr>
<tr>
<td>Within Groups</td>
<td>196</td>
<td>1553347.92</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Total</td>
<td>199</td>
<td>15788393.02</td>
<td></td>
<td></td>
</tr>
<tr>
<td>String Between Groups</td>
<td>3</td>
<td>26821076.6</td>
<td>22.617</td>
<td>.000</td>
</tr>
<tr>
<td>Within Groups</td>
<td>196</td>
<td>11681686.50</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Total</td>
<td>199</td>
<td>181517052.60</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Calculation Between Groups</td>
<td>3</td>
<td>70843492.88</td>
<td>85.023</td>
<td>.000</td>
</tr>
<tr>
<td>Within Groups</td>
<td>196</td>
<td>6220553.39</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Total</td>
<td>199</td>
<td>127391277.19</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Sort Between Groups</td>
<td>3</td>
<td>292457768.3</td>
<td>86.232</td>
<td>.000</td>
</tr>
<tr>
<td>Within Groups</td>
<td>196</td>
<td>3950844.318</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Total</td>
<td>199</td>
<td>334463029.48</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 5 indicates a significant difference regarding the average execution times of messages toward aggregation, composition, and dependency. The more effective way to design the system to execute messages is the use of aggregate, composition, and dependency.

There were significant differences between the average string execution time of concerning aggregation, composition, and dependency. Therefore, the designer should use dependency, composition, and aggregation.
There are significant differences in the average execution time of calculation concerning aggregation, composition, and dependency. The best way to design the system to execute calculation was also to utilize aggregation, composition, and dependency.

There were also significant differences in terms of the average execution time of sorting toward aggregation, composition, and dependency. The best way to design the system to execute calculation was to employ aggregation, composition, and dependency.

Therefore, functional programming was the fastest but not relevant for this study. The designers and developers should not just consider real-life situations but also the fastest way to develop computer systems.

ACKNOWLEDGMENT

This work was supported financially by King Mongkut’s University of Technology, North Bangkok, Grant Number: KMUTNB-61-GOV-A-08.

The researchers are thankful to the anonymous referees for their valuable suggestions.

REFERENCES


AUTHORS PROFILE

Assoc. Prof. Ngamsantivong Thavatchai received a BA. in Teaching Mathematics from Chulalongkorn University, Thailand and a Master’s Degree in Educational Technology from Technological University of the Philippines. During newly graduated was sent trained on computerized information systems in many countries such as West-Germany, Italy, Japan, and Australia. The author of many textbooks in Thai such as UML, OOP, SPSS. At present research on IoT, Fuzzy logic, and software engineering.

Dr. Rantanavilisagul Chaibwoot received a Ph.D. in Electrical Engineering from King mongkut’s institute of technology ladkrahang. I research in field Artificial Intelligence, Natural language processing, Machine learning, Searching Technique, Web Programming, Computer Network, Multimedia, Animation, and the author of “Optimization and Searching Algorithm”.

Retrieval Number:J107108810S19/2019©BEIESP
DOI: 10.35940/ijitee.J1071.08810S19
Published By:
Blue Eyes Intelligence Engineering & Sciences Publication