CHAPTER III
PROPOSED METHODOLOGY

In this study, three experiments were conducted. The first approach is to separately estimate the software effort of each phase of a project, known as "phase-wise" software effort, and sum up these phase-wise software efforts to compare with the overall estimation of the project. The second approach is to directly estimate the overall software effort of a project only from the input features. The third approach is to separately estimate the software effort of any phase using information from prior phase. The process of proposed methodology consists of the following steps.

1) Data collection.
2) Data transformation.
3) Feature selection.
4) Phase-wise, overall, and adaptive phase-wise estimation model construction.

3.1 Data Collection

Thirty eight software projects were collected from two local development organizations. One government organization is a certified ISO 9001:2008. The other is a company with certified CMMI development level 3. At the outset, a set of questionnaire was compiled accompanying an interview with the project manager and selected developers. Some key development documents were requested during the interview, such as software requirements specification, software design specifications, and user's manual.

Of the thirty eight projects being developed from 2007 to 2012 were primarily written in PHP, C#, Visual Basic, Java, Java Android, and Objective-C. Application types consist of web application, mobile application, website, and web-API. Application domains include government and business. Development types were composed of new development, enhancement, re-development, and customization. Software process models are composed of waterfall and v-model. Intended markets are composed public, external, and internal. Note that "public", "external", and "internal" refer to a project developed for public people, external organization, and self-organization, respectively. The maximum team size ranged from three to fifteen persons. Software size ranged from 0.27 to 112.28 thousand of SLOC, while software effort ranged from 92 to 1,278 man-days. The data collected, hereafter referred to as Phase-wise Neural Network Estimation (PNNE). PNNE-1 referred to thirty three projects from the government organization while PNNE-2 referred to five projects from the company. Summary of data were provided in this section, where the details of all software projects were descripted in Appendix A-D.

Table 2-7 show the number of software projects in each programming language, application type, application domain, development type, software process model, and intended market. Most of software projects are written in C#, came from web application and government domain, were new development, were carried out under waterfall model, and were for public users.1

Table 2: Number of projects in each programming language.

Language	PNNE-1	PNNE-2	PNNE
PHP	4	5	9
C#	20	0	20
Visual Basic	1	0	1
Java	5	0	5
Java android	1	0	1
Objective C	2	0	2

Table 3: Number of projects in each application type.

Application Type	PNNE-1	PNNE-2	PNNE
Web application	23	5	28
Mobile application	3	0	3
Website	1	0	1
Web API	6	0	6

Table 4: Number of projects in each application domain.

Application Domain	PNNE-1	PNNE-2	PNNE
Government	33	0	33
Business	0	5	5

Table 5: Number of projects in each development type.

Development Type	PNNE-1	PNNE-2	PNNE

New	30	4	34
Enhance	0	1	1
Re-development	2	0	2
Custom	1	0	1

Table 6: Number of projects in each software process model.

Software Process Model	PNNE-1	PNNE-2	PNNE
Waterfall	33	0	33
V-model	0	5	5

Table 7: Number of projects in each intended market.

Software Process Model	PNNE-1	PNNE-2	PNNE
Public	20	0	20
External	8	5	13
Internal	5	0	5

Table 8 shows descriptive statistics of software efforts through development phases providing values of min, max, mean, median, standard deviation, and *p*-value tested by using Jarque-Bera [48] with 0.05 (α=0.05) significance level. Given *p*-value is less than a predetermined significance level (*p*-value < 0.05) indicates that software efforts are not normally distributed.

Table 8: Descriptive statistics of software efforts through phases.

Phase	Descriptive statistics					
	Min	Max	Mean	Median	Std.	*p*-value
			PNNE-1 data set			
Requirement	2	142	36.12	33	30.03	0.002
Design	13	253	76.55	53	62.55	0.004
Coding	45	755	187.33	128	141.21	0.001
Testing	5	77	24.15	16	18.89	0.008
Transition	2	179	38.03	21	44.68	0.001
Overall	120	1278	362.18	277	252.73	0.001
			PNNE-2 data set			
Requirement	7	20	14.80	16	5.07	0.500

Design	9	259	126.20	65	114.02	0.238
Coding	25	121	62.80	50	36.09	0.354
Testing	3	61	22.00	19	23.75	0.147
Transition	11	31	19.60	21	7.99	0.500
Overall	92	422	245.40	195	158.32	0.237

Figure 7 compares averaged effort distribution percentage of PNNE, ISBSG [27], CSBSG [26], and COCOMO II [6] data sets. It indicates that the distribution percentages of PNNE and COCOMO II data sets are relatively similar in requirement, design, and transition phases, but are different in coding and testing phases. Only the transition phase of all three data sets exhibits similar the distribution percentage. At any rate, the PNNE data set provides the highest distribution percentage in coding phase and the lowest distribution percentage in testing phase.

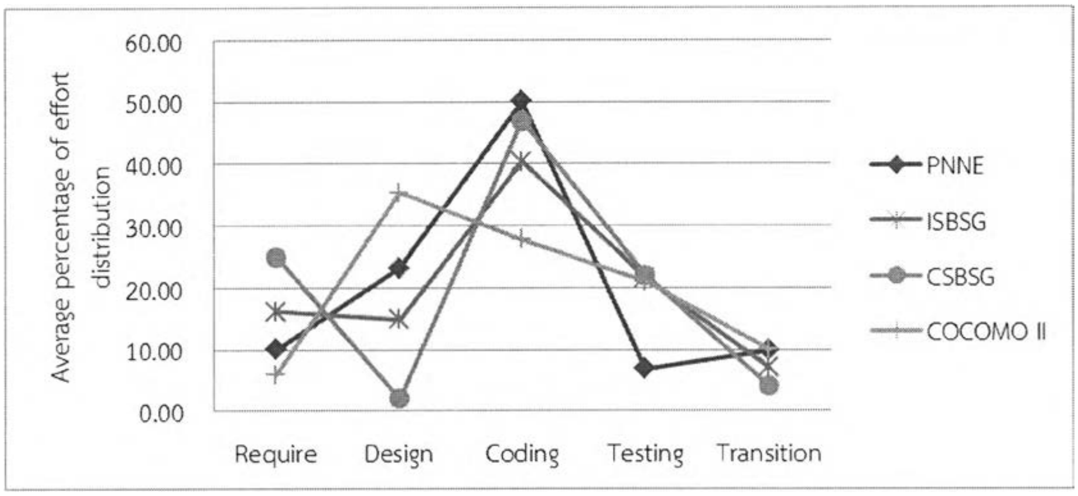


Figure 7: Comparison of averaged percentage of effort distribution between PNNE, ISBSG, CSBSG, and COCOMO II data sets.

Figure 8 compares averaged effort distribution percentage of PNNE-1 and PNNE-2 data sets. It indicates that the distribution percentages of PNNE-1 and PNNE-2 data sets are relatively similar in requirement, testing, and transition phases, but are different in coding and design phases.

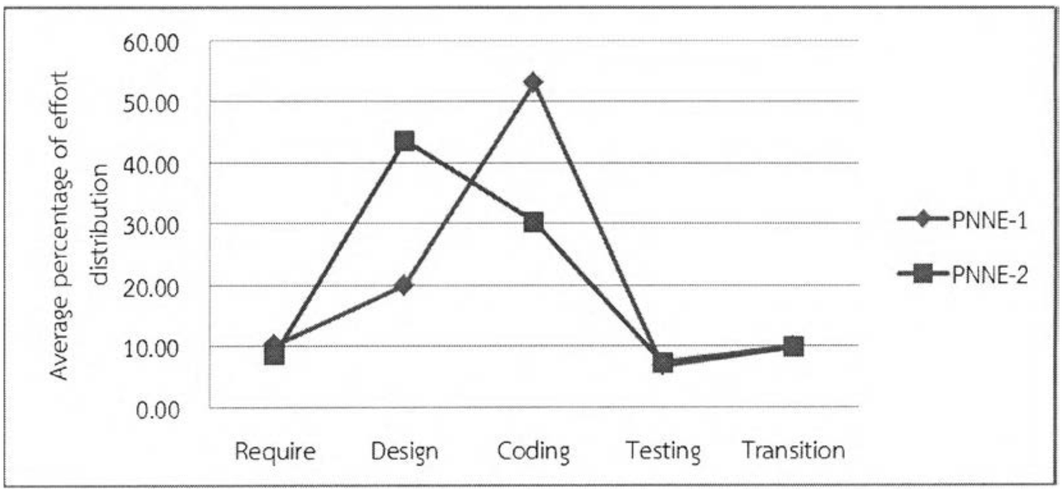


Figure 8: Comparison of averaged percentage of effort distribution between PNNE-1 and PNNE-2 data sets.

There were 44 features of each software project. Twenty three features were collected according to COCOMO II model [6], i.e., source lines of code, five scale factors, and seventeen effort drivers. Twelve features were according to general system characteristics of IFPUG FSM [21], where two features (reusability and complex processing) of the system characteristics were not collected for this experiment because the meanings were similar to features of the COCOMO II. Four features were corresponding to technical complexity of Mk II standard [21]. Six features were selected from UCP [12].

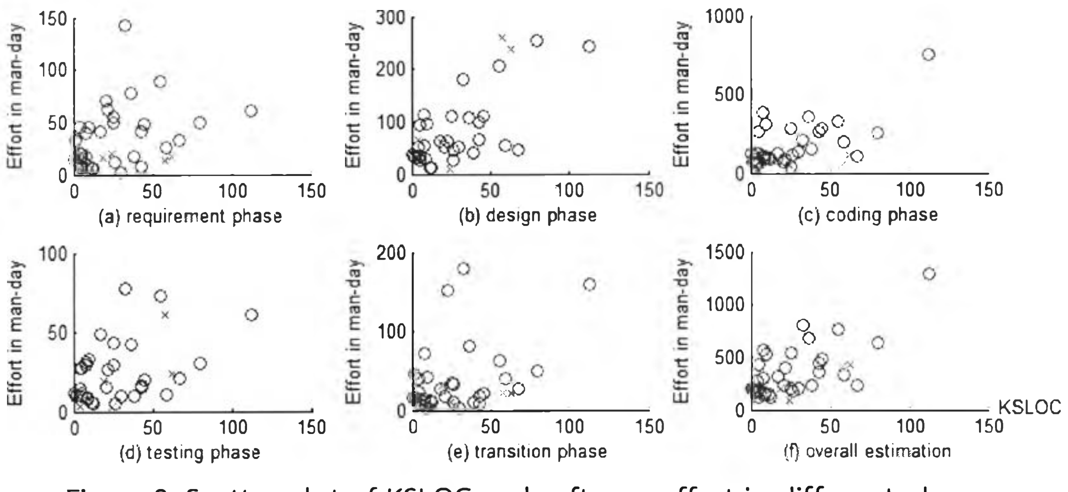


Figure 9: Scatter plot of KSLOC and software effort in different phases.

Figure 9 shows scatter of KSLOC and software effort of (a) requirement, (b) design, (c) coding, (d) testing, (e) transition, and (f) overall estimation. A cycle sign indicates the scatter of software projects in PNNE-1 data set while a cross sign indicates the scatter of those in PNNE-2 data set.

3.2 Data Transformation

In PNNE data set as shown in Appendix C, there were two types of data including quantitative and qualitative data. There was only one feature (i.e., KSLOC) representing as quantitative data in ratio scale which could be instantly used as predictor variable for the estimation. In contrast, all remaining features were represented as qualitative data in ordinal scale or 7-likert scale, i.e., extra low (XL), very low (VL), low (L), nominal (N), high (H), very high (VH), and extra high (XH). These features would be transformed before the estimation. Table 9 provided quantitative values of each rating scale in COCOMO II model [6].
Table 10-11 provide the values in IFPUG FSM [19] and Mk II FPA [21] while Table 12 yields the values in UCP [12]. For Table 9-12, first column refers to feature order appeared in Appendix C. The second column denotes feature abbreviation, and the remaining ones refer to values of each rating scale.

Table 9: Values of rating scales in COCOMO II model.

No.	Feature	Rating Scale						
		VL	L	N	H	VH	XH	
2	PREC	6.20	4.96	3.72	2.48	1.24	0	Scale factor
3	FLEX	5.07	4.05	3.04	2.03	1.01	0	
4	RESL	7.07	5.65	4.24	2.83	1.41	0	
5	TEAM	5.48	4.38	3.29	2.19	1.10	0	
6	PMAT	7.80	6.24	4.68	3.12	1.56	0	
7	RELY	0.82	0.92	1	1.10	1.26	-	Product
8	DATA	-	0.90	1	1.14	1.28	-	
9	CPLX	0.73	0.87	1	1.17	1.34	1.74	
10	RUSE	-	0.95	1	1.07	1.15	1.24	
11	DOCU	0.81	0.91	1	1.11	1.23	-	
12	TIME	-	-	1	1.11	1.29	1.63	Platform
13	STOR	-	-	1	1.05	1.17	1.46	
14	PVOL	-	0.87	1	1.15	1.30	-	
15	ACAP	1.42	1.19	1	0.85	0.71	-	Peop le
16	PCAP	1.34	1.15	1	0.88	0.76	-	

17	PCON	1.29	1.12	1	0.90	0.81	-	
18	APEX	1.22	1.10	1	0.88	0.81	-	
19	PLEX	1.19	1.09	1	0.910	0.850	-	
20	LTEX	1.20	1.09	1	0.910	0.840	-	
21	TOOL	1.17	1.09	1	0.900	0.780	-	
22	SITE	1.22	1.09	1	0.930	0.860	0.800	Project
23	SCED	1.43	1.14	1	1	1	-	

Table 10: Values of rating scales in IFPUG FSM.

No.	Feature	Rating					
		XL	L	VL	N	H	VH
24	COMM	0	1	2	3	4	5
25	DIST	0	1	2	3	4	5
26	PREF	0	1	2	3	4	5
27	CONF	0	1	2	3	4	5
28	TRAN	0	1	2	3	4	5
29	DESI	0	1	2	3	4	5
30	ENTR	0	1	2	3	4	5
31	UPDA	0	1	2	3	4	5
32	INST	0	1	2	3	4	5
33	OPER	0	1	2	3	4	5
34	ISIT	0	1	2	3	4	5
35	CHAN	0	1	2	3	4	5

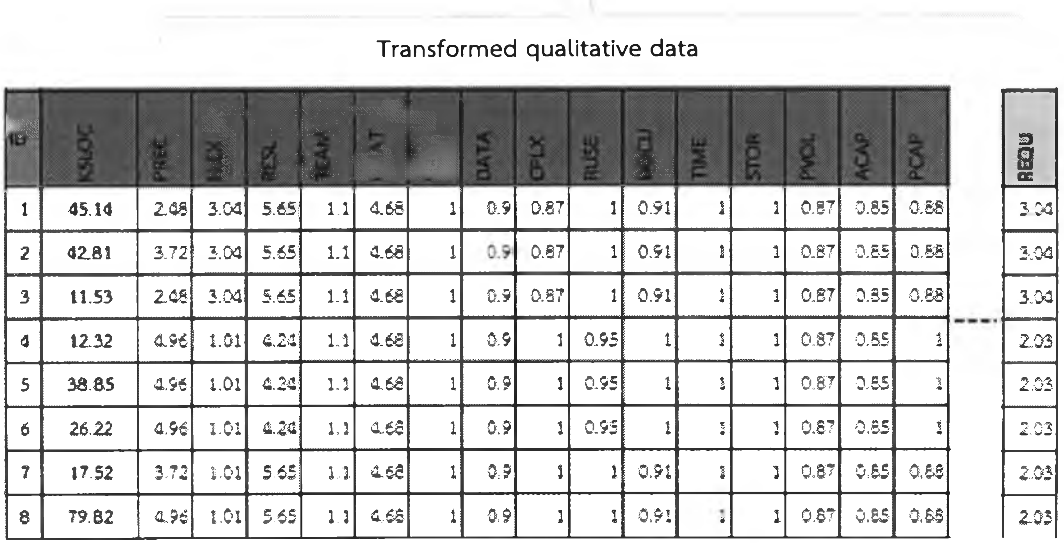
Table 11: Values of rating scales in Mk II FPA.

No.	Feature	Rating					
		XL	L	VL	N	H	VH
36	APPL	0	1	2	3	4	5
37	SECU	0	1	2	3	4	5
38	TRAI	0	1	2	3	4	5
39	PART	0	1	2	3	4	5

Table 12: Values of rating scales and weights in UCP.

No.	Feature	Rating Scale					
		XL	L	VL	N	H	VH
40	SWPR	0	1	2	3	4	5
41	WORK	0	1	2	3	4	5
42	MOTI	0	1	2	3	4	5
43	LANG	0	1	2	3	4	5
44	REQU	0	1	2	3	4	5

Since there was difference of the value of rating scale from difference data formats, the value of COCOMO II model as shown in Table 9 was kept while that of IFPUG FSM, Mk II FPA, and UCP was replaced with the value derived from COCOMO II. The values of all features of IFPUG FSM and Mk II FPA together with LANG and REQU of UCP were replaced with average rating scale value of product perspective of COCOMO II. Note that there was no extra low (XL) scale in COCOMO II, so it was replaced with very low (VL) for IFPUG FSM, Mk II FPA, and UCP. The values of remaining UCP features were replaced with average value of people perspective. Figure 10 shows examples of software project data transformed from qualitative data to quantitative data to be appreciate data for computing.

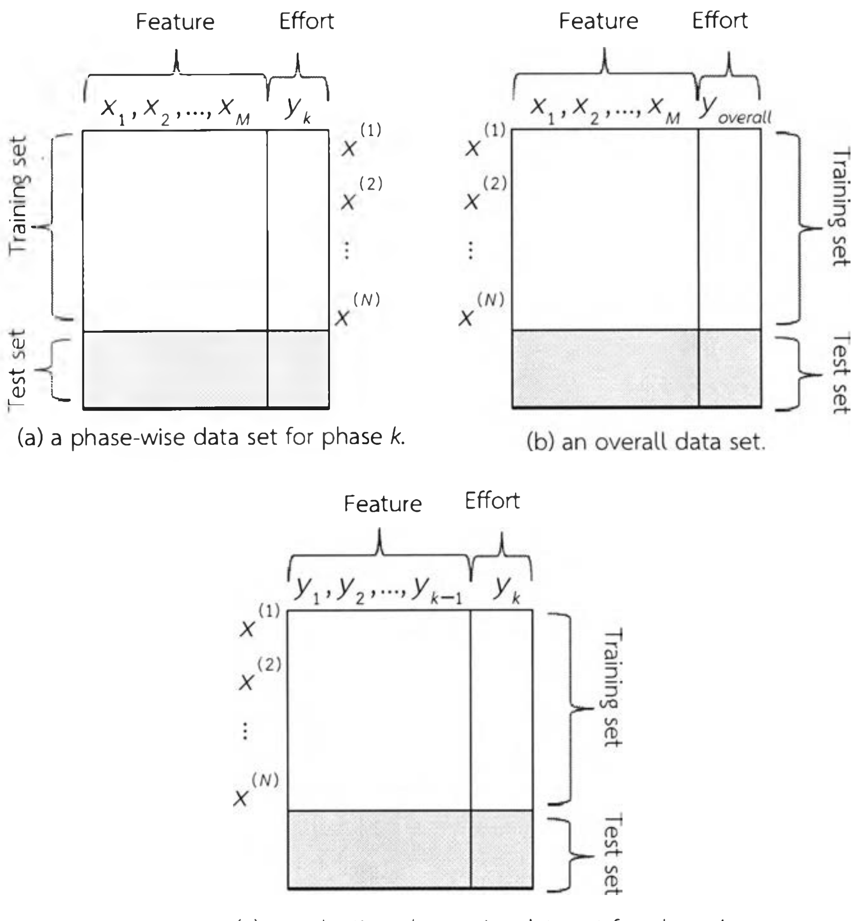

Transformed qualitative data

ID	KSLOC	PREC	FLEX	RESL	TEAM	AT	[illegible]	DATA	CPLX	RUSE	DOCU	TIME	STOR	PVOL	ACAP	PCAP	REQU
1	45.14	2.08	3.04	5.65	1.1	4.68	1	0.9	0.87	1	0.91	1	1	0.87	0.85	0.88	3.04
2	42.81	3.72	3.04	5.65	1.1	4.68	1	0.9	0.87	1	0.91	1	1	0.87	0.85	0.88	3.04
3	11.53	2.08	3.04	5.65	1.1	4.68	1	0.9	0.87	1	0.91	1	1	0.87	0.85	0.88	3.04
4	12.32	4.96	1.01	4.24	1.1	4.68	1	0.9	1	0.95	1	1	1	0.87	0.55	1	2.03
5	38.85	4.96	1.01	4.24	1.1	4.68	1	0.9	1	0.95	1	1	1	0.87	0.85	1	2.03
6	26.22	4.96	1.01	4.24	1.1	4.68	1	0.9	1	0.95	1	1	1	0.87	0.85	1	2.03
7	17.52	3.72	1.01	5.65	1.1	4.68	1	0.9	1	1	0.91	1	1	0.87	0.85	0.88	2.03
8	79.82	4.96	1.01	5.65	1.1	4.68	1	0.9	1	1	0.91	1	1	0.87	0.85	0.88	2.03

Figure 10: Examples of software project data transformed to quantitative data.

3.3 Feature Selection

Feature selection is a method to choose relevant features for constructing the estimation model. Let $X=\{x_1, x_2, \ldots, x_M\}$ for $1 \leq j \leq m \leq M$, $1 \leq i \leq N$ be a training feature set having N projects and M features, and $y=\{y_1, y_2, \ldots, y_K\}$ for $1 \leq k \leq K$ be a training software effort set having K phases. Effort estimation is based on the assumption that there exists some relevant features m out of M that actually exert significant effect on software effort estimation.



(a) a phase-wise data set for phase k.

(b) an overall data set.

(c) an adaptive phase-wise data set for phase k.

Figure 11: Examples of phase-wise and overall data sets.

This assumption leads to the main problem of how to select those relevant features. The solution to this problem depends on the observation that estimating any software effort from a set of given features is similar to developing a function to compute the value of software effort using the given features as its variables. To estimate software effort of each phase, let $x_j^{(i)}$ for $1 \leq j \leq m \leq M$, $1 \leq i \leq N$ be the value of feature *j* of project *i*. Thus, the software effort of project *i* of software development phase *k*, denoted by $y_k^{(i)}$, along with the corresponding features $\{x_j^{(i)} \mid 1 \leq j \leq m\}$ as can be depicted in Figure 11(a) written in the form of a mathematical function.

$$y_k^{(i)} = f_k\,(x_1^{(i)}, x_2^{(i)}, \dots, x_m^{(i)}) \tag{23}$$

where the software development phases refer to planning and analysis, design, coding, testing, and transition phase. For overall effort estimation in Figure 11(b), the form of a mathematical function is shown below.

$$y_{overall}^{(i)} = f_{overall}(x_1^{(i)}, x_2^{(i)}, \dots, x_m^{(i)}) \tag{24}$$

where $y_{overall}^{(i)}$ is the software effort of project *i* of an overall effort.

Both two equations should be used at the initial phase for planning and outsourcing. Adaptive phase-wise estimation would be introduced after a project is started. To estimate software effort of each phase, let $y_j^{(i)}$ for $2 \leq j \leq k-1$ be the value of feature *j* of project *i*. Thus, the software effort of project *i* of software development phase *k*, denoted by $y_k^{(i)}$ can be depicted in Figure 11(c) written in the form of a mathematical function.

$$y_k^{(i)} = g_k\,(y_1^{(i)}, y_2^{(i)}, \dots, y_{k-1}^{(i)}) \tag{25}$$

In the feature selection process, all constant features are investigated first. A constant feature is a feature which has the same value for all projects in the training set, such as Platform Volatility (PVOL) and Required Development Schedule (SCED) (see Appendix C). All constant features are eliminated from project set $\mathbf{X} = \{X_1, \dots, X_N\}$ because they cannot differentiate one project from the others in a data set or have low power of effort estimation.

Then backward search is applied to the remaining features in order to select an appropriate set of feature vectors to possibly achieve the highest estimation accuracy. There are two cases of anomalous projects that can degrade the estimation accuracy, i.e., (1) "outlying" projects and (2) groups of projects having the same features but different software efforts. If two or more projects have the same features but different software efforts, estimating the software effort of each project is indeterminate since it creates one-to-many relation between the features and software efforts. In this situation, only one project is selected and the others are ignored. The latter is called *"illegitimate"* project. The selection procedure will be elaborated in Algorithm 3.

For each selected feature set, two tests are conducted to confirm its relevancy to the accuracy of software effort estimation. The first test is by training the data set with the selected feature set and measuring the mean magnitude of relative error (MdMRE) [49]. The cross validation is applied for the training, so the final performance is average of MdMRE. The second test is by measuring the sum square Spearman rank coefficient of the selected feature set. For a selected feature set s_j, let $c_{k,l}^{(j)}$ be the square Spearman coefficient between features k and l of selected feature set j. The sum square Spearman rank coefficient of s_j, denoted by r_j, is defined as $r_j = \sum_{k,l;k\neq l;1\leq k,l\leq M} c_{k,l}^{(j)}$.

Let q be the number of projects that can be excluded from $\mathbf{X}$. In this study, the value of q was set to 10% of $|\mathbf{X}|$. The algorithm for feature selection is shown below.

Algorithm 1: Selecting Relevant Features.

```
1:  Let $\mathbf{s}_j$ be the $j^{th}$ set of indices of selected features.
2:  Let $\mathbf{S}$ be a set of each selected feature set $s_j$.
3:  Let s be a set of each MdMRE obtained from training the data set with
    each selected feature set.
4:  Let $\mathbf{R}$ be a set of sum square Spearman rank coefficients of each $\mathbf{s}_j$.
5:  Let $\mathbf{B} = \mathbf{X}$, where all $M$ features selected.
6:  Initialize variable BestMeanSquareError obtained from training $\mathbf{B}$ by a neural
    network,
    where anomalous projects in $\mathbf{B}$ are excluded before the training.
7:  For each feature $i$ such that $M \geq i \geq 2$
8:      Set $\mathbf{S} = \varnothing; \mathbf{E} = \varnothing; \mathbf{R} = \varnothing$.
9:      For each feature $j$ such that $1 \leq j \leq i$
10:         Put all indices of selected features from the first index to the $i^{th}$ index in set
            $\mathbf{s}_j$.
11:         Exclude feature $j$ from set $\mathbf{s}_j$.
12:         Based on selected feature set $\mathbf{s}_j$, let $\mathbf{A}$ be a set of illegitimate projects in $\mathbf{B}$
            found by Algorithm 2.
13:         Exclude $\mathbf{A}$ from $\mathbf{B}$.
```

```
14:     If Size of $|A| > q$
15:         Discard selected feature set $s_j$.
16:     Else
17:         Find all outlying projects from $B - A$ by Algorithm 3 and put them in $O$
            based on z-score and Mahalanobis distance.
18:         Let $v = q - |A|$.
19:         If $|O| > v$ then
20:             Exclude the first $v$ projects from $B$.
21:         End if
22:         Let $e_j$ be the MdMRE obtained from training $B$ with
            $s_j$ by an estimation technique (i.e., a neural network).
23:         Let $r_j$ be the sum square Spearman rank coefficients computed from $s_j$.
24:         Set $S = S \cup \{s_j\}$.
25:         Set $E = E \cup \{e_j\}$.
26:         Set $R = R \cup \{r_j\}$.
27:     End if
28:   End for
29:   If all $s_j$ for $1 \leq j \leq i$ are discarded then
30:     break the for loop.
31:   Else
32:     Let $g$ be a set of indices of minimum $e_j \in E$.
33:     For each index $k \in g$
34:         Let $b = \text{argmin}_k \{r_k \in R\}$.
35:     End for
36:     If $e_b > BestMeanSquareError$
37:         Terminate.
38:     Else
39:         Set $BestMeanSquareError = e_b$.
40:         Select feature set $s_b$.
41:     End If
42:   End if
43: End for
```

Consider the following example when 4 features are selected from total 5 features, forming 5 selected features sets, i.e., **S**={{2,3,4,5}, {1,3,4,5}, {1,2,4,5}, {1,2,3,5}, {1,2,3,4}}. Suppose that the MdMRE after training by using all selected feature sets in **S** are in *E*={0.01,0.1,0.5,0.7,0.01}. The sum square Spearman rank coefficient of each s_j is in **R**={2,4,3,3,1}. The minimum MdMRE are e_1 and e_5 which is equal to 0.01 and the corresponding indices are 1 and 5. In set **R**, the values of sum square Spearman rank coefficients corresponding to indices 1 and 5 are 2 and 1, respectively. The minimum coefficient is at index 5 which is equal to 1. Therefore, the selected feature set to be used for training and testing is s_5={1,2,3,4}.

Algorithm 2: Filtering Illegitimate Projects.

```
1: Compute the squared $z$-scores of software efforts of all projects in X.
2: Sort all projects according to their squared $z$-scores in descending order.
3: Set $\mathbf{A} = \varnothing$.
4: For each project $i$, $1 \le i < N$
5:     For each project $j$, $i < j \le N$
6:         If project $X_i$ and project $X_j$ have exactly the same feature values
7:         but different effort $y^{(i)}$ and effort $y^{(j)}$ then
8:             $\mathbf{A} = \mathbf{A} \cup \{X_i\}$
9:         End if
10:    End for
11: End for
```

Algorithm 3: Filtering Outlying Projects.

```
1: Let $\mathbf{O} = \varnothing$ be all outlying projects .
2: Let $\mathbf{O}_1 = \varnothing$ be outlying projects from software efforts.
3: Let $\mathbf{O}_1 = \varnothing$ be outlying projects from all projects in X.
4: Compute the squared z-scores of software efforts which are pairs of all projects
   in X and put the $z$-scores into Z and sort all squared z-scores in descending
   order.
5: Repeat
6: Test null hypothesis of the z-scores with Jarque-Bera test.
7:     if the null hypothesis is rejected then
8:         Put a project which has the relationship with software effort whose the
           top z-score into $\mathbf{O}_1$.
```

```
9:         Remove this z-score from **Z**.
10:     **End if**
11: **Until** test the hypothesis is accepted
12: Compute the Mahalanobis distance between a feature vector of each project and
    mean vector in **X** and sort the distance of all projects in descending order.
13: **Repeat**
14: Test null hypothesis of the distance with Mardia's multivariate test
15:     **if** the null hypothesis is rejected **then**
16:         Put the project from top distance into $O_2$.
17:         Remove this project from **X**.
18:     **End if**
19: **Until** test the hypothesis is accepted
20: $O = O_1 \cap O_2$
```

Outliers (outlying project) can affect low model performance [50]. So they should be handled properly before building estimation models. Jarque-Bera [48] and Mardia's multivariate [51] tests are used to detect and eliminate the outliers in **X**. The null hypothesis is tested whether the data have normal distribution. If the given *p*-value is less than a predetermined significance level (*p*-value < 0.001), the null hypothesis is rejected. Setting low *p*-value would enable the estimation to be highly tolerant to outlying projects as much as possible. An outlier can be the cause of the distances measured from different project features and different values of software efforts.

To find the outliers caused by the different software effort values, all effort values are transformed into squared z-scores and sorted in descending order. If the null hypothesis is rejected, a project which has a relationship with software effort whose first z-score is considered as an outlier. This software effort will be removed and the hypothesis testing is represented until it is accepted.

Note that the z-score is suitable only to a set of scalar values. Therefore, to detect an outlier caused by the distance of project features which are in the form of vectors, the Mahalanobis distance between a feature vector of each project and mean vector in **X** is computed and sorted in descending order. If the null hypothesis is rejected, a project which belongs to the first distance is considered as an outlying project. This project will be removed and the hypothesis testing is repeated until it is accepted.

3.4 Building Estimation Model and Estimating Software Effort.

To build a phase-wise estimation model, an estimation technique is used to create a function according to Equation (23). Similarly, an overall estimation and an adaptive phase-wise estimation models are built using an estimation technique to create a function according to Equation (24) and (25), respectively. Estimation techniques composed of neural networks, fuzzy inference system, support vector regression, regression analysis, classification and regression tree, and *k*-nearest neighbor.

Figure **12** shows flows of proposed phase-wise and overall effort estimation models while

Figure **13** shows those of adaptive phase-wise effort estimation model. Note that data transformation process is not included for the adaptive phase-wise model. Algorithm 4 describes how to build and evaluate an estimation model.

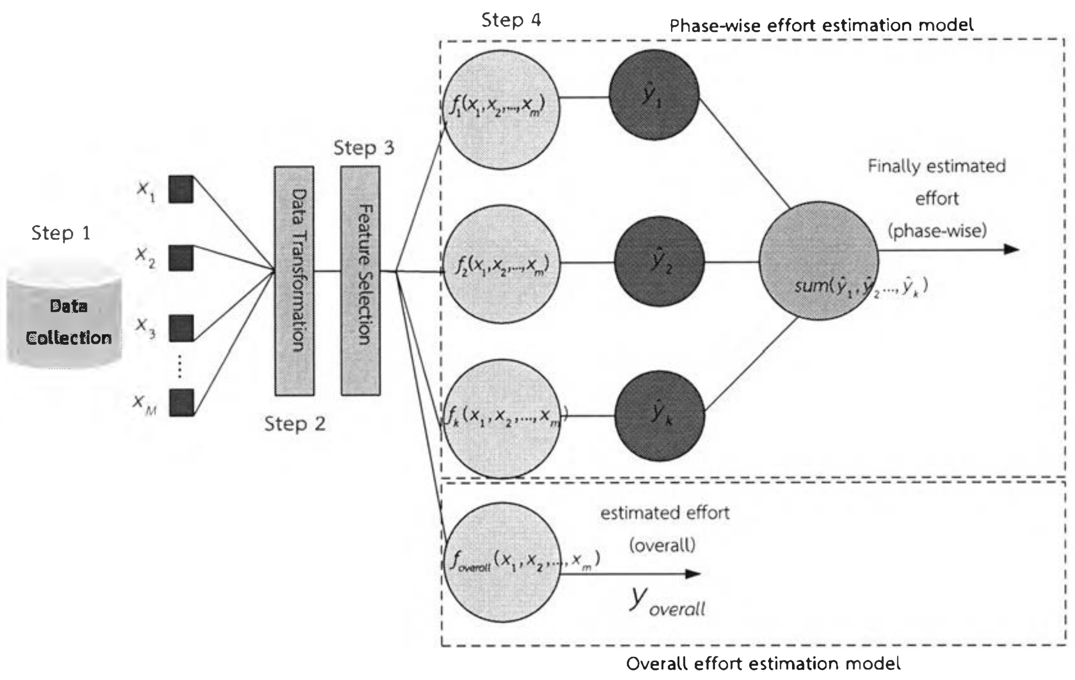


Figure 12: Flows of proposed phase-wise and overall effort estimation.

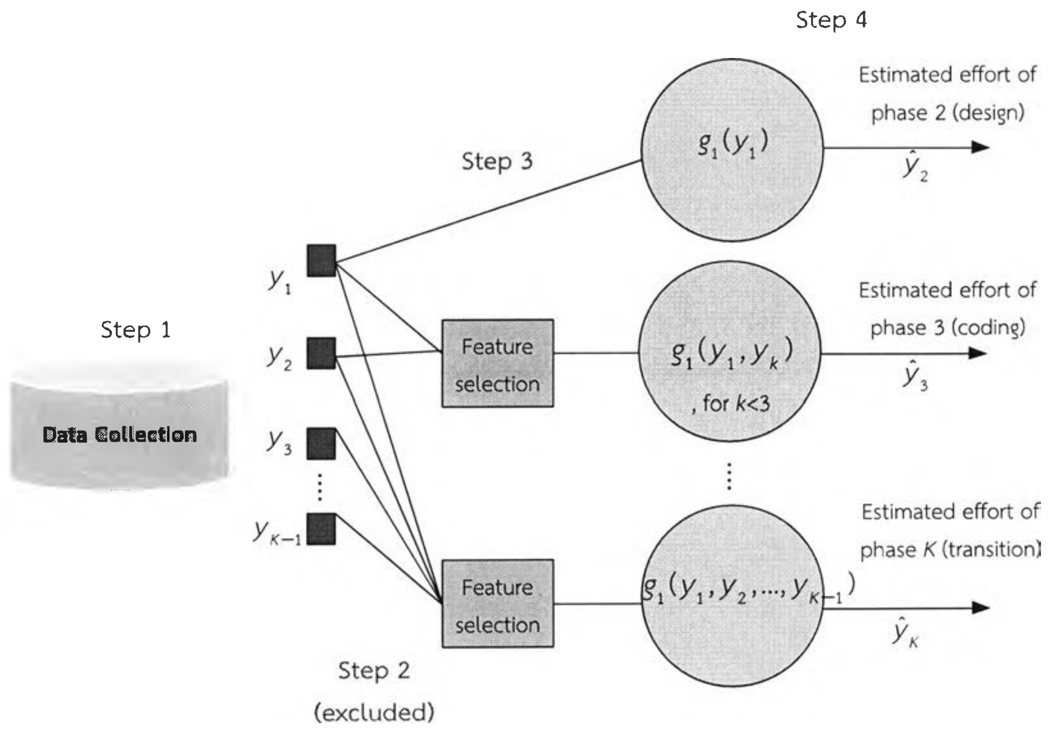

Step 4
Estimated effort of phase 2 (design)
$g_1(y_1)$
$\hat{y}_2$
Step 3
y_1
Step 1
y_2
Feature selection
$g_1(y_1, y_k)$, for k<3
Estimated effort of phase 3 (coding)
$\hat{y}_3$
Data Collection
y_3
y_{K-1}
Estimated effort of phase K (transition)
Feature selection
$g_1(y_1, y_2, \ldots, y_{K-1})$
$\hat{y}_K$
Step 2 (excluded)

Figure 13: Flows of proposed adaptive phase-wise effort estimation.

Algorithm 4: Building and evaluating estimation model.

Step 1: Data sets with all features are divided into k sub sets with k-fold cross-validation.

Step 2: For $1 \leq i \leq k$, let sub set$_i$ be a test set and other sub sets be a training set.

Step 3: The training set is taken into feature selection process.

Step 4: The training set with a feature set derived from the feature selection process is used to create an estimation model via an estimation technique.

Step 5: The test set is used to evaluate an estimation model by value of error metrics.

Step 6: Repeat Step 2.

Step 7: The performance of the model is considered from mean of all the values.