Evaluation of the Effect of Functional Similarities on Development Effort

Ozden Ozcan Top  
Middle East Technical University,  
Ankara, Turkey  
ozden@ii.metu.edu.tr

Seckin Tunalilar  
ASELSAN INC.  
Ankara, Turkey  
stunalilar@mgeo.aselsan.com

Onur Demirors  
Assoc. Prof. Dr., Middle East Technical University,  
Ankara, Turkey  
demirors@metu.edu.tr

Abstract

Although functional size is the primary input for effort and cost estimation formulas, the relation between functional size and development effort does not always hold in practice. Calculating the magnitude of functional similarity is perceived as a partial solution to overcome this problem. In this study we applied different approaches to identify the magnitude of functional similarity in five projects. The applicability of these approaches is evaluated by observing the impact of functional similarity on effort and code size of the projects. We also discuss the challenges, difficulties and opportunities faced during the case studies.

1. Introduction

The unrealistic expectations are one of the major reasons for software failures [29] and expectation of a software project is frequently depend on the software size. In most estimation techniques which use the functional size as input, we determine the required effort, cost, and time to complete a software product. That is, by knowing the accurate functional size of a software product and the relation between functional size and development effort it is possible to plan, execute and monitor projects successfully. However, the relation between the functional size and the required effort can not be defined as accurately as we wish by utilizing the concurrent estimation and measurement techniques [9].

Although the functional size of a software product can be measured with current methods, there are still difficulties in measuring the functional size that directly affect the magnitude of effort required to develop software products. One of these difficulties is related with the functional similarity of functional processes.

Functionally similar processes or components can be reused within the same or a new software product. Reuse is named “internal”, when a module or process created for a system is used multiple times within the same system and “external” when a module from a different system is used one or more times within a new system [7], [5]. Organizations usually take into account the functional similarity effect on effort in terms of external reuse. However, as external reuse, internal reuse has a significant impact on the total required development effort and time. Although software products have functionally similar modules or similar functional processes, it is not always easy to determine functionally similar software entities/processes especially at the beginning of the projects. Moreover it is not clear whether the functionally similar entities require exactly the same effort for the development or not, and what the impact of the similarity on the development effort should be.

In the literature, there are a few approaches to determine functional similarities. One of the approaches that being subject of this study is functional reusability approach which determines the similarities among functional processes assessing data groups and data movement action types [28] and do this earlier in the software life cycle. The other approach consists of the entity generalization/specialization practices which are widely used in object oriented methodologies and can be used in the grouping of the similar functional processes into one and as a result can be used to eliminate the replication of the same/similar functions [31]. The second approach can be applied at the beginning of the projects only by using expert judgment.

In this study we have conducted five separate case studies considering the functional similarity method of Santillo and Abran [28] and entity generalization approach [31] to assess the applicability of the functional similarity methods and to identify improvement opportunities related to them. We also evaluated the reflections of the functional similarity on
the relation between development effort and functional size.

The remainder of this paper is organized as follows: the related research is summarized in section 2; in section 3 the structure of the cases and the challenges are explained. Finally in the fourth section conclusions and lessons learned are given.

2. Related research

The idea of measuring size of a software product in terms of its functionality was first introduced by Alan Albrecht in 1979 [2]. The method is called Function Point Analysis (FPA) and has gained a considerable interest because it focuses on measuring the size from user perspective independent of the application itself.

Based on the Albrecht’s method several measurement techniques have been developed, each of which aiming the extension of the applicability of the techniques in different functional domains. Due to the proliferation of the techniques, the ISO/IEC workgroup has been initiated to identify fundamental concepts and to establish an international standard for functional size measurement [16] - [21].

Today, IFPUG FPA [14], MKII FPA [13], COSMIC FSM [15], NESMA FSM [22] and FISMA [23] are accepted as international standards for functional size measurement by ISO/IEC. All these methods measure the functionality from the perspective of the functionality provided to the user; however, they use different units and rules during measurement.

In terms of functionality, similarities on software applications have been subject to research projects and defined by using different terminologies. Fenton [7] defined a concept called “private reuse” as the extent to which modules within a product are reused within the same product. Cruickshank and Gaffney [5] also make first distinction for “internal” and “external” reuse in the literature from economical perspective.

Whatever the terminology is, the functional similarity concept has a significant effect on all phases of the life-cycle of the projects. For example, the effect of functional similarities and reusability in maintenance was evaluated by Abran and Desharnais in 1995 [1]. They have developed an approach for the identification and measurement of reuse in the enhancement projects by considering Function Point Analysis Method. Their approach depends on two key concepts: reuse indicator and predictor ratio. The study depicts how an alternative size measure can be obtained by combining predictive ratio and reuse indicator.

Functional similarity has been subject to one of the common functional size measurement methods, COSMIC. It defines the functional similarity concept in its Guideline for Sizing Business Applications document [30]. It is stated that developers might avoid duplications by realizing the functional reuse opportunities among functional processes; however, the user point of view ignores the functional similarities since the FURs are measured independently instead of grouping similar functional processes.

Abran and Maya [4] are one of the researchers evaluating similarities within a software product from a functional similarity perspective. They refined and extended the functional similarity measures to create a more precise measurement basis for the cost estimation and productivity models.

In addition to above, in the literature there are considerable numbers of research studies evaluating the software reuse performing at the source code level. However, few of these studies focus on developing methods to identify the functional similarities in the early phases of the software life cycle [3], [25]. Ho, Abran and Olingy [10] emphasize the importance of measuring the functional reuse impact in the early phases of the software life cycle rather than coding phase to improve the performance of the software engineering processes. Their work bases on extending the method of Abran et al [1] by using the COSMIC FFP method. The approach proposed in the paper considers only the reuses without modification and called black box approach. The approach utilizes the functional relationships among functional layers.

The development effort and the functional size correlation have been subject to research studies as well. In 2000, Meli [26] discussed the problems faced during the development effort and functional size correlation. He stated that in some situations, it is possible to aggregate much different logical functionality which leads to rapid and economic implementations with a small amount of working effort. As a result of this, the effort needed to realize the overall system will decrease, and will not be proportional at all to the logical functionalities required.

Only a few research studies focus on methods about the association of the size and the effort considering the functional similarity. Santillo and Della Noce [27] proposed a model named as “Worked Function Model” to achieve a more significant “work size” to be correlated with effort. Model includes “reuse”, “replication” and “similarity” adjustments.

In their study Santillo and Abran [28] proposed the approach called “functional similarity” to identify the software reuse from a functional perspective. The technique is based on uncovering the functional
similarities from a data set that comprises functional processes, data movements and data manipulations which are evaluated by using the COSMIC method. Although their study comprises a method sorting out functional similarities, it does not provide an approach for the relation of functional size and development effort.

Lastly, entity abstraction methods are also valid approaches for eliminating the measurement variances based on different point of views, providing abstract data sets by grouping similar functional processes and as a result eliminating the replication of the same functions. Thus, they can be evaluated as methods for determining the impact of functional similarities on size and development effort. A research study considering this approach has been conducted by Turetken et al. [31]. They depicted the utilization of entity generalization concept in COSMIC and IFPUG FPA Methods and evaluated the effect of different interpretations on the measurement results.

3. The case studies

We have conducted five different case studies with two analysts to evaluate different approaches to determine the applicability of the functional similarity approaches for different domains of application, to observe if functional similarity would improve the correlation between functional size and development effort, and to discover new improvement opportunities related to the methods.

The case studies involved the measurement of the case products by using the COSMIC method and evaluation of the functional similarities by applying the method of Santillo and Abran, and “entity generalization” approach.

The functional similarity method defined by Santillo and Abran were applied to all case products and one analyst applied entity abstraction method to one case product. The details of the cases are explained in the following chapters.

3.1. The cases

One of the cases –KAMA- is a conceptual modeling tool development project. The tool provides a common notation and a method for the conceptual model developers in different modeling and simulation development projects particularly in the military domain.

The other case easyARCHIDIM is a software size measurement tool development project. The tool has been developed to automate the measurement process of software products’ using ARCHitectural DIMensions Based Functional Size Measurement Method [8]. The tool keeps detailed information about the measurement process, calculates the size when the required details are enrolled to the system and provides detailed reports to the users.

The last three cases include the development of different simulators which have been nominated to the embedded platforms. They all have a 1553 communication interface which enables user to connect to a terminal side where data are taken from and send to. Simulators have a user friendly GUI which shows and controls all sent and received messages.

The cases can be grouped according to the functional domains of the products. With respect to CHAR Method defined in [20], the functional domain of the KAMA and easyARCHIDIM is ‘Information System’ and three simulation products is “Complex Data Driven Control System”.

3.2. Conduct of case studies

The case products were measured independently by using software requirements specification (SRS) documents by two separate teams each consisting one expert on the COSMIC method. SRS documents of each case are conformant with the IEEE Standard 830-1998 [11]. The measurement results were verified by a different team member who did not involved in the measurement process of the verified project. One of the two analysts measured Information System Projects; KAMA and easyARCHIDIM, the other analyst measured Complex Data Driven Control Systems; SIM-1, SIM-2, SIM-3.

The measurement results for the COSMIC Method are given on Table 1.

<table>
<thead>
<tr>
<th>Case Project</th>
<th>No of Functional Processes</th>
<th>No of Entries</th>
<th>No of Exits</th>
<th>No of Reads</th>
<th>No of Writes</th>
<th>Functional Size (CFP)</th>
</tr>
</thead>
<tbody>
<tr>
<td>KAMA</td>
<td>130</td>
<td>204</td>
<td>164</td>
<td>184</td>
<td>142</td>
<td>694</td>
</tr>
<tr>
<td>easy ARCHIDIM</td>
<td>7</td>
<td>21</td>
<td>22</td>
<td>20</td>
<td>12</td>
<td>75</td>
</tr>
<tr>
<td>SIM-1</td>
<td>53</td>
<td>107</td>
<td>98</td>
<td>39</td>
<td>33</td>
<td>277</td>
</tr>
<tr>
<td>SIM-2</td>
<td>44</td>
<td>73</td>
<td>33</td>
<td>24</td>
<td>63</td>
<td>193</td>
</tr>
<tr>
<td>SIM-3</td>
<td>11</td>
<td>82</td>
<td>16</td>
<td>4</td>
<td>17</td>
<td>129</td>
</tr>
</tbody>
</table>

Implementation of the method defined by Santillo and Abran. The functional similarity method
developed by Santillo and Abran has two stages. The first one which is called as “the first order evaluation” compares the functional processes only from data movements’ point of view. Similarity among functional processes are determined by comparing the data group and data movement relationships; in addition to this, in some cases where the comparison technique does not suffice, it is suggested that the analyst make her best judgments in order to identify the functional similarities. The second one, “second order evaluation” determines the functional similarities by considering both data movement and data manipulation action types.

Our standpoint to the evaluation of the functional similarity method consisted of both the “first order evaluation” and “second order evaluation” approaches. We applied the second order evaluation to four case studies: easyARCHIDIM, SIM71, SIM72, and SIM73. However, the COSMIC method assumes that all data manipulations within a functional process are associated with one of the four data movement types; therefore data manipulations are not considered in the measurement process. Since these sub processes are ignored, detailed information about the determination of the data manipulations within a software product does not exist in the COSMIC Manual [15].

For all the case products the first step was to determine the functional size of the products by using the COSMIC method. After this step, detailed size measurement data sets were arranged to uncover the amount of functional similarities. The second step was to compare the data group and data movement couples within different functional processes, which required comparing all the couples with each other.

When the defined methodology of Santillo and Abran was examined in detail, it was observed that their functional similarity approach was verified by only small scale projects. Observing this one of the aims of this study has been to evaluate the applicability of their approach to a large scale project.

In the initial part of the research, the “first order evaluation approach” was applied to a portion of the KAMA, the case product, which is a relatively large scale project and has a potential to highlight the new improvement opportunities on the functional similarity methods. During the evaluation of its functional similarities, since KAMA had 694 data movements, 694*693 numbers of comparisons were required. As Santillo and Abran have emphasized, comparisons of the action types among the functional processes are very time consuming. Beyond this, it is impossible to accomplish such a job without errors.

In order not to evaluate 694*693 action types one by one, the best solution was to develop a Reuse Evaluation Program (REP) which automatically calculated the percentage of the functional similarity of the functional processes. With the automation of the process, comparison time was decreased to seconds. The REP provided a significant improvement to constitute a similarity matrix; however, it is not capable of making heuristic interpretations in the identification of functional similarities. Therefore, it is insufficient in the situations where the best judgment of the analyst is required.

After having measured the size of the KAMA, the records were evaluated from two different perspectives by using the REP. In the first evaluation, the functional similarities were extracted from among 130 functional processes; in the second evaluation, these functional processes were clustered into 9 modules according to types of “conceptual module elements” (such as Entity, Actor, Role…) with respect to the expert judgment. The amount of their functional similarities was determined among modules.

When 130 functional processes were compared, we reached a similarity matrix comprising of 16900 cells. An example set of this evaluation is given on Table 2. The first horizontal and vertical cells identify the functional processes (A, B, C and D...) and the numbers on this table are the functional similarity percentages which mean for instance A is 100% similar to A and B is 33 % similar to A.

<table>
<thead>
<tr>
<th>Functional Processes</th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>1</td>
<td>0.076923</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>B</td>
<td>0.33333</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>C</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0.33333</td>
</tr>
<tr>
<td>D</td>
<td>0</td>
<td>0.25</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

When the functional processes were grouped into the modules, we reached a similarity matrix consisting of 81 cells instead of 16900. An example portion of the results are given on Table3. On this table the first vertical and horizontal cells identify the modules, and the numbers identify the functional similarity percentages. As can be seen, the similarity results increase with this approach.

Table 2 an example set for the standard approach of KAMA
Table 3: an example set for the clustering approach of KAMA

<table>
<thead>
<tr>
<th>Functional Modules</th>
<th>Entity Model Element</th>
<th>Actor Model Element</th>
<th>Role Model Element</th>
</tr>
</thead>
<tbody>
<tr>
<td>Entity Model Element</td>
<td>1</td>
<td>0.96842</td>
<td>0.97895</td>
</tr>
<tr>
<td>Actor Model Element</td>
<td>0.96842</td>
<td>1</td>
<td>0.98947</td>
</tr>
<tr>
<td>Role Model Element</td>
<td>0.93</td>
<td>0.94</td>
<td>1</td>
</tr>
</tbody>
</table>

As Santillo and Abran emphasize, the average of the functional similarities can be calculated in order to be able to make a judgment about the reuse capacity of the product. Our two sided approach to the same product revealed that calculating the average of the functional similarity might not be the best approach. Since we evaluated exactly the similarity of the same product, we reached completely different functional similarity results as can be seen on Table 4.

Table 4: the results of Functional Similarity of KAMA

<table>
<thead>
<tr>
<th>CASE STUDY</th>
<th># of Functional Processes</th>
<th># of Modules</th>
<th>Size of the Product (CFP)</th>
<th>Functional Similarity Percentage</th>
</tr>
</thead>
<tbody>
<tr>
<td>KAMA</td>
<td>130</td>
<td>–</td>
<td>694</td>
<td>8.78%</td>
</tr>
<tr>
<td>KAMA clustered</td>
<td>–</td>
<td>9</td>
<td>694</td>
<td>59.02%</td>
</tr>
</tbody>
</table>

While reading Functional User Requirements, it has been found out that the similarities among functional modules are clearly more remarkable than the similarities among functional processes. Therefore, it can be said that the clustering approach reflects a better similarity ratio.

In the second part of the research, both the first order and the second order evaluation approaches have been applied to case products; Simulator Projects and easyARCHIDIM. The results of the analysts were also verified by REP. The first order and the second order evaluation results are given on Table 5. During the second order evaluation we came to the conclusion that there may be other kinds of data manipulations apart from the ones given in Santillo and Abran’s research. These other type of manipulations may depend on the application types to provide appropriate granularity. For instance, it was realized that even if two of the calculation data manipulations within the easyARCHIDIM were different, they were considered similar because of the current data manipulation types. However, for this situation the complexities behind the calculation processes should be taken into account.

Table 5: The results of first order and second order evaluations

<table>
<thead>
<tr>
<th>CASE STUDY</th>
<th># of Functional Processes</th>
<th>Size of the Product (CFP)</th>
<th>Avg. Functional Similarity Percentage (First Order)</th>
<th>Avg. Functional Similarity Percentage (Second Order)</th>
</tr>
</thead>
<tbody>
<tr>
<td>SIM-1</td>
<td>53</td>
<td>277</td>
<td>79.0%</td>
<td>52.41%</td>
</tr>
<tr>
<td>SIM-2</td>
<td>44</td>
<td>193</td>
<td>57.75%</td>
<td>34.37%</td>
</tr>
<tr>
<td>SIM-3</td>
<td>11</td>
<td>129</td>
<td>25.8%</td>
<td>22.36%</td>
</tr>
<tr>
<td>easy ARCHIDIM</td>
<td>7</td>
<td>74</td>
<td>42.37%</td>
<td>39.54%</td>
</tr>
</tbody>
</table>

Actually a closer study of the FURs provided a clear insight to the analyst about the similarities among functional processes. In our study, the functional similarities obtained from the Simulator Projects verified the observed functional similarity results. In the next step, when we applied the second order evaluation technique in the simulator projects and easyARCHIDIM, we observed decreases in the similarity percentages, which we had predicted. The reason why the first order evaluations are higher than the second order evaluations is that in the first evaluation, any of the four data movement types is used instead of data manipulations. What causes this change is the replacement of some of the data movements with data manipulations. In this way the same two data group data movement couples become different from each other.

Productivity comparisons: The other significant reason for this research was to investigate if the functional similarity results could be used for the correlation of the functional size and the development effort. In order to reach out for this aim we selected three different projects, SIM-1, SIM-2, and SIM-3 which were developed in the same environment, using the same language and by the same team in the same organization. The reasons why the SIM projects were selected were that their design documents and user manuals were fully detailed and could be used for accurate evaluations. In addition, the project team had collected the effort metrics in man-day basis during the development of the projects. This enabled us to reach actual effort values. As mentioned above, since the projects were developed under the same conditions it is
assumed that the productivity values were the same and could therefore be compared. For the evaluation of productivities we compared the all three effort values for the whole lifecycle. In the table below, initial productivity values represent the productivities in which the functional similarities were not taken into account.

<table>
<thead>
<tr>
<th>Project</th>
<th>Functional Size (CFP)</th>
<th>Initial productivity</th>
<th>Productivity Results After Similarity Considered</th>
</tr>
</thead>
<tbody>
<tr>
<td>SIM-1</td>
<td>277</td>
<td>9,8993</td>
<td>4,24</td>
</tr>
<tr>
<td>SIM-2</td>
<td>193</td>
<td>6,89</td>
<td>3,39</td>
</tr>
<tr>
<td>SIM-3</td>
<td>129</td>
<td>4,44</td>
<td>3,19</td>
</tr>
</tbody>
</table>

When we evaluate the initial productivity results of the projects, we observed that the effort required to complete these projects do not depend on the sizes of the projects since productivity values are so different. Moreover, although the size of the SIM-1 is greater than the other simulator projects, it had been finished in less time, with a higher productivity rate.

When we measure the size only from the user point of view and try to predict the effort by looking the historical metrics we attained three different effort values. In a sense even a new product would have been required to be developed by the same team it would be very difficult to estimate the effort required based on the historical data.

To attain more reliable productivity values, we counted Simulator Projects’ functional size again by considering functional similarities. In one instance, an enhancement of a module in one of these projects was required, and then we assumed that the effort needed to develop for this specific part was neglected. Therefore during the size measurement we ignored the effort required to modified part and reached new productivity results given in the 4th column on Table 6. These new productivity results reflected the real productivity of the team better since the values were closer to each other than the values in the 3rd column.

**Implementation of Entity Generalization Method**

The other method which can be used for identifying the amount of functional similarities is the entity generalization approach. We recounted the same portion of the KAMA by unifying all entities into one special entity and eliminated functionally similar processes [31]. The generalized entities are shown in Figure 1.

![Figure 1 Entities of KAMA case product](Image)

For example instead of having separate functional processes such as “constitution of entity model element” or “constitution of mission model element” we determined a single functional process called as the “constitution of conceptual model elements” and eliminated the replicated movements. As a result, for the KAMA we reached a 97 CFP size, instead of 694 CFP. Functional similarity is considered after having measured the functional size of a product to reach more usable size results. Since the entity generalization approach takes into account the functional similarities during size measurement; two steps which required for the evaluation of functional similarities and the evaluation of influence of reuse on the functional size are eliminated.

**3.3 SLOC and Functional Similarity Compatibility**

We hypothesize that when the functional similarity increases, the amounts of SLOCs will decrease. We analyzed the relationships between the source lines of code (SLOC) values and average functional similarity percentages of the case products. Four of the case products have been developed by using JAVA and the other has been developed by C# programming languages. According to data stated by Quantitative Software Management, Inc in [32], JAVA and C# are assumed to be the same numbers of SLOCs for the same functionality. Therefore we used the SLOC values as is for comparison.

We grouped the results of comparisons according to the functional domains of the projects. In the simulator projects while the similarities among functional processes increase; the numbers of SLOC decrease except for the SIM3. The reason behind this difference is that the SIM-3 is a smaller project and there exist less modules in it than the other simulator projects, therefore the probability of having similar functions reduces. The Information Systems projects also support this hypothesis. However, the difference between the
SLOCs does not correspond to the difference between the similarity percentages. Details of the comparisons are given on Table 7.

**Table 7 SLOCs and Average Functional Similarity Percentages of the Projects**

<table>
<thead>
<tr>
<th>Case Product</th>
<th>Functional Size</th>
<th>Average Functional Similarity</th>
<th>SLOC</th>
<th>SLOC (for 1 unit size)</th>
</tr>
</thead>
<tbody>
<tr>
<td>KAMA</td>
<td>694</td>
<td>59,02%</td>
<td>12397</td>
<td>18</td>
</tr>
<tr>
<td>easy ARCHIDIM</td>
<td>75</td>
<td>42,37%</td>
<td>3664</td>
<td>50</td>
</tr>
<tr>
<td>SIM1</td>
<td>277</td>
<td>79,00%</td>
<td>7586</td>
<td>27</td>
</tr>
<tr>
<td>SIM2</td>
<td>193</td>
<td>57,75%</td>
<td>8603</td>
<td>44</td>
</tr>
<tr>
<td>SIM3</td>
<td>129</td>
<td>25,80%</td>
<td>4211</td>
<td>32</td>
</tr>
</tbody>
</table>

4. Conclusions

In this research paper we presented the case study results evaluating the applicability of the functional similarity techniques and the impact of functional reuse on the effort and SLOCs of the projects.

We applied Santillo and Abran [28] for five different cases and entity generalization approaches [31] for one case.

One of the challenges we faced during the evaluation of the method defined by Santillo and Abran is that the approach does not easily scale up on large scale applications since the evaluation process is error prone and time consuming. We developed a program to partially automate the process to overcome these problems. The program, which can be used both for the first and second order evaluations, provided a significant improvement on the effort required for the evaluation process and decreased the possibility of occurrence of an error.

Since the functional similarities among functional processes are sparse, we have clustered functional processes into modules and depicted the results of both approaches. By adopting these two perspectives we were able to observe the impact of analyst judgment on the functional similarity values. The remarkable difference in the average percentages of the two perspectives revealed that according to the analyst point of view, there may arise significant differences in the results.

Since the COSMIC manual does not contain detailed explanations about how the data manipulations should be classified and uncovered from FURs, we had difficulty in the application of the second order evaluation of the method defined by Santillo and Abran. The sorting out processes of the data manipulations was not as easy as data movements. We had to make assumptions such as “the calculation data manipulations” are the same within the different processes; however, the algorithms may be completely distinct.

Entity generalization method resulted in eliminating similar processes and we reached the size of 97 CFP, instead of 694 CFP. It can be concluded that the method would provide a significant improvement for the correlation of effort and size.

The results of SLOC and CFP comparisons were promising. Although the amount of SLOCs decreases when the functional similarity percentages increases; the ratio between the average similarity percentage and the SLOC does not hold.

We also observed an increased correlation between CFP and development effort. When the functional similarities were not considered, distinct productivity values have been obtained for the same team. On the other hand when the functional similarities were taken into account productivity values were closer to each other. This reveals that functional size and productivity increase as we consider the functional similarities. However, the result of this study also indicates that future research is needed to perform more precise functional similarity calculations as well as to increase the correlation between development effort and functional size.

5. References


