Comparative Study of Back-End Vs Front-End System by Performance Analysis during Preliminary Design stages

S. Jagannatha, T.V Suresh Kumar, R. RajaniKanth

Department of Computer Applications,
M.S. Ramaiah Institute of Technology, Bangalore (INDIA)
jagannatha@msrit.edu, sureshkumar@msrit.edu, rajankanth@msrit.edu

Abstract

Performance is an important non-functional attribute of the software system to be considered for producing quality software. Effort estimation helps in the calculation of size of the software and database separately, which is one of the required data in assessing the performance. We use Entity Relationship(ER) and Use case mode for compute the data base effort and software effort respectively. In this chapter we proposed a methodology for performance analysis of back-end and front-end applications by estimation of effort and compute response time. We propose the comparative study of response time and validate the results.

1. Introduction

The effort estimation is the process of estimating the total effort required to complete a software project. Cost estimation is one of the key aspects of project management based on which performance goals can be achieved. The effort estimation depends on the complexity of software, ER diagram, data requirements. Performance problems may be so strict that they require extensive changes to the system architecture in Preliminary design stages. If these changes are made late in the development process, they can increase development costs, delay deployment, or adversely affect other desirable qualities of a design. It is vital to maintain for early assessment of the performance characteristics of distributed database system since its functionality is decentralized.

The data are required for the database effort also depends on the complexity of entities, attributes, relationship type and structural constraints. Predicting these factors reflect the level of complexity and predict in terms of number of lines of codes required for implement the business constraints. The data required for estimation for front end application based on functionality which represented in usecase diagram. The functionality The Use case Points estimation methods introduced in 1993 by Karner estimates effort in person hours based on use cases. This use case point method classified actors and use cases into three categories as simple, average and complex and assigned different weight factors to actors based on their interaction with the system, like using defined application user interface. Similarly the weight factors assigned to use cases based on the number of transactions. Then they calculate the unadjusted use case point (UUCP) by adding unadjusted actor weights (UAW) and unadjusted use case weights (UUW). After this they assign values to the technical and environmental factors multiply by their weights and calculated the weighted sums for generating technical complexity factor (TCF) and environmental factors (EF) for the final estimation of adjusted use case points (UCP) by using formula UCP=UUCP * TCF * EF . Finally the effort was estimated by multiplying UCP with person-hours needed to implement each use case point [1].

Performance is an important issue for a software system. Even several development methodologies such as performance considerations in hardware industries, system software and networking environment and application programming. Performance evaluation system must address performance user point i.e. response time of the software solutions.

Software Performance Engineering (SPE) is a method to predict the performance of software systems early (analysis phase) in the life cycle. SPE continues through the detailed design, coding and testing stages to predict and manage the performance of the evolving software and to monitor, report actual performance against specifications and predictions. Modelling software systems to predict the performance using Unified Modelling Language (UML).

The rest of the paper is organized as follows: Review of the related literature is given in Section 2. Section 3 discusses the basic concepts involved in this paper. The proposed methodology is described with an algorithm in Section 4. The illustration of proposed technique with a case study is presented and the results
are validated in section 5. The paper concludes with observations and the future work in section 6

2. Literature Survey.
Many researchers are made significant contribution in the field of software effort estimation using in the field of software development. In [1][4][5] estimation technique based on the function-oriented software development frame work. ER model for the estimation of backend effort cost has been proposed [9][11][20][38].

Research results on estimating efforts based on complexity of the development of database part of the software has narrowly been reported in the literature. In [39][40] use case point approach is widely used industry for estimation of software cost. Use case approach mainly based on an actor, compute unadjusted actor weights Calculate unadjusted use case weights, based on use case, Calculate unadjusted use case points consider the assumed rates of technical and environmental factors. Calculate adjusted use case point, compute line of code using gearing factor LOC ← UCP * gearing factor. Convert LOC into number of executable statements, Calculate equivalent size of software in KB. Adjust the size for software resource requests. Solve the use case performance model to obtain response time. The effort estimation using this approach is illustrated and its effectiveness is proved with case studies in [43], [44], [45], [46], [47].

Some authors in [41] used number of primary key and foreign key for effort estimation of ER model using the formula. Many techniques for estimating the cost of the relational database development based on ER model have been reported in the literature. However, these are actually execution details and difficult to estimate during the early stage of software development. The entity relationship (ER) model is well used in conceptual modeling for data base part of software systems. The path complexity metric is a complexity metric and used for effort estimation. This path complexity is computed from a graph derived from the ER diagram. Compute the information about total entities, total relationships, total constraint complexity and based on these information, the total complexity and then the effort can be estimated [1].

Software Performance Engineering has evolved over the years and has been demonstrated to be effective during the development of many large systems [3], [29][30].

3. Propose Methodology
Effort estimation helps in the calculation of size of the software which is one of the required data in assessing the performance of software systems using Entity Relationship (ER) model. [1].

In this chapter we proposed a methodology for performance analysis of DDS is as follows:

i. Procedure to estimation of effort is based on the complexity of ER model and compute response time. We consider the Technical and Environment factor for computing adjusted ER Point (ERP) for effort estimation. Computing Lines of Code (LOC) from the estimated effort and compute response time and the resource requirement of each of the hardware device for the scenario by considering the size is adjusted as a typical workload for the software resource.

ii. Estimate effort based on the use case diagram by computing the UCP. Computing Lines of Code (LOC) from the estimated effort in step i, and compute response time and the resource requirement of each of the hardware device for the scenario by considering the size is adjusted as a typical workload for the software resource using the smith approach.

4. Algorithm
The algorithm to calculate the response time is as follows:

1. Consider the ER model for the DDs application under assessment. Calculate Compute adjusted ER Point (ERP) using the widely used formula ERP = TC×TCF×EF using ER model approach [1].

2. Estimate The size by using equation Size in KB = (ERP*NLOC*NW*Wlength)/(1024*28) , Where ERP = is the adjusted ER point obtained in step 1, NLOC=Number of Lines of Code per ER point NW= Number of words required per instruction, Wlength=Word length in bits.

3. Calculate the response time using the resource overhead matrix given in table 1.

4. Consider the Use case model of DDA applications. Compute UCP=UUCP * TCF * EF. Where UUCP is unadjusted use case point, TCF is the technical factor and EF is the environment factor.

5. Compute the KB = The size by using equation Size in KB = (UCP *NLOC*NW*Wlength)/(1024*28) , Where ERP = is the adjusted ER point obtained in step 1, NLOC=Number of Lines of Code per ER point NW= Number of words required per instruction, Wlength=Word length in bits.
6. Calculate the response time using the resource overhead matrix given in table 1.
7. Compare the response time for back end and front applications.

5. Illustrate the Problem with the Case Study.

5.1 ER Model Approach
In this chapter we consider a case study of a Company Database System. The company organized into departments and have several locations and implements several projects. Employees in each department supervise the projects. Company keeps track of all dependents for insurance purpose. We use Entity Relationship (ER) diagram for effort estimation which is presented in the Figure 1 the use case diagram represented in the figure 2. We consider eight different projects having different number of constraints. We estimate the performance analysis by computing the response time and the results are validated with the proven results of Connie U. Smith [3].

5.2 Over Head matrix
Processing Time of each parameter in each resource
Table 1 Processing Resource Overhead Matrix

| Input(Per KB) | 0.002 | 1 | 0.25 | 1 |
| DB Access(Per Access) | 0.0005 | 0.25 | 1 |
| LocalDB (Per KB) | 0.01 | 2 |
| Page size(Per KB) | 0.0005 | 1 |
| Data size(Per KB) | 0.0005 | 1 |
| Service time in Sec | 1 | 0.003 | 0.1422 | 1 | 0.0016 |

The overhead matrix in Table 1 shows the resources that are considered for the processing with their processing time. Table 1 shows that are 4 CPUs, 3 disks, one network, one delay and there is one delay the service units of the devices are mentioned in the service units row. The input size is dividing into 5 parameters input data - the number of Kbytes in the input messages. DB Access – the number of access to the database, Local DB – the number of access to the local database, Page size – the number of Kbytes in the page and Data size - the number of Kbytes in data retrieved.

5.3 Performance Analysis
The ER model of Company Database System is shown in Figure 1 The ER point of the Company Database System is shown in table 2 in the second row. We have considered eight different projects having different number of constraints and ER point (ERP) and estimated effort with actual effort is shown in table 2.
### Table 2: Computation of ERP versus Actual Effort

<table>
<thead>
<tr>
<th>PR NO</th>
<th>TEC</th>
<th>TRC</th>
<th>TSCC</th>
<th>TC</th>
<th>TFACTOR</th>
<th>TCF</th>
<th>Efactor</th>
<th>EF</th>
<th>ERP</th>
<th>Estimated Effort</th>
<th>Actual effort</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>28</td>
<td>5</td>
<td>7.5</td>
<td>40.5</td>
<td>47</td>
<td>1.07</td>
<td>28</td>
<td>0.56</td>
<td>24.26</td>
<td>24.26</td>
<td>28</td>
</tr>
<tr>
<td>2</td>
<td>23.5</td>
<td>4</td>
<td>7.5</td>
<td>35</td>
<td>28</td>
<td>0.88</td>
<td>28</td>
<td>0.56</td>
<td>17.248</td>
<td>17.248</td>
<td>25</td>
</tr>
<tr>
<td>3</td>
<td>26</td>
<td>5</td>
<td>7.5</td>
<td>38.5</td>
<td>42</td>
<td>1.025</td>
<td>28</td>
<td>0.56</td>
<td>22.099</td>
<td>22.099</td>
<td>27</td>
</tr>
<tr>
<td>4</td>
<td>10</td>
<td>7</td>
<td>6</td>
<td>23</td>
<td>25</td>
<td>0.85</td>
<td>28</td>
<td>0.56</td>
<td>10.948</td>
<td>10.948</td>
<td>17</td>
</tr>
<tr>
<td>5</td>
<td>16.5</td>
<td>3</td>
<td>7.5</td>
<td>27</td>
<td>37.5</td>
<td>0.97</td>
<td>28</td>
<td>0.56</td>
<td>14.664</td>
<td>14.664</td>
<td>21</td>
</tr>
<tr>
<td>6</td>
<td>14</td>
<td>4</td>
<td>12</td>
<td>30</td>
<td>39</td>
<td>0.99</td>
<td>28</td>
<td>0.56</td>
<td>16.632</td>
<td>16.632</td>
<td>26</td>
</tr>
<tr>
<td>7</td>
<td>16.5</td>
<td>4</td>
<td>12</td>
<td>32.5</td>
<td>42</td>
<td>1.025</td>
<td>28</td>
<td>0.56</td>
<td>18.655</td>
<td>18.655</td>
<td>28</td>
</tr>
<tr>
<td>8</td>
<td>22</td>
<td>3</td>
<td>13.5</td>
<td>38.5</td>
<td>40</td>
<td>1</td>
<td>28</td>
<td>0.56</td>
<td>21.56</td>
<td>21.26</td>
<td>32</td>
</tr>
</tbody>
</table>

### Table 3: Computation of ERP versus Actual Effort

<table>
<thead>
<tr>
<th>Project Number</th>
<th>Actual Effort/ERP</th>
<th>LOC</th>
<th>Size (KB)</th>
<th>Response Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>28</td>
<td>1484</td>
<td>71.232</td>
<td>8.0890</td>
</tr>
<tr>
<td>2</td>
<td>25</td>
<td>1325</td>
<td>63.6</td>
<td>7.7311</td>
</tr>
<tr>
<td>3</td>
<td>27</td>
<td>1431</td>
<td>68.688</td>
<td>9.2956</td>
</tr>
<tr>
<td>4</td>
<td>17</td>
<td>901</td>
<td>43.248</td>
<td>5.2011</td>
</tr>
<tr>
<td>5</td>
<td>21</td>
<td>1113</td>
<td>53.424</td>
<td>7.5773</td>
</tr>
<tr>
<td>6</td>
<td>26</td>
<td>1378</td>
<td>66.144</td>
<td>9.4794</td>
</tr>
<tr>
<td>7</td>
<td>28</td>
<td>1484</td>
<td>71.232</td>
<td>9.6142</td>
</tr>
<tr>
<td>8</td>
<td>32</td>
<td>1696</td>
<td>81.408</td>
<td>11.6631</td>
</tr>
</tbody>
</table>

Average Response Time: 8.5813

### Table 4: Estimation of Response Time with the Actual Effort

<table>
<thead>
<tr>
<th>Project Number</th>
<th>ERP</th>
<th>LOC</th>
<th>Size (KB)</th>
<th>Response Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>24.26</td>
<td>1285.78</td>
<td>61.717</td>
<td>6.7294</td>
</tr>
<tr>
<td>2</td>
<td>17.248</td>
<td>914.144</td>
<td>43.879</td>
<td>4.9132</td>
</tr>
<tr>
<td>3</td>
<td>22.099</td>
<td>1171.247</td>
<td>56.220</td>
<td>7.4883</td>
</tr>
<tr>
<td>4</td>
<td>10.948</td>
<td>580.244</td>
<td>27.852</td>
<td>3.0012</td>
</tr>
<tr>
<td>5</td>
<td>14.664</td>
<td>777.192</td>
<td>37.305</td>
<td>5.2742</td>
</tr>
<tr>
<td>6</td>
<td>16.632</td>
<td>881.496</td>
<td>42.312</td>
<td>6.0742</td>
</tr>
<tr>
<td>7</td>
<td>18.655</td>
<td>988.715</td>
<td>47.458</td>
<td>6.2173</td>
</tr>
<tr>
<td>8</td>
<td>21.56</td>
<td>1142.68</td>
<td>54.849</td>
<td>7.8681</td>
</tr>
</tbody>
</table>

Average Response Time: 5.9457
Where lines of code is estimated is LOC = ERP *53 and the size of code is computed using formula: Size = (ERP * NLOC * NW * Wlength) / (1024 * 28). Where NLOC: number of lines of code per ERP (53),NW: Number of words per instruction (6), Wlength: word length (64 bits). There for the size is computed as follows: Size = (ERP *53*6*64)/(1000*8). We estimated the software size using the ER point approach and found the response time using the algorithm shown above. We considered the response time as performance metric and calculated the response time of company database system. The ER point, size in KB and response time of all eight projects of estimated and actual effort shown in table 3 and table 4 respectively.

5.3 Use Case Approach: Consider the company database system. The use case diagram shown in the figure 2

![Use case model for Company Database System](image)

Estimation of the size of the use cases: we use the following algorithm for the estimation of the size of the use cases. The use case model given in the figure 2 consists of manage department, project implementation, maintain insurance and supervise employees. The algorithm for the proposed methodology is given below. The algorithm uses the procedure for calculating UCP.

- Identify key performance scenarios
  - For all scenarios S do
    - Identify use cases u and actors a
    - Develop use case model U
    - [Calculate unadjusted actor weights]
      - for all actor a ∈ U do
        - if a is simple then
          - asimple ← asimple + 1
        - else if a is average then
          - aaverage ← aaverage + 1
        - else
          - acomplex ← acomplex + 1
        - end if
      - UAW ← asimple*1+aaverage*2+acomplex*3
    - end for
  - Calculate unadjusted use case weights
    - for all use case u ∈ U do
      - if u is simple then
        - usimple ← usimple + 1
      - else if u is average then
        - uaverage ← uaverage + 1
      - else
        - ncomplex ← ncomplex + 1
      - end if
      - UUCW ← usimple*5+uaverage*10+ucomplex*15
    - end for
  - Calculate unadjusted use case points
    - UUCP ← UAW + UUCW
    - for all technical factors tf do
      - assign rate rtf
    - end for
    - TFactor ← \( \sum_{tf=1}^{13} r_{tf} \times w_{tf} \)
    - TCF ← 0.6 + (0.01 * TFactor)
    - [Calculate environmental factor]
      - for all environmental factor ef do
        - assign rate ref
      - end for
    - EFactor ← \( \sum_{ef=1}^{8} r_{ef} \times w_{ef} \)
    - EF ← 1.4 + (-0.03 * EFactor)
    - [Calculate adjusted use case point]
      - UCP ← UUCP*TCF*EF
    - if using gearing factor then
      - LOC ← UCP*gearing factor
    - [Calculate LOC]
      - Convert LOC into number of executable statements
    - Calculate equivalent size of software in KB using the formula
    - Size = (ERP * NLOC * NW * Wlength) / (1024 * 28). Where NLOC: number of lines of code per ERP (53).
    - NW: Number of words per instruction (6), Wlength: word length (64 bits). There for the size is computed as follows: Size = (ERP *53*6*64)/(1000*8).
    - Adjust the size for software resource requests
    - Solve the use case performance model to obtain response time

```java
if a is simple then
  asimple ← asimple + 1
else if a is average then
  aaverage ← aaverage + 1
else
  acomplex ← acomplex + 1
end if
UAW ← asimple*1+aaverage*2+acomplex*3
end for

{Calculate unadjusted use case weights}
for all use case u ∈ U do
  if u is simple then
    usimple ← usimple + 1
  else if u is average then
    uaverage ← uaverage + 1
  else
    ncomplex ← ncomplex + 1
  end if
UUCW ← usimple*5+uaverage*10+ucomplex*15
end for

{Calculate unadjusted use case points}
UUCP ← UAW + UUCW
for all technical factors tf do
  assign rate rtf
end for
TFactor ← \( \sum_{tf=1}^{13} r_{tf} \times w_{tf} \)
TCF ← 0.6 + (0.01 * TFactor)
for all environmental factor ef do
  assign rate ref
end for
EF ← 1.4 + (-0.03 * EFactor)
UCP ← UUCP*TCF*EF
for all technical factors tf do
  assign rate rtf
end for
TFactor ← \( \sum_{tf=1}^{13} r_{tf} \times w_{tf} \)
TCF ← 0.6 + (0.01 * TFactor)
if using gearing factor then
  LOC ← UCP*gearing factor
end for
LOC ← UCP*gear
```
The results shown in the table 5 is the response time of the individual use case and total response time is 152.29 seconds of complete applications for company database system using use case approach. Whereas the response time of using ER model of the same applications is 7.7311 seconds. Hence the ER model gives only used for effort estimation can be considered

### Table 5. Response time for the Company Data base System. (Comparative Study)

<table>
<thead>
<tr>
<th>Use case</th>
<th>Size in KB</th>
<th>ER Model Approach</th>
</tr>
</thead>
<tbody>
<tr>
<td>Manages Dept</td>
<td>30.234</td>
<td></td>
</tr>
<tr>
<td>Implements Projects</td>
<td>54.23</td>
<td></td>
</tr>
<tr>
<td>Maintain Dependents for Insurance</td>
<td>36.871</td>
<td></td>
</tr>
<tr>
<td>Supervise Employees</td>
<td>30.955</td>
<td></td>
</tr>
<tr>
<td>Total Response time</td>
<td>152.29</td>
<td></td>
</tr>
</tbody>
</table>

The results shown in the table 5 is the response time of the individual use case and total response time is 152.29 seconds of complete applications for company database system using use case approach. Whereas the response time of using ER model of the same applications is 7.7311 seconds. Hence the ER model gives only used for effort estimation can be considered

### 6. References


[25] Dorin, B. Petriu, Daniel Amyot, Murray Woodside, and Bo Jiang; ‘Traceability and Evaluation in Scenario Analysis by Use Case Maps’, LNCS,


