Using Grey Relational Analysis to Predict Software Effort with Small Data Sets

Qinbao Song*, Martin Shepperd†, and Carolyn Mair‡

Abstract

The inherent uncertainty of the software development process presents particular challenges for software effort prediction. We need to systematically address missing data values, feature subset selection and the continuous evolution of predictions as the project unfolds, and all of this in the context of data-starvation and noisy data. However, in this paper, we particularly focus on feature subset selection and effort prediction at an early stage of a project. We propose a novel approach of using Grey Relational Analysis (GRA) of Grey System Theory (GST), which is a recently developed system engineering theory based on the uncertainty of small samples. In this work we address some of the theoretical challenges in applying GRA to feature subset selection and effort prediction, and then evaluate our approach on five publicly available industrial data sets using stepwise regression as a benchmark. The results are very encouraging in the sense of being comparable or better than other machine learning techniques and thus indicate that the method has considerable potential.

Keywords*: software project estimation, effort prediction, feature subset selection, empirical evaluation, Grey Relational Analysis, Grey System Theory.*

1. Introduction

Software development, or our understanding of it, is a gradual and evolving process; we know less at the beginning than at the completion of a project. As the software development proceeds, we gather and analyze incomplete information and try to optimize the development process. But the optimization may not be unique, it is modified or changed as more information becomes available. This means that effort prediction methods must also be suitable for dealing with uncertainty and for continuous effort estimating.

At the same time, in this process, the relationship between project effort and the feature subset that affects the effort is unclear or the relational information is incomplete[1].

It can be difficult to select feature subsets and predict project effort with traditional statistical methods or machine learning methods. These methods usually require a large complete sample or data that follow a certain statistical distribution. Unfortunately, current effort prediction methods do not properly take into account these critical characteristics of the software development process and the project data sets. Although there is a great deal of research activity on this topic, a wide range of prediction techniques are being proposed and increasing numbers of empirical studies published, however, no one technique is consistently effective, and the software industry does not have a good track record for accurate cost estimation with typically 75% of projects reporting overruns[2]. Therefore, we should systematically address effort prediction in the context of small, incomplete data sets, this includes missing data imputation, feature subset selection and continuous (or ongoing) effort prediction.

Grey System Theory (GST), a recently developed system engineering theory based on the uncertainty of small samples, was first developed by Deng in 1982 [8]. The system was named by using grey as the colour which indicates the amount of known information in control theory. For instance, if the internal structures and features of a system are completely unknown, the system is usually denoted as a 'black box'. In contrast, 'white' means that the internal features of a system are fully explored. Between white and black, there is a grey system indicating that part of the information is clear, while another part is still unknown.

GST has certain distinct advantages, such as having simple processes to study complex systems and providing reliable analysis results, and allowing us to utilize only a few known data to establish an analysis model. In the context of data-starvation, GST is known to be effective and has been successfully and widely applied to address the real world problems in image processing [15], mobile communication [26], machine vision inspection [13], decision making [17], stock price prediction [27], and system control [11]. GST

*Xi'an Jiaotong University, P.R. China - qbsong@mail.xjtu.edu.cn
†Brunel University, UK - martin.shepperd@brunel.ac.uk
‡Brunel University, UK - carolyn.mair@brunel.ac.uk

[1] Not all features (variables) that are available are useful and some may not only be redundant, they may positively hinder the prediction task. Removing such features is known as feature subset selection.

[2] See for example Moløkken and Jørgensen [20].

is not based on statistical theory, thus only requires a limited amount of data to estimate the behaviour of an uncertain system. We explore the GST-based approach to address project effort prediction questions.

In this paper, we focus on feature subset selection and effort prediction with between-project data sets at an early stage of a development process and leave the imputation and evolutionary aspects of prediction for future work. Feature subset selection is the process of identifying and removing as many irrelevant and redundant features as possible from an original feature set for the purpose of providing better prediction accuracy. It has been widely used to obtain predictive features without creating new features based on transformations or combinations of the original feature set. However, most of the feature subset selection methods are for classification tasks, and can give poor results when the sample size is small [12]. At the same time, software project effort is a continuous valued feature that makes classification methods generally inappropriate. Therefore, we seek to improve upon traditional feature subset selection methods. By analyzing the influence of various features on the continuous values of project effort, GST can help us gain the most predictive feature subset from a small data set.

Grey Relational Analysis is a method of GST. We propose a Grey Relational Analysis based software projeCt Effort prediction (GRACE) method including feature subset selection. Our experiments, using five publicly available real-world data sets, explore the advantages this new approach might offer relative to existing methods.

The remainder of the paper is organized as follows. In the next section we present related work on software effort prediction. This is followed by the introduction of the Grey Relational Analysis (GRA) method and the GRA based effort prediction method GRACE. After that, we describe the data sets, the validation method, and the evaluation criteria we used. The results follow with a concluding discussion and suggestions for further work.

2. Related Work

Machine learning methods have been used to predict software effort since the 1990s. A primary advantage is that they are nonparametric and adaptable, and many of them make no or minimal assumptions about the form of the function under study [25]. Of course, there are also problems, not least that they are difficult to configure to new situations and there is little theory so that their application often becomes a matter of trial and error. The most commonly used methods include case based reasoning (CBR), regression and decision trees, and artificial neural networks (ANN).

Mukhopadhyay et al. [21] presented early work using CBR. They used Kemerer's data set [16] and found that their analogy-based model Estor outperformed COCOMO [2]. Shepperd and Schofield [24] compared an analogy based method with stepwise regression (SWR) on nine different industrial data sets and report that in all cases analogy equalled or outperformed SWR. Finnie and Witting [10] compared CBR with different regression models using function points (FPs) and ANNs. They report that CBR outperformed regression models based on function points. On the other hand, Briand et al. [5] and Jørgensen et al. [14] obtained conflicting results where the regression model generated significantly better results than a CBR approach. Finnie and Witting [10] also found ANN outperformed CBR.

Regression and decision trees are another method that has been used to predict software effort. Briand et al. [4] compared the optimized set reduction (OSR) strategy with COCOMO and SWR. They used the COCOMO81 data set [3] as a training set and the Kemerer data set as a test set. OSR outperformed COCOMO and SWR. Srinivasan and Fisher [25] illustrated the use of CARTX to predict software effort. They also used the COCOMO81 data set for training and the Kemerer data set for testing. They report that CARTX outperformed COCOMO and SLIM [22].

Witting and Finnie [28] report the use of back propagation (BP) artificial neural networks (ANN) to predict software effort. Although the results are encouraging, unfortunately, the data sets were large and only a small number of projects were used for testing. Srinivasan and Fisher [25] found BP neural networks outperformed regression trees. Samson et al. [23] compared an ANN method with linear regression using the COCOMO81 data set. Although the ANN method outperformed the linear regression method, both methods performed badly.

For a more detailed review see Briand and Wieczorek [6]. Nonetheless, even this brief introduction to machine learning methods for software effort prediction reveals that the results are quite mixed. Apart from using different data sets and different experimental methods, these methods tend to borrow from other disciplines where large data sets are the norm or data that follow a certain statistical distribution are necessitated. Our GRACE method is quite different from these methods; it is at least an alternative method for software effort prediction.

3. GRA Based Software Effort Prediction

3.1 Grey Relational Analysis

GRA is based on the conceps of Grey Space. Hence, in this subsection, we respectively introduce the basic concepts and the specific analysis method.

3.1.1 Basic Concepts

Factor space. Let $p(X)$ be a theme characterized by a factor set X, and $\mathcal{Q}$ be an influence relation, $\{p(X); \mathcal{Q}\}$ is a factor space. The factor space $\{p(X); \mathcal{Q}\}$ has the following properties:

- Existence of key factors, for example, the key factors for a basketball player can be height and rebound.
- That the number of factors are limited and countable.
- Factor independence, so each factor must be independent.
- Factor expansibility, so for example, besides the height and rebound, weight can be added as another factor.

Comparable series. Suppose $x_i = \{x_i(1), x_i(2), \ldots, x_i(m)\}$, where $i = 0, 1, 2, \ldots, n \in N; m \in N$, is a series. This series is said to be comparable if, and only if, the following conditions are met:

- Dimensionless. Factors must be processed to become non-dimensional, irrespective their units and scales.
- Scaling. The factor value $x_i(k)(k = 1, 2, \ldots, m)$ of different series $x_i(i = 0, 1, 2, \ldots, n)$ must be at the same level.
- Polarization. The factor value of $x_i(k)$ of different series $x_i(i = 0, 1, 2, \ldots, n)$ are described in the same direction.

Grey relational space. If all the series in a factor space $\{p(X); \mathcal{Q}\}$ are comparable, the factor space is a grey relational space which is denoted as $\{p(X); \Gamma\}$. In a grey relational space $\{p(X); \Gamma\}$, X is a collection of data series $x_i(i = 0, 1, \ldots, n)$, in which $x_i = \{x_i(1), x_i(2), \ldots, x_i(k)\}$, is the series; and $k = 1, 2, \ldots, m$, are the factors. Γ, which is the grey relation map set and based on geometrical mathematics, has the following four properties [30].

- Normality:

$$0 \leq \Gamma(x_i(k), x_j(k)) \leq 1, \forall i, \forall j, \forall k$$

$$\Gamma(x_i, x_j) = 1 \Leftrightarrow x_i \equiv x_j,$$

$$\Gamma(x_i, x_j) = 0 \Leftrightarrow x_i \cap x_j \in \emptyset.$$

- Symmetry:

$$\forall x_i, \forall x_j \in X,$$

$$\Gamma(x_i, x_j) = \Gamma(x_j, x_i) \Leftrightarrow X = \{x_i, x_j\}.$$

- Entirety:

$$\forall x_i, \forall x_j \in X = \{x_\sigma | \sigma = 0, 1, \ldots, n\}, n \geq 2,$$

$$\Gamma(x_i, x_j) \overset{often}{\neq} \Gamma(x_j, x_i).$$

  This means $\Gamma(x_i, x_j) \neq \Gamma(x_j, x_i)$ almost holds if and only if it is a multi–input and single–output system, given x_i the output series and x_j the input series of a system.

- Proximity: $\Gamma(x_i(k), x_j(k))$ increases as $\Delta(k) = |x_i(k) - x_j(k)|$ decrease for $\forall k \in \{1, 2, \ldots, m\}$.

3.1.2 Method

GRA is used to quantify all the influences of various factors and the relationship among data series that is a collection of measurements. It is based on the grey relational model which is a kind of influence measurement model. This model is used to measure the trend relationship between two systems or two elements of a system, and this relationship can change with time. The influence measurement is referred to as relational grade. For a given system, if the developing trend between two elements of the system tend towards concordance, the relational grade is considered to be large; otherwise, it is regarded as small. Therefore, the GRA is founded upon measuring the similarity of emerging trends among factors or data series.

Before identifying the emerging trend, GRA removes anomalies associated with different measurement units and scales by the normalization of raw data. The raw data can be transformed into dimensionless forms either by initial-value processing or average-value processing. Which type of processing to be used depends on the nature of data. Generally, average-value processing is applied to data series that are independent of time sequence, and initial-value processing is more appropriate for data that vary with time.

Initial-value processing divides the elements in each series by the corresponding first component:

$$x_i(k) = \frac{x_i(k)}{x_0(k)}, \tag{1}$$

where $i = 0, 1, 2, \ldots, n$ and $k = 1, 2, \ldots, m$. Whereas average-value processing makes use of the average value of elements in all series as the divisor:

$$x_i(k) = \frac{x_i(k)}{\frac{1}{n+1}\sum_{j=0}^{n} x_j(k)}, \tag{2}$$

where $i = 0, 1, 2, \ldots, n$ and $k = 1, 2, \ldots, m$.

GRA uses the grey relational coefficient to describe the trend relationship between an objective series and a reference series at a given point in a system. Suppose $x_0 \in X$ is

the reference series and $x_1, x_2, \ldots, x_n \in X$ are the objective series, the grey relational coefficient $\gamma(x_0(k), x_i(k)) \in \Gamma$ between the reference series x_0 and the objective series $x_i (i = 1, 2, \ldots, n)$ at point $k \in \{1, 2, \ldots, m\}$ was defined by Deng as follows:

$$\gamma(x_0(k), x_i(k)) = \frac{\Delta_{min} + \zeta\Delta_{max}}{\Delta_{0,i}(k) + \zeta\Delta_{max}}, \quad (3)$$

where $\Delta_{0,i}(k) = |x_0(k) - x_i(k)|$ is the difference of the absolute value between $x_0(k)$ and $x_i(k)$; $\Delta_{min} = min_{\forall j} min_{\forall k} |x_0(k) - x_j(k)|$ is the smallest value of $\Delta_{0,j} \forall j \in \{1, 2, \ldots, n\}$; $\Delta_{max} = max_{\forall j} max_{\forall k} |x_0(k) - x_j(k)|$ is the largest value of $\Delta_{0,j} \forall j \in \{1, 2, \ldots, n\}$; and ζ is the distinguishing coefficient, $\zeta \in (0, 1]$, expressed as the contrast between the background and the object to be tested. The ζ value will change the magnitude of $\gamma(x_0(k), x_i(k))$, a best ζ value should be determined to meet the system need.

As there are too many relational coefficients to be compared directly, further data reduction makes use of average-value processing to convert each series' grey relational coefficients at all points into its mean. The mean is also referred to as the grey relational grade.

The grey relational grade $\Gamma(x_0, x_i)$ between an objective series $x_i (i = 1, 2, \ldots, n)$ and the reference series x_0 was defined by Deng as follows:

$$\Gamma(x_0, x_i) = \frac{1}{n} \sum_{k=1}^{n} \gamma(x_0(k), x_i(k)). \quad (4)$$

However, in practice the influence of each factor on the system is not exactly the same, for example, Eqn. 4 can be modified as follows:

$$\Gamma(x_0, x_i) = \sum_{k=1}^{n} \beta_k \gamma(x_0(k), x_i(k)), \quad (5)$$

where β_k is the normalized weight for point k, and $\sum_{k=1}^{n} \beta_k = 1$. It can be set to any value that reflects the relative importance of all the series $x_i (i = 1, 2, \ldots, n)$ to series x_0 at point k.

By narrowing the interval between Δ_{min} and Δ_{max} to be within [0,1] and eliminating the influence of ζ, Wu [29] proposed directly calculating the grey relational grade instead of firstly computing the relational coefficient. Wu's grey relational grade is as follows:

$$\Gamma(x_0, x_i) = \frac{\Delta_{min} + \Delta_{max}}{\bar{\Delta}_{0,i} + \Delta_{max}}, \quad (6)$$

where $\bar{\Delta}_{0,i} = \sqrt{\frac{1}{n} \sum_{k=1}^{n} [\Delta_{0,i}]^2}$.

3.2 Selecting Feature Subsets with GRA

From the properties of Γ (see Subsection 3.1.1 for details) we know that if a series shows a higher influence on the output than others, then the series can be considered to be more important to the output. We view the feature data as series and software effort as output and apply GRA to select the optimal feature subset for software effort prediction.

Let $\mathcal{D} = \{x_1, x_2, \ldots, x_i, \ldots, x_n\}$ be a software project data set, and $x_i = \{x_i(1), x_i(2), \ldots, x_i(m), e_i\}$ where $i = (1, 2, \ldots, n \in N)$ is a project. For each project $x_i \in \mathcal{D}$, $x_i(1), x_i(2), \ldots, x_i(m)$ are the feature data of the project, and e_i is the corresponding software effort. The software project data set $\mathcal{D}$ can be represented in the form of a matrix as follows:

$$\mathcal{D} = \begin{pmatrix} x_1(1) & x_1(2) & \ldots & x_1(m) & e_1 \\ x_2(1) & x_2(2) & \ldots & x_2(m) & e_2 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ x_i(1) & x_i(2) & \ldots & x_i(m) & e_i \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ x_n(1) & x_n(2) & \ldots & x_n(m) & e_n \end{pmatrix}. \quad (7)$$

When using GRA to select the optimal feature subset, the following procedure is used:

Step 1: Data series construction. View each of the column vectors of matrix $\mathcal{D}$ as a data series. We obtain a total of $m + 1$ series. These series are: $x_1 = \{x_1(1), x_2(1), \ldots, x_n(1)\}$, $x_2 = \{x_1(2), x_2(2), \ldots, x_n(2)\}$, ..., $x_m = \{x_1(m), x_2(m), \ldots, x_n(m)\}$, and $x_0 = \{e_1, e_2, \ldots, e_n\}$. Of course, x_0 is considered to be the reference series and $x_1, x_2, \ldots, x_m$ are regarded as the corresponding objective series.

Step 2: Normalization. Normalize the each series $x_0, x_1, x_2, \ldots, x_n$, thus each element has the same degree of influence and the method cannot be affected by the choice of units and scales.

Step 3: Grey relational grade calculation. Compute the grey relational grades between the reference series x_0 and each of the objective series $x_1, x_2, \ldots, x_n$ respectively by both Eqns. 3 and 4 or directly by Eqn. 6.

Step 4: Feature subset selection. Sort the grey relational grades. The features with higher score comprise the optimal feature subset. These scores will be normalized and then used as the weights of corresponding features when computing the grey relational grade using Eqn. 5 for software effort prediction purpose.

3.3 Predicting Software Effort with GRA

GRA was originally used to measure the relationship among data series. We view project data as a series and apply this technique here to select projects that exhibit a stronger impact on the reference project i.e. the one for which we wish to predict effort.

The GRA based software effort prediction method GRACE first constructs data series from a project data set and measures the grey relational grade among these series and then uses the most influential projects to predict the effort for a new project. The procedure for GRACE is as follows.

Step 1: Data series construction. View each of the row vectors of matrix $\mathcal{D}$ as a data series, and obtain a total of n series. These series are: $x_0 = \{x_1(1), x_1(2), \ldots, x_1(m)\}$, $x_1 = \{x_2(1), x_2(2), \ldots, x_2(m)\}$, $x_2 = \{x_3(1), x_3(2), \ldots, x_3(m)\}$, ..., and $x_n = \{x_n(1), x_n(2), \ldots, x_n(m)\}$. Of course, x_0 is supposed to be the reference series whose effort needs to be predicted and $x_1, x_2, \ldots, x_m$ are regarded as the corresponding objective series.

Step 2: Normalization. Use the reference series x_0 to normalize the corresponding objective series $x_1, x_2, \ldots, x_n$, thus each feature has the same degree of influence and the method cannot be affected by the choice of units and scales.

Step 3: Grey relational grade calculation. Compute the grey relational grades between the reference series x_0 and each of the objective series $x_1, x_2, \ldots, x_n$ respectively by both Eqn. 3 and Eqn. 4 (or Eqn. 5) or only by Eqn. 6.

Step 4: Effort prediction. Aggregate the most influential *k* projects' effort as the prediction of the new project. It is defined as follows:

$$\hat{\mathcal{E}} = \sum_{i=1}^{k} w_i \times \mathcal{E}_i,$$

where $w_i = \frac{\Gamma(x_0, x_i)}{\sum_{j=1}^{k} \Gamma(x_0, x_j)}$, $\mathcal{E}_i$ is the effort of the ith most influential project.

However, in this procedure, there is an outstanding problem, i.e. the selection of ζ and k. Since our purpose is predicting software effort from small data sets, we preset $K = \{1, 2, 3, 4, 5\}$. For each $k \in K$, we learn ζ from the objective series. Specifically, by changing ζ from 0 to 1 with an increment of 0.01, we predict effort for each of the projects in the objective series with the pair of (k, ζ). By analyzing the relationship between the relative errors and these two parameters, we obtain the optimal k and ζ which are finally used to predict software effort.

We have developed the corresponding software effort prediction tool GRACE. In the current version, we have implemented both Deng's (Eqns. 3 and 4) and Wu's Grey relational grade calculation methods (Eqn. 6). The experimental results show that Deng's method is more accurate. At the same time, for Deng's method, we have also compared Eqns. 4 and 5. We found that the former has higher precision.

Name	Projects	Features	Description	Source
Albrecht (ALBR)	19	8	IBM DP Services projects	[1]
COCOMONASA (CMNS)	60	17	Aerospace applications from 1980s to 1990s	[19]
COCOMO81 (CM81)	63	17	Business, scientific, and system software projects	[3]
Desharnais (DESH)	77	9	Canadian software house - commercial projects	[9]
Kemerer (KEME)	15	17	Large business applications	[16]

Table 1. Data sets used in the experiments

4. Experimental Method

4.1 Data Sources

Different researchers usually use different data sets to test their methods which makes it hard to compare their results. In order to compare our results with other researchers' results, and allow other researchers to confirm our results, we used five publicly available data sets as our data source. Most of these data sets have been used by more than one researcher to evaluate software cost estimation models. For example, COCOMO81 is the data set that was used by Boehm [3] to build the COCOMO model and also was used by Briand et al. [4], Srinivasan and Fisher [25], and Samson et al. [23] to compare different effort prediction methods; the Kemerer data set was coded by Kemerer [16] using the same attributes as Boehm and later was used by Briand et al. [4], Srinivasan and Fisher [25], and Shepperd and Schofield [24]; the Albrecht data set actually is the IBM DP Services data but was first used by Albrecht and Gaffney [1] and was also used by Shepperd and Schofield [24] to validate software size and effort estimation methods.

Tables 1 and 2 summarize these data sets and show that application domains range from business, science, and technology to system software. The project size ranges from 1.98 KSLOC to 1150 KSLOC (or from 62 FPs to 1116 FPs), the project effort ranges from 0.5 person months to 11400 person months (or from 546 person hours to 23940 person hours), and the number of projects ranges from 15 to 77. Therefore, the data sets are quite diverse.

4.2 Validation method

In practice, software project data sets are frequently small. The holdout method makes inefficient use of the data, because generally a third of the data set is hidden from the prediction method. In order to more efficiently use data

Dataset	Size (KSLOC/FP)	Effort (PM/PH)
ALBR	Mean = 61.08, Min = 3, Max = 318	Mean = 21.88, Min = 0.5, Max = 105.2
CMNS	Mean = 74.59 Min = 2.2, Max = 423	Mean = 406.41, Min = 8.4, Max = 3240
CM81	Mean = 77.21, Min = 1.98, Max = 1150	Mean = 683.32, Min = 5.9, Max = 11400
DESH	Mean = 282.39 FPs, Min = 62, Max = 1116	Mean = 4833.91 PHs, Min = 546, Max = 23940
KEME	Mean = 184.37, Min = 39, Max = 450	Mean = 219.25, Min = 23.2, Max = 1107.31

Table 2. Descriptive statistics for project size and effort

and cover all projects, we used the *jack knife* methodology to validate the proposed GRACE method. Specifically, for each of the n projects of a given data set, we predicted its effort with (k, ζ) which were learned from the remaining $n-1$ projects. The evaluation measures defined in subsection 4.3 are then computed over the n predictions.

For the purpose of evaluating the performance of the proposed GRACE method, we compared it with the stepwise regression method. For stepwise regression, the prediction models were generated using the entire data set. This means the results are likely to be slightly biased in favour of the regression models. In addition, we also compared our results with ANNs, linear regression, CARTX, and Analogy where the accuracy indicators and evaluation procedures permit.

4.3 Evaluation Criteria

We used the Bias, which refers to how far the average statistic lies from the parameter it is estimating, to assess the error which arises when estimating software effort. It is defined as follows:

$$Bias_i = \frac{\mathcal{E}_i - \hat{\mathcal{E}}_i}{\mathcal{E}_i} \times 100 \tag{8}$$

where $\hat{\mathcal{E}}_i$ is the prediction of effort $\mathcal{E}_i$.

A common criterion for evaluating software effort prediction methods is the Magnitude of Relative Error (MRE). For a project i whose effort is predicted, the corresponding MRE_i is defined as follows:

$$MRE_i = |Bias_i|, \tag{9}$$

By averaging MRE_i over multiple projects n, Mean MRE (MMRE) is obtained:

$$MMRE = \frac{1}{n}\sum_{i=1}^{i=n} MRE_i. \tag{10}$$

As MMRE is sensitive to individual predictions with excessively large MREs, we also use the median of MREs for the n projects (MdMRE), which is less sensitive to extreme values, as another measure. For both MMRE and MdMRE, a higher score means worse prediction accuracy.

When using MRE as a measure of prediction accuracy, we suppose the error is proportional to the size of the project. Thus, PRED(l) is usually used as a complementary criterion. This is defined as follows:

$$PRED(l) = \frac{k}{n} \times 100 \tag{11}$$

where k is the number of projects where $\text{MRE}_i \leq l\%$, and n is the number of all predictions. Unlike both MMRE and MdMRE, for PRED(l), a higher score implies better prediction accuracy.

5. Experimental Results

Tables 3 and 4 contain the accuracy of respective methods in terms of MMRE, MdMRE, PRED(25), and Bias. From Table 3 we observe that the MMREs and MdMREs of GRACE are much smaller than those of SWR except for the Desharnais data set. Here the two methods obtained almost the same MMRE and the SWR's MdMRE is slightly better. Excluding the Desharnais data set, MMREs have been reduced by at least 67.95% and MdMREs have been reduced by at least 21.36%. This suggests that GRACE tends to allow more accurate predictions than SWR.

From Table 4 we observe that PRED(25) values are more mixed. GRACE has better values for 3 out of 5 data sets though this may in part be due to the arbitrary effect of setting the theshold to 25%. We also observe that the biases of GRACE for all data sets are no more than 19%, and are much smaller than those of SWR excepting the Desharnais data set. In 4 out of 5 data sets, GRACE over estimated effort. This means there may be scope for improving accuracy.

At the same time, the use of the same data sets and the same experimental method — the jack knife — allow us to compare GRACE with Analogy [24], ANN [23], linear regression [23], and CARTX [25]. Table 5 contains the published results. The results show that compared with Analogy, the MMRE and PRED(25)[3] have been improved by GRACE by 2.82% and 37.3% respectively with the Albrecht data set, by 22.14% and -16.67% respectively with the Desharnais data set, and by 5.11% and -33.33% respectively with the Kemerer data set. The results show that compared with ANN, linear regression, and CARTX, the MMRE [4] has been improved by GRACE by 82.23%,

[3]For the Analogy method, only MMRE and PRED(25) are available.
[4]For these three methods, only MMRE is available.

Dataset	MMRE(%) of		MdMRE(%) of	
	GRACE	SWR	GRACE	SWR
ALBR	60.25	103.02	21.35	45.02
CMNS	32.88	102.59	28.38	49.98
CM81	76.09	1540.84	60.52	445.13
DESH	49.83	49.63	33.93	30.94
KEME	58.83	102.61	46.94	59.69

Table 3. MMRE and MdMRE of GRACE and SWR with different data sets

Dataset	PRED(25)(%) of		Bias(%) of	
	GRACE	SWR	GRACE	SWR
ALBR	52.63	42.11	-12.13	50.38
CMNS	46.67	35.00	17.34	34.15
CM81	20.63	6.67	-18.89	578.18
DESH	30.00	32	-16.52	-14.99
KEME	26.67	33.33	-7.07	-47.54

Table 4. PRED(25) and Bias of GRACE and SWR with different data sets

85.39%, and 39.23% respectively with the COCOMO81 data set.

We are also aware that Burgess and Lefley [7] using a Genetic Programming approach obtained a better result of MMRE=44.5% for the Desharnais data set. Unfortunately, a different experimental method — holdout — was used, and the model was tested on only 22% projects. The test data set is small, as it is common to designate 1/3 of the data as the test set. Therefore, it is not directly comparable. Generally, it is not recommended to compare methods with the same data sets but with different experimental methods. For example, Mair et al.[18] report a 57% of MMRE with the Analogy method with the Desharnais data set. Unfortunately, for the same data set, Shepperd and Schofield's [24] result is 64%. The reason is the former used a holdout method, whereas the experimental method used by the latter is the jack knife. Another example is both Srinivasan and Fisher [25] and Briand et al.[4] used the regression and decision trees method and the same data sets. More importantly they used the same experimental method – holdout, but different training sets and test sets. Unfortunately, they obtained very different results. Srinivasan and Fisher's result is 364% in terms of MMRE and Briand et al. obtained a 94% of MMRE.

To summarize, in 4 out of 5 data sets, GRACE is more accurate than SWR and almost as good as SWR for another data set; in all the comparable cases, GRACE tends to be more accurate than the other compared methods at least for the given data sets.

Dataset	MMRE (%) of	
ALBR	GRACE	60.25
	Analogy	62
CM81	GRACE	76.09
	ANN	428.11
	Lin.RegR.	520.71
	CARTX	125.2
DESH	GRACE	49.83
	Analogy	64
KEME	GRACE	58.83
	Analogy	62

Table 5. Comparison with published results with the same dataset and the same experimental method

The boxplot is a type of graph that is used to show the shape of the distribution, its central value, and variability. We used boxplots to explore the spreads of absolute residuals of prediction accuracy for GRACE and SWR methods with all five data sets.

Figs. 1, 2, 3, 4, and 5 are the boxplots of absolute residuals of prediction accuracy for GRACE and SWR with the Albrecht, COCOMONASA, COCOMO81, Kemerer, and Desharnais data sets respectively.

From Fig. 1 we observe that: 1) the smaller box of SWR indicates reduced variability of absolute residuals; 2) the median of GRACE is far smaller than that of SWR. This means that at least half of the predictions of GRACE are more accurate than those of SWR.

From Fig. 2 we observe that: 1) the range of absolute residuals of GRACE excluding outliers is smaller than that of for SWR, which means smaller variance; 2) the box of GRACE overlays the lower whisker. This reveals that the data are probably skewed towards the lower end of the scale, and also shows more accurate results; 3) the smaller box of GRACE indicates reduced variability of absolute residuals; 4) the median of GRACE is smaller than that of SWR.

From Fig. 3[5] we observe that: 1) the range of absolute residuals of GRACE excluding outliers is much smaller than that of SWR, which means smaller variance; 2) the box of GRACE overlays the lower whisker. This reveals that the data are probably skewed towards the lower end of the scale, and also means more accurate results; 3) the smaller box of GRACE indicates reduced variability of absolute residuals; 4) the median of GRACE is much smaller than that of SWR.

From Fig. 4 we observe that 1) the range of absolute residuals of GRACE excluding outliers is somewhat smaller than that of SWR, which means smaller variance; 2) the smaller box of GRACE indicates reduced variability

[5]The scale has been truncated so the even worse outliers for SWR are not visible.

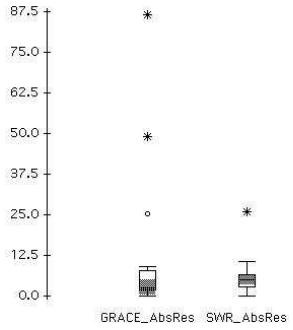

Figure 1. Absolute residuals of prediction accuracy for GRACE and SWR with the Albrecht data set

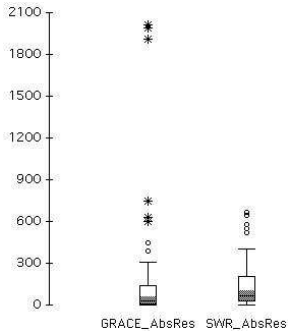

Figure 2. Absolute residuals of prediction accuracy for GRACE and SWR with the COCOMONASA data set

of absolute residuals; 3) yet again the median of GRACE is smaller than that of SWR.

From Fig. 5 we observe that 1) similar sized boxes meaning similar variability of the non-outlier absolute residuals; 2) almost the same median means that at least half of the predictions are at the same accurate level; 3) there are more extreme outliers for the GRACE method.

Figs. 1 - 5 are all characterized by a few extreme outliers, but more so for GRACE. After investigating the data sets, we found that these outliers are related to the noise in the corresponding data sets. The reason why GRACE is more sensitive to noise is that SWR uses all the projects' data. That is, it uses the models learned from projects to predict themselves; but GRACE uses $n-1$ projects to predict the unseen project. Therefore, if a 'new' project itself is noise, the corresponding result is an outlier as well. Of course, any machine learning method can suffer from "garbage in garbage out (GIGO)" and we cannot learn useful knowledge from garbage. It may also be useful in the future to explore when a prediction cannot sensibly be made.

Thus, quality data is a precondition for obtaining quality knowledge, and data quality is a constant issue for machine learning and data mining researchers. This reminds us that before predicting software effort, we should firstly remove outliers. Alternatively one might consider dividing highly distinct projects into separate data sets.

To summarize, the boxplots of absolute residuals of prediction accuracy for GRACE and SWR with five data sets show GRACE outperformed SWR.

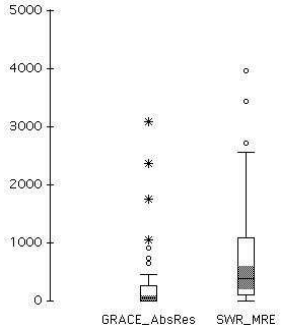

Figure 3. Absolute residuals of prediction accuracy for GRACE and SWR with the COCOMO81 data set

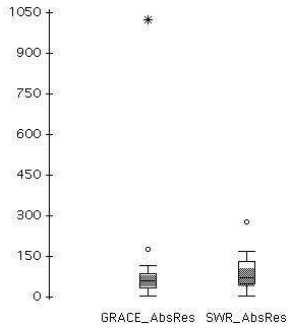

Figure 4. Absolute residuals of prediction accuracy for GRACE and SWR with the Kemerer data set

6. Conclusions and Future Work

In this paper, we have proposed a novel approach of using Grey Relational Analysis of Grey System Theory to address feature subset selection and software effort prediction at an early stage of a software development process.

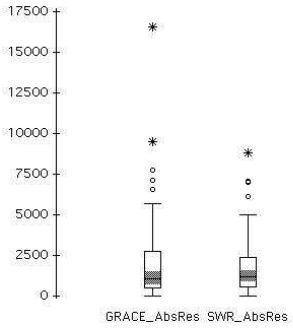

Figure 5. Absolute residuals of prediction accuracy for GRACE and SWR with the Desharnais data set

Using five publicly available data sets, we have compared the proposed method with prediction models based on stepwise regression, and when available, also with Analogy, artificial neural networks, linear regression, and regression and decision trees. The results show that the proposed method outperformed other methods in terms of MMRE, MdMRE, PRED(25) and Bias for the majority (4 out of 5) of data sets we used. This is very encouraging and indicates that the method has considerable potential.

However, this work is just a first step in using GST to systematically address software effort prediction questions. Therefore, future work, apart from improving our GRACE method, includes developing a GST-based missing data imputation method. This is because many software project data sets contain missing data, and all the project data are incomplete in the early stages of development processes. At the same time, almost all the project prediction methods require complete data sets. Therefore, to use these methods we must firstly make data sets complete either through missing data ignoring techniques or through missing data imputation techniques. However, the former techniques make small software project data sets more smaller, and further hinder effort prediction. GST is therefore proposed for processing small data sets since by measuring the relationship between data series, we can find the appropriate values for missing data.

It may also be fruitful to develop a GST based project effort continuous prediction method. Remember that software processes take place over time. This calls for continuous effort prediction. Taking these factors into account, our effort prediction method will be based on GST that will dynamically predict project effort at different time-points during software projects capitalising upon updated or new information.

Thus overall, we are of the opinion that using Grey System Theory for software project effort prediction should be further explored. We have demonstrated accuracy levels that are broadly comparable or better than regression models or other machine learners. We have also argued that there is more of the theory that can be usefully exploited to deal with missing data values and to support ongoing as opposed to one-shot prediction. However, independent validation of these ideas is essential before it will start to be possible to generalize from our results to other data sets.

Acknowledgments

This work was funded by the UK Engineering and Physical Sciences Research Council under grant GR/S45119. The authors would like to thank the three anonymous reviews for their insightful and helpful comments.

References

[1] Albrecht, A. and Gaffney, J. “Software Function, Source Lines of Code, and Development Effort Prediction”. *IEEE Transactions on Software Engineering*, Vol.9, No.6, pp639–648, 1983.

[2] Boehm, B.W., “Software Engineering Economics”. *IEEE Transactions on Software Engineering*, Vol.10, No.1, pp4–21, 1984.

[3] Boehm, B.W., *Software Engineering Economics*, Prentice-Hall, 1981.

[4] Briand, L., Basili, V., and Thomas, W., “A Pattern Recognition Approach for Software Engineering Data Analysis”. *IEEE Transactions on Software Engineering*, Vol.18, No.11, pp931–942, 1992.

[5] Briand, L., T. Langley and I. Wieczorek. “Using the European Space Agency data set: a replicated assessment and comparison of common software cost modeling techniques”. *22ndInternational Conference on Software Engineering*, pp377–386, Limerick, Ireland: Computer Society Press, 2000.

[6] Briand, L.C. and Wieczorek, I.,“Resource Modeling in Software Engineering,” in *Encyclopaedia of Software Engineering*, J. J. Marciniak, Ed., 2nd ed. New York: John Wiley, 2002.

[7] Burgess, C.J. and Lefley, M., “Can genetic programming improve software effort estimation? A comparative evaluation”. *Information and Software Technology*, Vol.43, No.14, pp813–920, 2001.

[8] Deng, J. L., “Control problems of grey system”. *System and Control Letters*, vol. 1, pp288–94, 1982.

[9] Desharnais, J. M., “Analyse statistique de la productivitie des projets informatique a partie de la technique des point des fonction”. *Masters Thesis*, University of Montreal, 1989.

[10] Finnie, G.R and Witting, G.E., “A comparison of software effort technique: using function points with neural networks, case based reasoning and regression models”. *Journal Systems and Software*, vol.39, pp281–289, 1997.

[11] Huang, S. J., Huang, C. L., “Control of an inverted pendulum using grey prediction model”. *IEEE Transactions on Industry Applications*, vol.36, no.2, pp452–458, 2000.

[12] Jain, A. and Zongker, D., “Feature Selection: Evaluation, application and small sample performance”. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol.19, no.2, pp153–158, 1997.

[13] Jiang, B.C., Tasi, S. L., and Wang, C. C., “Machine vision-based gray relational theory applied to IC marking inspection”. *IEEE Transactions on Semiconductor Manufacturing*, vol.15, no.4, pp531–539, 2002.

[14] Jørgensen, M., Indahl, U., and Sjøberg, D.,“Software effort estimation by analogy and ‘regression toward the mean’ ”. *Journal of Systems and Software*, Vol.68, No.3, pp253–262, 2003.

[15] Jou, J. M., Chen, P. Y., and Sun, J. M., “The gray prediction search algorithm for block motion estimation”. *IEEE Transactions on Circuits and Systems for Video Technology*, vol.9, no.6, pp843–848, 1999.

[16] Kemerer, C.F., “An empirical validation of software cost estimation models”. *Comm. ACM*, vol.30, no.5, pp416–429,1987.

[17] Luo, R. C., Chen, T. M., and Su, K. L., “Target tracking using a hierarchical grey-fuzzy motion decision-making method”. *IEEE Transactions on Systems, Man and Cybernetics, Part A*, vol.31, no.3, pp179–186, 2001.

[18] Mair, C., Kadoda, G., Lefley, M., Phalp, K., Schofield, C., Shepperd, M. and Webster, S., “An investigation of machine learning based prediction systems”. *Journal of Systems and Software*. Vol.53, No.1, pp23–29, 2000.

[19] Menzies, T., Port, D., Chen, Z., Hihn, J. and Stukes, S., ”Validation Methods for Calibrating Software Effort Models”. *The 27th International Conference on Software Engineering*. St Louis Missouri, USA, 15–21 May, 2005.

[20] Moløkken, K. and Jørgensen, M., “A review of surveys on software effort estimation”, *2nd Intl. Symp. on Empirical Software Engineering*, pp223-230: IEEE Computer Society, 2003.

[21] Mukhopadhyay, T., Vicinanza, S.S., and Prietula, M.j. “Examining the feasibility of a case-based reasoning model for software effort estimation.” *MIS Quarterly*, vol.16, pp155–171, june, 1992.

[22] Putnam, L. H., “A general empirical solution to the macro software sizing and estimation problem”. *IEEE Transactions on Software Engineering*, vol.4, no.4, pp345–381,1978.

[23] Samson, B., Ellison, D., and Dugard, P., “Software coast estimation using an Albus Perceptron (CMAC)”. *Information and Software Technology*, vol.39, nos.1/2, 1997.

[24] Shepperd M. and Schofield C., “Estimating Software Project Effort Using Analogies”. *IEEE Transactions on Software Engineering*, Vol.23, No.12, pp736–743, 1997.

[25] Srinivasan, K. and Fisher, D., “Machine Learning Approaches to Estimating Software Development Effort”. *IEEE Transactions on Software Engineering*, Vol.21, No.2, pp126-137, 1995.

[26] Su, S. L., Su, Y. C., Huang, J. F., “Grey-based power control for DS-CDMA cellular mobile systems”. *IEEE Transactions on Vehicular Technology*, vol.49, no.6, pp2081–2088, 2000.

[27] Wang, Y. F. “On-demand forecasting of stock prices using a real-time predictor”. *IEEE Transactions on Knowledge and Data Engineering*, vol.15, no.4, pp1033–1037, 2003.

[28] Witting, G.E. and Finnie, G.R, “Using artificial neural networks and function points to estimate 4GL software development effort”. *Australian Journal of Information Systems*,vol.1, no.2, pp87-94, 1994.

[29] Wu, J.H. and Chen C. B., “An alternative form for Grey Relational Grade”. *Journal of Grey System*, vol.11, no.1, pp7–11,1999.

[30] Wu, H. S., Deng, J. L., and Win, K. L., *Introduction to Grey Theory* (in Chinese). Taipei, Taiwan: Gao-Li Co., 1996.