UML Class Diagram Abstraction According to Page Rank Algorithm

Abstract

1. **Introduction**

In this paper, we propose a method to abstracted class diagrams and get the big picture of a class diagram as Figure 1 shows. We map a class diagram to a graph with nodes and edges. Nodes in the graph stand for classes in the class diagram and edges stand for relationships. Then according to the graph we do the following operations:

1. Compute the ranks of nodes: Nodes stand for classes. A rank of a class stands for the importance of the class in a class diagram. If the rank of a class is higher then we can say the class is more important in a class diagram. In this paper, we use a algorithm which is similar to Page Rank to calculate class ranks. Then, we get a one dimensional vector R.
2. Compute the correlations between nodes: A correlation between two nodes is used to judge whether a strong association was existed between the two nodes. If there exists strong association between two nodes we can use a line to connect them. The calculation of correlation is according to the distance between two nodes. Then, we get a weighted graph G(V,E1). E1 is edge weighted with correlation.
3. Compute the relationships between nodes: Nodes stand for classes. If there is a path between two classes it may exist relationships between them. We use a method called relationship abstraction to get the direct relationship between two classes. Then, we get a weighted graph G(V,E2). E2 is edge weighted with relationship.

Finally, according to G(V,E1), G(V,E2) and R we get a new graph G1(V,E) which is the “big picture” of the original graph G(V,E). At last, we get the abstracted class diagram according to graph G(V,E). The abstracted class diagram is the “big picture” of the original class diagram.

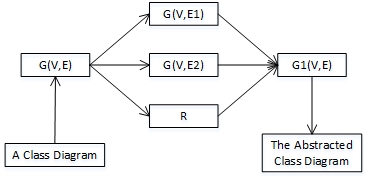
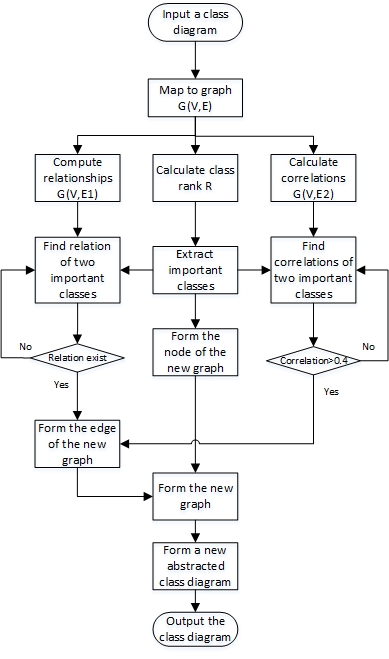


Figure 1. General steps of class diagram abstraction

Figure 2 shows the process of a class diagram abstraction. It generates the abstracted class diagram of a original class diagram. The abstracted class diagram is a big picture of original class diagram.

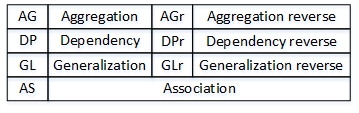
1. We input a class diagram
2. We map the class diagram to a graph G(V,E). The nodes of the graph stand for the classes and the edges stand for relationships in class diagram.
3. According to G(V,E) we compute relationships and correlations between classes. We also calculate ranks of classes.
4. According to ranks of classes we extract important classes. Then we find correlations of any two important classes in G(V,E2) and relationships in G(V,E1).
5. We create a line to connect two important classes and the type of the line is the relationship between them if the correlation between them is higher than 0.4 and the relationship between them is existed.
6. We repeat the step (5) until any two of important classes are connected. Then we get a new graph with only important classes and relationships between them. It is the “big picture” of the original graph.
7. Finally, we generate a new class diagram according to the new graph. The new class diagram is the “big picture” of the original class diagram and it is the abstraction of the original one.

Figure 2. The process of class diagram abstraction

1. **Calculate Ranks of Classes**

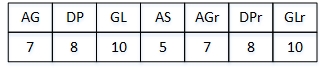
**2.1. Definitions**

Figure 3 shows the shorthand of relationships. For example AG is short of aggregation. An expression like A x AG x B stands for A aggregate B. The expression like A x AGr x B means that B aggregate A.

Figure 3. Shorthand of relationships

**Class Rank (Rc):** It is a float value between [0,1000]. It stands for the importance of a class in a class diagram. A class is more important if the class rank of it is higher.

**Relationship Rank (Rr):** It is a integer value between [0,10]. It stands for the importance of a relationship. For example, the importance of generalization is larger than association. We set different relationship ranks for different relationships as Figure 4 shows.

Figure 4. Ranks of relationships

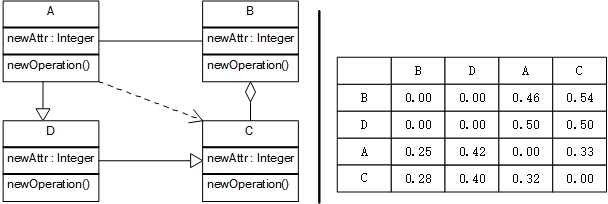
**Transition probability (TP):** It means the probability of the rank transit from one class to another.

TP(i,j) is the rank transition probability from class i to class j. Rr(i,j) is the relationship rank of edge (i,j). For example, Rr(i,j) equals to 7 if the relationship of edge (i,j) is AG.



**Transition Probability matrix:** It is a matrix. It represents the transition probability of class ranks from one class to another. For example, the left of Figure 5 shows a simple class diagram and the right of it shows the probability transition matrix of the class diagram. The transition probability of class rank from class A to class C is 0.33. Rr(A,B) equals to 5 because the relationship between class A and B is AS. Rr(A,C) equals to 8 because the relationship between class A and C is DP. Rr(A,D) equals to 10 because the relationship between class A and D is GL. So TP(A,C) equals to 0.33.



Figure 5. A simple class diagram and it’s transition matrix

**2.2. Calculate Class Ranks**

We calculate the rank of a class according to the rank and transition probability of its adjacent classes.

The calculation process of class ranks is as following shows:



**(1) Set initial matrix of class rank**: Take Figure 5 as an example. There are 4 classes in the class diagram. So the initial rank of every class equals to 1000/4=250. Then, the initial matrix of class rank equals to R1={250,250,250,250}.

**(2) Calculate the transition probability matrix (M):** The right of Figure 5 shows the transition probability matrix of the simple class diagram example in Figure 5. The calculation of transition probability matrix is according to transition probability of classes in a class diagram.

**(3) iteration:** In each iterative process, we use the last class rank matrix multiply the transition probability matrix and get a new class rank matrix.

For example, in Figure 5, R2=R1\*M={132,204,321,343}.



1. **Skip out of current iteration:**



In the iteration process of time i, if every class rank element in rank matrix Ri meets the following condition we skip out of current iteration and get the final class rank matrix Ri.

Ri[k] is the rank of class K in the iteration process of time i. Ri-1[k] is the rank of class K in the iteration process of time i-1.



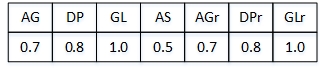
In the example of Figure 5, R11={158,243,294,305} and R10={159,245,292,304}. Every rank element in R11 meets the condition so R11 is the final class rank matrix of the class diagram in Figure 5.

**3. Compute Correlations of Classes**

**3.1. Definition**

**Correlation of two classes:** A correlation of two classes is decided by the paths between the two classes. We can use a line to connect two classes if the correlation of them is high.

**Strength of a relationship:** The strength of a relationship is similar to relationship rank. It is used to calculate the correlation of two class. Figure 6 shows the strength of each type of relationship. For example , the correlation of class A and B is 0.5 if class A associate with class B. However, the correlation of class A and B is 0.8 if class A depend on class B.

Figure 6. Strength of relationships

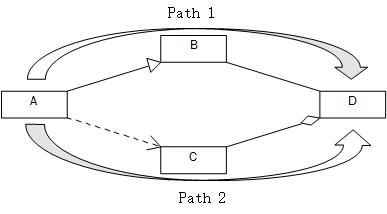
**3.2. Calculate correlation of two classes**

We need to compute the correlation if two classes is not directly connected with each other. For example, Figure 7shows a path between class A and class D. In this path, class A and D are not directly connected with each other. There are middle class B and C between them. We calculate the correlation of class A and D as following shows:

1. we get the correlation of class A and B is 0.5 because the relationship between class A and B is AS, the correlation of class B and C is 0.7 because the relationship between class B and C is AG, the correlation of class C and D is 1.0 because the relationship between class C and D is GL.
2. We get the correlation of class A and D equals C(A,D)=0.5\*0.7\*1.0=0.35

onepathcorrelationFigure 7. Calculate correlation of two classes in a path

We need to calculate the correlation of two classes in each path if there exists more than one path between the two classes. For example, Figure 8 show two paths between class A and D. For Path 1, we calculate the correlation between class A and D equals to C1(A,D)=1.0\*0.5=0.5. For Path 2, we calculate the correlation between class A and D equals to C2(A,D)=0.8\*0.7=0.56. We choose C2(A,D) as the final correlation of class A and D because C2(A,D) is higher than C1(A,D).

Figure 8. A example contains more than one path between two classes

C(i,j) is the correlation of class i and j. Path(n) is a path between class i and j. K is a relationship in the Path(n) such as AG, AS. S(k) is the strength of the relationship k.



**3.3. Calculate the correlation matrix of a class diagram**

For any two classes in a class diagram, we calculate the correlation of them and get the correlation matrix of the class diagram. Then we get the graph G(V,E1) of the class diagram. E1 are edges with weighted correlations.

**4. Compute Relationships of Classes**

**5. A Example**

**6. Related Works**

Our work is based on a lot of previous research findings such as UML abstraction rules, programming techniques, information retrieval, software visualization and so on. We focus on UML class diagram abstraction, software analysis, software visualization and information retrieval.

1. . UML Class Diagram Abstraction. UML class diagram is a kind of structural model of a system. It can be seen as an abstraction of the system. The class diagram will be very large when a system is complex. Designers can easily become overwhelmed with details when dealing with large class diagrams. So the abstraction of large class diagram is need. The abstraction of large class diagrams mean refining class diagrams. UML class diagram abstraction transforms a low-level class diagram to a high-level class diagram[1].

Eyged . A find a series of abstraction rules for class diagram abstraction which contains class abstraction and relationship abstraction. The relationship between any two class in a class diagram can be calculated by using the relationship abstraction rules[2]. The abstracted class can be calculated by class abstraction rules. His way of UML class diagram abstraction need to calculate all the relationships between any two classes in a class diagram. The complexity of it is very costly because the computational complexity of pattern matching is known to be very costly[3]. Our work use the page rank algorithm to calculate the rank of class diagram and get the most important classes, then we do the relationship abstraction for the most important classes.

1. . UML Class Diagram Layout. UML class diagram layout belongs to graph drawing. How to present the UML class diagram to makes it more easy to read is very important. A good layout of a class diagram can make it more easy for designers to read. The mostly used drawing method is hierarchical drawing algorithm proposed by Sugiyama. For UML class diagram, it uses a hierarchical way to lay out the classes[6]. Seemann J. extended the Sugiyama algorithm for drawing UML class diagrams[4]. Another layout method is rank-direct layout method [7]which centered on higher ranked classes and clustered lower ranked classed to higher ranked classes.

Our work also center on higher ranked classes but we use the relationship abstraction to connect these higher ranked classes and ignore lower ranked classes. We presents a higher abstracted class diagram layout.

1. . Software Visualization. Software visualization is important because most software artifacts are invisible[10]. With the increasingly complexity of a software system it is more and more difficult for people to program, understand, and modify the software[11]. UML is the mostly used method to visualize a software. UML contains behavioral diagrams (activity diagram, sequence diagram, state machine and so on) and structural diagrams (class diagram, package diagram, component diagram, use case diagram and so on). Our work is based on UML and we center on the abstraction of a class diagram and help developers understand the class diagram design of a software system easily.
2. . Software Analysis.
3. . UML Diagram Reading. UML diagram reading refers to how modelers read UML diagram. By researching modelers’ habits of reading UML diagrams researchers can propose more readable diagram layouts. Harald Storrle. et. al. find that various factors like layout quality, modeler experience, and diagram type lead to significant differences in diagram reading strategies[8,9]. Helen C. Purchase. et. al. investigated the preference of UML modelers and fond that joined inheritance arcs and directional indicators are preferred for class diagrams[5]. Our work presents a highly abstracted class diagram for readers. The highly abstracted class diagram contains the least classes. It can stand for the original class diagram because readers can infer other classes and main function of original class diagram from it.

7. Conclusions

Reference

[1] Egyed A. Automated abstraction of class diagrams[J]. ACM Transactions on Software Engineering and Methodology (TOSEM), 2002, 11(4): 449-491.

[2] Egyed A. Semantic abstraction rules for class diagrams[C]//Automated Software Engineering, 2000. Proceedings ASE 2000. The Fifteenth IEEE International Conference on. IEEE, 2000: 301-304.

[3] Fahmy H, Holt R C. Software architecture transformations[C]//Software Maintenance, 2000. Proceedings. International Conference on. IEEE, 2000: 88-96.

[4] Seemann J. Extending the Sugiyama algorithm for drawing UML class diagrams: Towards automatic layout of object-oriented software diagrams[C]//Graph Drawing. Springer Berlin Heidelberg, 1997: 415-424.

[5] Purchase H C, Allder J A, Carrington D A. Graph layout aesthetics in UML diagrams: user preferences[J]. J. Graph Algorithms Appl., 2002, 6(3): 255-279.

[6] Sugiyama K, Tagawa S, Toda M. Methods for visual understanding of hierarchical system structures[J]. Systems, Man and Cybernetics, IEEE Transactions on, 1981, 11(2): 109-125.

[7] Hu H, Fang J, Lu Z, et al. Rank-directed layout of uml class diagrams[C]//Proceedings of the First International Workshop on Software Mining. ACM, 2012: 25-31.

[8] Störrle H, Baltsen N, Christoffersen H, et al. On the Impact of Diagram Layout: How Are Models Actually Read?[C]//International Conference on Model Driven Engineering Languages and Systems (MoDELS) 2014. 2014: 31-35.

[9] Störrle H. On the impact of layout quality to understanding UML diagrams[C]//2011 IEEE Symposium on Visual Languages and Human-Centric Computing. 2011: 135-142.

[10] Ball T, Eick S G. Software visualization in the large[J]. Computer, 1996, 29(4): 33-43.

[11] Bassi S, Keller R K. Software visualization tools: Survey and analysis[C]//Program Comprehension, 2001. IWPC 2001. Proceedings. 9th International Workshop on. IEEE, 2001: 7-17.