Result and evaluation

A. Evaluation of the results of two cases

We need to validate the results of the two cases described above. The cases we use come from real practice, and we can take the actual module we divide as consideration. To make it more reasonable and formal, we create a checklist as follows and give a score to the result per question:

1. Does the services in the result can fulfill the whole system? (Functionality)

All(10), more than 70%(5), others(0)

2. Does the entities in one service are related to each other tightly? (Cohesiveness)

Exactly one module(10), more than 70% are should together(5), others(0)

3. Does the entities not in one service have close relations? (Compatibility)

Less than 10% relations(10), Less than 30% relations(5), others(0)

4. Too many services? (Granularity) Yes(0), No(10)

5. Too few services? (Granularity) Yes(0), No(10)

In order to give a general evaluation to the results, we define 3 categories according to the checklist and our experience.

1. excellent (average score>=9)

2. reasonable (6<=average score<9)

3. unreasonable (average score<6)

From discussion above, we can see that the result of the first case is excellent () while the second case is just reasonable () but a little different from the best result we expect.

B. compared to Service-Cutter’s method[]

We also compare our method with use the method in Service-Cutter[]. Since our methods are based on different models, we should build another model based on Entity Relation Model and other user representations according to their criteria. So, we convert our two cases to Service-Cutter’s input format and get the decomposition results as follows:

By using our checklist, the results compared with ours are showed below:

|  |  |  |
| --- | --- | --- |
|  | Case 1 | Case 2 |
| Our method | Excellent(9) | Reasonable(7) |
| Service Cutter | Unreasonable(3) | Reasonable(6) |

We can see our methods’ advantage over theirs in these two cases. To detail, the result of the first case using their method divide all entities to just one service which obviously is unreasonable. // add more detail

// discuss complexity problems?

In addition, I want to discuss the process when using these two methods, our method based on the Data Flow Diagram and it well described the procedure of the real system. However, the method used by Service Cutter is based on Entity Relation Model and many other complex criteria which make the modeling process more complex and not intuitionistic.