# Self-Organising Maps (SOMs) in Software Project Management

Lois Dai

A thesis submitted to
Auckland University of Technology
in partial fulfillment of the requirements for the degree of
Master of Computer and Information Sciences (MCIS)

2011

School of Computing and Mathematical Sciences
**Primary Supervisor: Professor Stephen MacDonell**
**Co-supervisor: Jim Buchan**

# Table of Contents

# List of Figures

# List of Tables

# Attestation of Authorship

“I hereby declare that this submission is my own work and that, to the best of my knowledge and belief, it contains no material previously published or written by another person (except where explicitly defined in the acknowledgements), nor material which to a substantial extent has been submitted for the award of any other degree or diploma of a university or other institution of higher learning.”

Yours sincerely,

(Lois DAI)

# Acknowledgements

*“The fear of the LORD is the beginning of wisdom: and the knowledge of the holy is understanding.”*

*Proverbs 9:10*

First and foremost, I would like to express my deepest gratitude to my primary supervisor Professor Stephen MacDonell, who has provided me with his valuable guidance and extreme patience at every stage of the writing of this thesis.

I also want to extend my appreciation to my secondary supervisor Mr. Jim Buchan, for his enlightening instructions and impressive kindness that not only guided me throughout the course of my study, but have also encouraged me to continue with my study in the future.

Last but not least, I must thank my family and all friends, especially Mr. Baoyi Liu and Mr. Andy Wong, for their selfless help and support.

## Abstract

Although numerous researchers have devoted much time and effort to the issue, generating a reliable and accurate cost estimate at an early stage of the development life cycle remains a challenge to software engineers. In recent years an increasing number of studies have turned their attention to the employment of machine learning, especially Artificial Neural Networks (ANNs), in performing such estimation activities. A Self-Organising Map (SOM) is a particular type of ANN that utilises a neighbourhood function that can be used as an unsupervised clustering tool. Its ability to project multi-dimensional data into a two-dimensional map makes the SOM appealing to software engineers.

In addition, the vague and ambiguous nature of real world software data demands techniques that can handle fuzziness. Accordingly, researchers have introduced fuzzy logic approaches such as fuzzy sets, fuzzy rules, fuzzy inference and the associated fuzzy clustering techniques into the original area of neural networks. Following a thorough literature review, it was decided that Self-Organising Maps could be an appropriate candidate for estimation in software project management. In order to investigate our hypothesis we build predictive models using Self-Organising Maps and compare them with Linear Regression models. The Fuzzy C-means algorithm is utilized in our study to pre-process ambiguous and vague real world data, which also refines the clustering outcome.

This study presents and analyses the results of three case studies that use data sets from different software projects. The findings indicate that Self-Organising Maps surpass Linear Regression in all three cases (even when noise was introduced), both in terms of generating more accurate estimates and presenting easy-to-understand relationships among the project features, when compared to Linear Regression models. Alternative approaches and extensions are suggested in order to overcome the limitations of the study. Other recommended future study areas include, but are not limited to, exploring alternative approaches to forming Fuzzy Self-Organising Maps (FSOMs),

adopting new versions of the Fuzzy C-means algorithm, and investigating further the sensitivity of SOMs and FSOMs.

# 1 Introduction

## 1.1 Brief Background and Research Objective

Cost estimation in software project management generally refers to the prediction of the personnel effort required in development, and is part of the activity and schedule planning management tasks undertaken by software project managers (Kurbel 2008). Generating reliable and accurate cost estimates at an early stage of the development life cycle is an ongoing and significant challenge for software engineers.

Factors such as developer fatigue, team dynamics, and the likely effect of new techniques and tools are among the variables that may be difficult to model in a quantitative sense, although experienced managers may be able to take these factors into account qualitatively. Such experience and knowledge are clearly vulnerable to loss. That is, if a manager who possesses significant project knowledge leaves an organisation, retaining his or her knowledge about the relationships between factors can be important for the organisation to stay in business. Even if they have recorded this information, managers may not be aware of appropriate methods for leveraging it in terms of effort prediction.

One approach used to address this issue involves developing models based on historical data, by mining trends and patterns to estimate aspects of interest (including effort) based on factors (metrics) as accounting for specification size, developer expertise and experience, and code quality and complexity. However, model development and subsequent calibration are far less practicable for immature organisations which suffer from a lack of such an historical database. Gray & MacDonell (1997) found that fuzzy systems can be applied to software metrics in early estimation where sufficient information for more detailed models is not available or where data is only available in small quantities (or even not at all). They reach this conclusion after comparing a range of modelling techniques that could be suitable for predictive software metric model development, including least squares regression, robust

regression, neural networks, fuzzy systems (adaptive), hybrid neuro-fuzzy systems, and regression trees.

More recently, Berlin, Raz, Glezer & Zviran (2009) compared linear regression and artificial neural networks (ANNs) and found that such methods are characterised by unclear and closed structure that make them complex and opaque. It is therefore difficult for a project manager without specialist mathematical knowledge to understand the common sense underlying the computation processes. The use of fuzzy models, and language close to the domain of project managers, may help to address this issue.

Hsiao, Lin & Chang (2008) proposed a fuzzy membership function approach to transform verbal opinions into numbers. They conducted two experiments to compare the performance of this value-based measure with traditional variance-based methods and an entropy measure. They argued that the fuzzy membership-based consensus measure indeed improves performance, especially when a large number of people are involved in the decision making. To improve the ability of processing of numerical and categorical data in similarity measurement and to decrease uncertainty, Azzeh, Neagu & Cowling (2010) also employed Fuzzy set theory with Grey Relational Analysis (GRA) as a new formal Estimation by Analogy (EA) model. These studies serve to illustrate that fuzzy logic modelling may assist managers when producing predictions for software projects.

MacDonell & Gray (2003) presented a fuzzy logic software toolset called FUZZYMANAGER that can effectively incorporate manager knowledge in a model either with or without historical data. The toolset consists of two modules: CLUESOME (CLUster Extraction for SOftware MEtrics) derives membership functions and rules, while FULSOME (FUzzy Logic for SOftware MEtrics) generates and refines the graphical output of membership functions and rule bases and then supports the prediction process via fuzzy inference. In two case studies, MacDonell & Gray (2003) demonstrated that in certain circumstances, the fuzzy logic approach not only outperforms linear regression

in representing software project management relationships, but also is capable of dealing with uncertainty and vagueness in classification and prediction. This is due to the fact that fuzzy logic methods create models based on the existing management expertise and allow adjustment when new knowledge is gained.

In a later paper, MacDonell (2005) described the empirical analysis of Kohonen self-organizing maps (SOMs) that utilise multiple attributes to create a model suitable for classification and prediction. As a neural-network based representation of data distributions, SOMs provide a two-dimensional visualization to expose the dispersion of artifacts/vectors and the interrelationships among factors. The author found that the SOM method was accurate and outperformed a corresponding regression model in classification and unbiased prediction in most runs of a software size prediction exercise. This suggests that SOM-based clustering may be a good candidate for modelling and prediction, as proposed in this research. Considering that the traditional SOM fails to deal with uncertainties, Li, Kuo & Tsai (2010) integrated the SOM with the fuzzy c-means (FCM) algorithm (Jain, Murty & Flynn 1999). FCM is a popular fuzzy clustering algorithm, which Jain et al. (1999) applied to their intelligent decision-support model for clustering, visualization, and linguistic information processing.

The studies described above reflect that traditional parametric models cannot handle complex data and uncertainty well. Furthermore, compared with other machine learning methods such as analogy and standard artificial neural networks, or statistical techniques such as regression, a fuzzy logic approach and associated techniques can deal better with imprecision, which is likely to be a factor in regard to project management data. Thus there is reason to assert that fuzzy logic modelling (fuzzy sets, fuzzy rules, and fuzzy inference) and associated techniques such as fuzzy clustering could be a more suitable approach in the domain of software project management estimation.

The objective of this research is to assess the effectiveness of the Self-Organising Maps (SOMs) algorithm and its enhanced version – the Fuzzy Self-

Organising Maps (FSOMs) algorithm – for clustering project management data. These clusters are then used to forecast the size (i.e. lines of code) of software artifacts or the effort required to produce them. In other words, the clusters can be used for software project estimation. Therefore this study addresses the following research question:

> *"Is the Self-Organising Map an appropriate candidate for estimation in software project management?"*

## 1.2 Research Design

In Information Systems research, presenting the accomplishment of an artifact or proposed framework with robust evidence from case studies can serve the purpose of demonstrating support or otherwise for a research hypothesis. Such an approach is embodied in the Design Science methodologies. For that reason, the Design Science Research Process (DSRP) model of Peffers, Tuunanen, Gengler, Rossi, Hui, Virtanen & Bragge (2006) has been utilised in this research.

Specifically, this research pursues the evaluation of the Self-Organizing Map (SOM) and Fuzzy SOM (FSOM) in software project management. We adopt as a benchmark Linear Regression, which is one of the most commonly used statistical prediction techniques. To ensure a fair and complete comparison, we also create for each case study a model – namely Fuzzy Linear Regression – as the Fuzzified version of the original Linear Regression to parallel FSOM to SOM.

Data sets from three software contexts are employed in our study in order to test our models: 1) the 4GL (i.e. Fourth-generation programming language) Systems data set, which was collected at the University of Otago in New Zealand; 2) the Desharnais data set, which is a publicly available data set for software engineering research; and 3) the Miyazaki data set, collected and published by the Fujitsu Large Systems Users Group. Prior to the construction of predictive models, correlation analysis is conducted in order to select

appropriate variables from the original data sets to avoid noise adversely affecting the estimation results. In order to assess the accuracy of models, a diversity of statistical approaches is utilized in the data analysis of prediction outcomes of SOM, FSOM, Linear Regression, and Fuzzy Linear Regression.

## 1.3 Thesis Structure

This thesis is structured as follows: the next chapter provides a literature review of previous research concerned with software project planning and general clustering techniques. Chapter 3 explains the research methodology using four examples, and highlights the application of the DSRP model in this research. In Chapter 4, a study of SOM considers its features and drawbacks, along with applications and extensions. Chapter 5 compares the approaches for constructing Fuzzy SOM and their applications. Chapter 6 reviews the benefits of Fuzzy C-Means as a clustering tool. Chapter 7 describes in detail the three data sets with variable selection. Information about tools for creating SOM and FSOM models is also provided in this chapter. Chapter 8 presents analyzed empirical evidence based on case studies that utilise the three data sets. Chapter 9 summarizes and synthesizes the case study results, Chapter 10 points out the limitations of this research, and offers recommendations for future research.

# 2 Literature Review

This chapter begins by reviewing contemporary project management estimation techniques, providing a clear understanding of the context for my study. It also provides justification for investigating a machine-learning approach and associated clustering techniques. This is followed by a survey of related clustering techniques, concluding with an evaluation of their suitability in a project management estimation context. This motivates the research question for this study, on the suitability of self-organizing maps.

## 2.1 Software Project Planning

According to the Project Management Institute (PMI) the activities involved in project management can be classified into five processes: initiating, planning, executing, monitoring and controlling, and closing a project. These processes can take place in a single project phase or occur cyclically throughout an entire project. In software development and maintenance projects, determining precise estimates of duration, cost and required effort at the beginning of the software life cycle is one important determinant of project success as such estimation has an impact on resource allocation and project feasibility (Corbel, 2008). Underestimated costs can lead to forced investment with minimal or even no profit, while overestimated costs could cause unnecessary project cancellation. Estimation should also not be a one-shot activity: both Pfleeger (2001) and Sommerville (2007A) state that when more accurate project information is obtained or when project aspects change, the estimation needs to be refined.

Generally, personnel effort is the biggest component of software project cost (Fleeter, 2001). It is determined by how many staff-days (some managers of larger projects would utilize months rather than days) will be necessary for carrying out the project. While it is essential to determine the required effort for completing a project, effort is the component with the highest degree of uncertainty among all the cost components.

Due to the nature of software development, most processes and activities have inter-relationships that imply that isolation is impossible. For instance, important factors that a manager seeks to control, such as time, cost and quality, are co-related and affected by various other factors in a complex manner. Therefore, managers need to pay attention to a large number of variables, and take into account their complicated interrelationships. Pfleeger (2001) identified several key factors that influence the estimate, such as system complexity, system size, project team capabilities and experience, the anticipation of changes in customer requirements, team size, available resources, and others.

There are many different techniques used to perform estimation for software development projects. The Project Management Body of Knowledge (PMBOK) categorises them into three classes: expert judgment, empirical data modelling, and machine-learning (ML).

### 2.1.1 Expert Judgment

One of the most commonly used effort-estimation methods is expert judgment. Naturally, prediction accuracy when this approach is used depends on the experience, competence, perception, and motivation of the estimator(s) (Fleeter, 2001). Experts in relation to the proposed software development application and perhaps the software domain may be consulted and their individual cost estimates are then compared and discussed until an agreed estimation is reached (Sommerville, 2007B). In some cases this may involve weighting the estimates according to individuals' expertise.

In a similar vein, analogy-based estimations are widely used as well. By analogy, the cost of the new project is estimated based on one or more finished projects. In addition, the method can be extended so that if system A and system B are similar, while the complexity or size of A is double that of B, then one can suppose A to cost double the cost of B. However, projects that appear

to be analogous could in fact be very different. Even if the differences between projects are identified, their influence on project cost may still be uncertain, as the relation between and project characteristics and cost is not always known. Furthermore, some factors other than those associated with the product being constructed may be influential. For instance, the larger the project team, the more time may be needed for communication and co-ordination.

Besides its inherent subjectivity and variability, expert judgment also strongly relies on current data. To reflect the current practices, the data for expert judgment must be updated regularly. Moreover, as pointed out by Pfleeger (2001), most expert judgment techniques are far too simple and can ignore factors that have an impact on the effort needed for a project.

Furthermore, MacDonell & Gray (2003) indicated that when experienced project managers leave an organisation, the knowledge they take with them may be crucial for project planning and could be difficult to replace. Especially in those organisations that are not mature in operation, such knowledge could even not be replaceable. Historical data can be utilized for model development, indexed for retrieval, and mined for trends and patterns; less mature organisations are categorized by the absence of such an historical database. Unfortunately, most modelling methods assume that such data exist.

### 2.1.2 Software Metric Models

A metric is defined by the IEEE Standard Glossary of Software Engineering Terms (Pressman, 2001, p.81) as “a quantitative measure of the degree to which a system, component, or process possesses a given attribute”.

In a software development project, when the criteria of measurement are established, it can be fairly straightforward to gather direct measures such as cost and effort applied, product aspects including Lines Of Code (LOC), and other attributes. Nevertheless, product characteristics such as quality,

efficiency, complexity, functionality, reliability, and maintainability, have to be indirectly measured, as they are difficult to assess.

Therefore, planners and project managers have been known to default to using LOC as it can be so easily counted. As a result a huge proportion of data predictions based on LOC exist in the literature, and LOC or KLOC (thousand lines of code) is one of the key inputs in many software estimation models. However, accurate estimates for LOC in advance of analysis and design are difficult to achieve. Furthermore, LOC measures cannot accommodate nonprocedural programming languages in an effortless manner and they tend to unfairly penalize shorter but well-designed programs.

The Function Point (FP) metric that was first proposed by Albrecht in 1979 is derived from empirical relationships grounded in direct measures. Planners and project managers estimate whether a particular entry is complex, average, or simple with conventions established by internationally standardised function point methods. Similar to LOC, function points are used to standardize measures for software quality, productivity, and other project aspects. It is worth noting, however, that the determination of complexity is subjective to some extent.

Originally, the function point measurement approach was designed for business information systems applications. Hence, it is inadequate for most engineering and embedded systems in contrast to information systems, which deliberately segregate function and control data dimensions. To remedy, a superset of the basic function point measure has been proposed. One of them is a feature point method that accommodates applications that have high algorithmic complexity. As a result, process control, embedded, and real-time software applications are amenable to quantification using the feature point approach.

### 2.1.3 Empirical Data Modelling

In most software estimation models, the relationship between size, effort and cost and the elements that influence them are presented as equations. Effort is normally set as a dependent variable while several elements such as size, experience, and application types are the independent variables. LOC or FP is calculated by empirically derived formulas and the resultant values are plugged into the estimation model. As Pressman (2001) pointed out, since most estimation models are based on empirical data derived from limited project samples, it is necessary to exercise caution in regard to the scope of applicability of the results. The majority of these models utilize project size as their key element. Such an emphasis obviously places extensive reliance on the accuracy of size measurement given its role as the primary variable. Since estimations are normally demanded before a system is expressed as LOC, the models simply 'shift' the challenge of effort estimation to one of size estimation.

The original Constructive Cost Model (COCOMO) was created in the 1970s. Boehm selected size as the principal determinant of cost and adjusted the initial estimate according to several cost drivers reflecting aspects of the project, the development environment, the product, and attributes of staff. Boehm then created COCOMO II, which incorporates three sizing techniques to reflect the evolution of software development (Boehm & Valerdi, 2008). Instead of using LOC as its key input, COCOMO II reflected the futility of obtaining an accurate value for LOC in the early stages of the development cycle. In COCOMO II, planners and project managers start by determining prototypes for high-risk aspects including software and system interaction, user interface, performance and so on. In the early design stage, designers have to state alternative architectures and concepts of operation. Development begins in the post-architecture stage when further details are unveiled, and many costly elements become more predictable. Most importantly, size can be more accurately estimated in terms of LOC or FP.

However, Boehm & Valerdi (2008) highlighted that COCOMO II does not cover some development styles, therefore additional COCOMO II related models were developed. Figure 2.1 below illustrates the COCOMO suite of models.

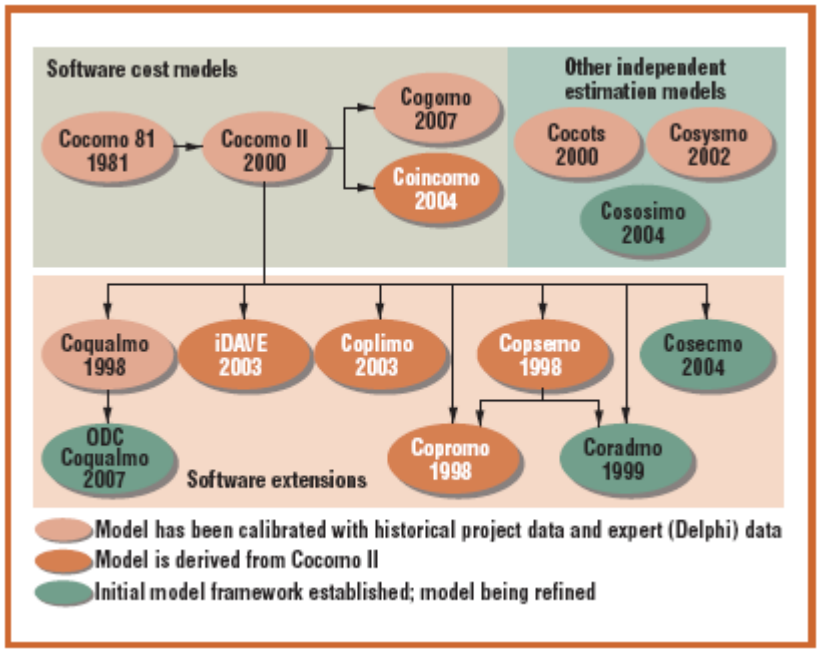


*Figure 2.1.* The COCOMO Suite of Models of Boehm & Valerdi (2008)

The software engineering field is continually being reinvented: consider structured methods, abstract data types, agile development processes, and emerging programming languages. Boehm & Valerdi (2008, p.80) suggested that modeling should prune the less relevant software engineering experiences while retaining the parts with durable value.

### 2.1.4 Machine Learning Models

In reviewing the literature it appears that traditional metrics and empirical (primarily statistical) data models cannot satisfy the needs of accurate software project planning in the ever-evolving software environment (e.g. see the review of Jørgensen & Shepperd, 2007). More and more researchers and practitioners have started to turn their attention to machine learning to leverage clustering and prediction algorithms that can be used to estimate aspects of their software projects (e.g. see Kocaguneli, Menzies & Keung, 2011).

Machine learning techniques are broadly used to automatically extract knowledge. Machine-learning approaches discussed in Muzaffar & Ahmed (2010) include rule induction, analogy, regression trees, evolutionary computation, Bayesian belief networks, artificial neural networks, and fuzzy logic. The setting of the parameters of the underlying techniques affect the classification or prediction accuracy of machine learning methods. In terms of model accuracy, MacDonell (2003) notes that the neural network is more effective for development effort estimation than case-based reasoning and rule induction methods.

However, Dick, Meeks, Last, Bunke & Kandel (2004) highlighted that when identifying minority classes in a skewed data set machine learning algorithms tend to be less effective without preliminary treatment of the data (for instance, oversampling the minority class(es)). In addition, small variations from the overriding linear behaviour that could be the most important features, would generally be considered noise. On the other hand, Moreno, Ramos, García, & Toro (2008) pointed out that in general only one output variable is pursued in the use of machine learning techniques. Additionally, from a non-mathematical specialist perspective, the complex and closed structures of machine learning methods can make them difficult to interpret (MacDonell, 2005; Berlin, Raz, Glezer & Zviran, 2009).

### 2.1.5 Software Quality

Apart from delivering the proposed software on time and on budget, ensuring its quality and reliability is also vital in software project development. Especially for systems where software failures could cause severe consequences, the demand for software quality prediction remains paramount.

One of the mechanisms to enhance software quality is to identify, locate and treat the causes of intolerable variations for software quality monitoring and control. The ongoing utility of quality measurements requires the collection of software metrics from different process phases of the software development.

Although different metrics determine different characteristics, many tend to be related not only to each other (through the common dimension of size) but also to the number of failures in a module. Therefore, models based on software metrics can identify the number of faults expected in potential error-prone modules, so that project managers pay more attention to high-risk modules when inspecting and prioritizing development effort and planning maintenance and reengineering activities.

Typically, software quality classification models are trained with software measurements and defect (software quality) data from prior development experiences with similar projects. Such an approach presumes the organisation has had previous experience with similar project(s) and that defect data exist for all modules as training data. In such a case, models are based on supervised learning since the software quality measurement guides the training process. In software engineering practice, however, the measurements may be inaccurate, incomplete, or even unavailable. These situations may occur when the organisation does not have experience of developing a similar system or relevant and accurate software quality measurements from prior system releases. Moreover, when the organisation has no previous development experience of a similar system, inappropriate usage of measurement and defect data can occur in modelling.

Since the supervised learning approach to software quality modelling is inapplicable due to the absence of defect labels and/or training data, labeling each program module as either fault-prone or not fault-prone relies on expert judgment, an approach that can become time-consuming, laborious and expensive. Particularly in the last decade several relevant studies have been carried out. As fault-prone modules that have similar measurements would be clustered together, unsupervised learning methods that can group modules by their software metrics value (while not needing knowledge of dependent variables as characterized by class labels) are more appropriate for model building.

Yuan, Khoshgoftaar, Allen & Ganesan (2000) presented a modelling technique that predicted the number of faults by fuzzy subtractive clustering, and then evaluated the model by module-order modelling. Instead of a quantitative approach with a crisp classification method, fuzzy logic started with the concept of a fuzzy set that had no clearly defined boundary and admitted a likelihood of partial membership. Membership functions of the fuzzy set mapped its appropriate value to the element of the domain. Generally, it is complicated to elicit fuzzy rules from software engineering experts. To offer an alternative, subtractive clustering produces fuzzy inference rules through clustering the training data automatically. Each fuzzy inference rule is represented as a cluster center. Moreover, a Gaussian membership function is then designed for each variable.

Later, Zhong, Khoshgoftaar & Seliya (2004) developed a clustering-based and expert-based software quality estimation method for an interactive software quality evaluation system that involved software engineering experts in the process. Two different clustering methods (k-means and Neural-Gas) were studied and the authors found that the k-means algorithm runs much faster. At the same time, these authors (Seliya, Khoshgoftaar & Zhong (2005)) proposed a semi-supervised clustering scheme for software quality estimation with incomplete fault-proneness defect data. When comparing Neural-Gas clustering with expert-based labeling, the former scheme yielded better classification results. Furthermore, Seliya & Khoshgoftaar (2007) introduced a constraint-based semi-supervised clustering scheme that utilized a k-means algorithm for clustering modules that were already labeled as either fault-prone or not fault-prone by a software engineering expert.

Dick, Meeks, Last, Bunke & Kandel (2004) were one of the proponents of fuzzy c-means clustering as it permitted ambiguity and noise that clearly reflected the reality of software failure analysis better. Likewise, Pedrycz & Succi (2005) established a user-friendly and straightforward two-phase hyper-box approach in which fuzzy c-means clustering from a collection of “seeds” of the hyper-boxes were used in the first phase, then genetic algorithm “grown” (expanded)

hyper-boxes were utilized in the second phase. Reported in Aroba, Cuadrado-Gallego, Sicilia, Ramos & Garcia-Barriocanal (2008), a fuzzy-clustering based segmented model exhibited better predictive capabilities, presented higher explicative capabilities, and were able to aggregate estimation from different components of partial models.

## 2.2 General Clustering

In discriminant analysis (a form of supervised classification), a collection of pre-classified training data are provided as labeled patterns so that the model can learn the descriptions of classes. However, in more exploratory pattern-analysis and machine-learning situations (such as pattern classification, data mining, document retrieval, and image segmentation), there are few statistical models available, and the decision-maker must avoid making assumptions about the data. Therefore, clustering (i.e. unsupervised classification) that explores the interrelationships among the data based on similarity is more appropriate. The given collection of unlabeled patterns (usually represented as points in a multidimensional space, or vector of measurements) is grouped into meaningful clusters. Patterns within the same cluster are more similar to each other than patterns from different clusters. Labels for categorizing clusters are solely obtained from data, in other words, they are data driven (Jain, Murty & Flynn, 1999).

### 2.2.1 Case-based Reasoning and Analogy-based Estimation

In case-based reasoning, a method that mimics the process of decision making by an expert, stored observations that are the closest to a new one would be used for new value estimation. This approach has found favour is some prior research in software project management. Gray & MacDonell (1997) found that a case-based reasoning system outperformed FP and COCOMO models, and was close to the level of an expert. For this reason, they suggested that expert reasoning by analogy be used as a management support tool. Berlin, Raz, Glezer & Zviran (2009) also mentioned that a case-based approach named

ESTOR was reported to achieve significantly better performance than COCOMO and function point analysis on restricted samples of problems estimation. However, case-based reasoning systems are not without their problems: they are intolerant to irrelevant features and noise, and are also strongly influenced by the similarity function used in terms of performance.

Although the approach seems to suit effort estimation well in principle (especially when the software product is poorly understood), analogy-based estimation still faces challenges such as the uncertainty of software attribute identification and measurement because of the involvement of human judgment. Another challenge is the variability of data set structures such as number of attributes, training data set size, missing values, nominal and ordinal scale attributes, outliers and collinearity (Azzeh, Neagu & Cowling, 2010). On the other hand, Huang, Chiu & Liu (2008) suggest applying suitable adjustment and weightings to improve the accuracy of analogy-based software effort estimation.

Recently, Azzeh, Neagu & Cowling (2010) integrated Fuzzy modelling in Grey Relational Analysis (GRA) to form a new model called Fuzzy GRA. Because of the employment of the concept of absolute point-to-point distance between cases, GRA is considered as a simple form of case-based reasoning which flexibly models complex nonlinear relationships between cost drivers and effort. Fuzzy GRA comprises four main stages: data preparation, feature identification, case retrieval, and effort prediction. To reduce the uncertainty and imprecision inherent in attribute measurement, fuzzy set theory is used to provide a representation scheme and mathematical operations with a formal quantitative model to capture and handle vagueness in natural language. In addition, a fuzzy model is employed for moderating uncertainty or similarity degree between reference tuple and treatment. There are several limitations of the Fuzzy GRA model such as the absence of a linear search to find the best value of the distinguishing coefficient for each data set, and the demand for sufficient numbers of observations for constructing the fuzzy sets.

### 2.2.2 Neural Networks

Over the past three decades, neural networks have been used extensively in many software metric modelling studies for both classification and clustering. Neural networks are capable of representing complex non-linear relationships and approximate functions. The neural networks commonly used in this domain are “feed-forward” networks trained using the back-propagation algorithm.

In back-propagation trained feed-forward neural networks the number of layers and neurons in each layer are first selected along with determination of how the neurons will be connected to each other, a transfer function, and parameters for the training algorithm. Then the network is trained by iteratively adjusting the weights between the input nodes and the output nodes to narrow down the gap between its predicted output and the actual output. To optimize the network’s ability for generalization (which is measured by its predictive performance on unseen data), this process needs to be stopped before the training data has been completely learned. Since the architecture of the networks affect their performance, and also to ensure good generalisability, a range of architectures are normally tried and assessed by using a validation data set.

Jain, Murty & Flynn (1999) summarized three important features of neural networks in pattern clustering: they 1) require quantitative features to represent patterns for processing numerical vectors, 2) incorporate parallel and distributed processing architectures, and 3) can operate as pattern normalizers and feature selectors with appropriate weights provided. Well-known examples of neural networks for clustering include k-means cluster analysis, vector quantization, and Self-Organizing Map (SOM).

In a comparison of linear regression estimation models and models derived from Artificial Neural Networks (ANN), Berlin, Raz, Glezer & Zviran (2009) found that the ANN models did not outperform regression in many aspects. They suggested, however, that SOM could be a potential candidate for

outstanding prediction performance.

### 2.2.3 Principal Components Analysis

In some cases it is possible to enhance model interpretation by focusing on just some of the attributes in the data set, and in doing so the computational workload of automatic pattern recognition or classification can be concentrated. Another benefit of adopting feature reduction is enhancing the clustering algorithm performance by eliminating noise from a data set. Feature reduction techniques include Principal Components Analysis, Nonlinear Component Analysis, Independent Component Analysis and others (Dick, Meeks, Last, Bunke & Kandel, 2004).

Principal Components Analysis (PCA) was first used in ecology, and it has become one of the most popular data set reduction methods in the past few decades. In Principal components analysis, points in the data set are treated as a feature space hyper ellipsoid with a few large axes and many small ones where the directions of the axes of the hyperactive ellipsoid and the length of these axes could be measured.

With respect of its variable extraction procedure, PCA is considered as a standard statistical technique. Low dimension artificial variables (i.e. principal components) are exploited as criterion variables or predictors in PCA to represent a high dimension data set. Therefore, the non-parametric method of PCA can be found in not only neuroscience fields but also computer graphics fields such as image compression and face recognition.

### 2.2.4 K-means Algorithm

The k-means algorithm is widely adopted in cluster analysis as it is easy to implement, and its time consumption depends on the number of patterns. After analysis by k-means clustering, a set of n projects would be partitioned into k classes. Software projects in the intra-cluster space are analogous while the

projects in the inter-cluster space are disparate. The steps of the k-means clustering process are: 1) choosing k cluster centers to concur with k random patterns or points; 2) computing the mean vector of all software projects in each cluster as the cluster center of gravity; 3) assigning each project to the closest cluster center; and 4) repeating steps 2 and 3 until convergence criterion is met. Typical convergence criteria could be minimal reassignment or minimal decrease in squared error.

There are variations of the k-means algorithm (Jain, Murty & Flynn, 1999). One variation allows the selection of a different criterion function altogether. Another variation facilitates the splitting and merging of the resulting clusters. Others attempt to provide an ideal initial partition, or to assist the algorithm to discover the global minimum value.

For the sake of downsizing the dimensions of each effort driver for clustering software projects, Huang, Chiu & Liu (2008) adopted the k-means method and Scheffe's method in their data-clustering model construction. The k-means method was utilized for effort drivers with ratio scales whereas Scheffe's method was employed for effort drivers with nominal scales. Based on these effort drivers, all historical software projects were clustered into separate groups and Huang et al. (2008) then built their respective effort estimation models.

### 2.2.5 Vector Quantization

To relieve the burden of heavy computation, instead of comparing every data item with all of the other ones, classical Vector Quantization (VQ) uses a much smaller set of models to represent the set of all data items. In VQ, vector-valued input data is clustered into a limited set of adjoining regions, and codebook vectors are used to represent each region as single model vectors. In the finest partitioning, the mean distance between each input data item and its respective closest codebook vector is minimised, hence the average quantization error is also minimized. In this sense, Kohonen (2008) interprets

the VQ learning principle as “Every input data item shall select and modify only the codebook vector that matches best with it, in such a direction that the degree of matching is increased”. In Somervuo & Kohonen (1999, p. 310), Learning VQ was used for the prototype sequences refinement to obtain optimal class separation.

However, Kohonen (2008) pointed out two weaknesses of VQ: the codebook vectors may fail to reflect any structures of the data; and the optimum state may only be local rather than global. On the other hand, although the k-means algorithm in VQ generally minimizes the root mean square quantization error, Kohonen, Nieminen & Honkela (2009) found that SOM could present a smaller quantization error than VQ.

### 2.2.6 Brief Comparison

In practice, and returning focus to the domain of interest in this study, there are four major project management issues that may impact the selection of a clustering technique: availability/lack of adequate historical data, presence/absence of an experienced expert, knowledge/uncertainty of the software project, and ease/difficulty of understanding the technique. Thus, the following table (Table 2.1) appraises the clustering approaches from the software project management perspective. The preferred responses to the criteria in the context of software project management are as follows: Require history data? No/Yes; Rely on expert? Yes/No; Handle uncertainty? Yes; Visualize results..? Yes. None of above approaches addresses these issues completely. Hence the self-organizing map that represents multidimensional data into a two-dimensional form is proposed as an alternative clustering technique here. The clusters then are to be used to create a fuzzy model of project estimation rules. In principle, this approach would meet the criteria as stated.

Table 2.1
*Comparison of Clustering Techniques*

| | Case-based Reasoning | Analogy-based Estimation | Neural Network | Principal Components Analysis | K-means Algorithm | Vector Quantization | Fuzzy Logic |
|---|---|---|---|---|---|---|---|
| **Require history data** | Yes | Yes | Yes | Yes | Yes | Yes | No |
| **Rely on expert** | No | Yes | No | No | No | No | No |
| **Handle uncertainty** | No | No | No | No | No | No | Yes |
| **Visualize results to help understanding** | Yes | Yes | No | No | Yes | No | No |

From the review of literature on the current state of project estimation research given in this chapter, it is clear that my research question is both relevant and novel. The next chapter describes and justifies the design of the research methods used to answer this question systematically and rigorously.

# 3 Research Methodology

## 3.1 Information Systems Research

The overall objective of Information Systems (IS) research is to acquire an understanding of how IS and the embedded Information Technology (IT) might be embraced by individuals, organisations and society with effectiveness and efficiency.

Throughout the systems development life cycle, then, practitioners conceptualize the problem to be solved in the feasibility study phase and represent it functionally in the requirements definition phase. Candidate solutions are considered in the systems design phase, followed by selection, construction and implementation of the most suitable solution. After evaluating the system with appropriate criteria in the testing phase, practitioners should acquire knowledge about how and why certain developed systems work or do not work (March & Smith, 1995).

Thus, IS research not only serves the purpose of understanding why things take place in particular circumstances to boost theoretical knowledge, but also benefits individuals, organisations and society activities ultimately. In short, IS research is an applied research discipline which is based on the development and use of theory to answer practical problems (Adams & Courtney, 2004).

Nunamaker, Chen and Purdin (1991) believed that some broad research domains like engineering and information systems need to adopt multiple methodologies to go through the concept-development-impact research life cycle, especially when an issue of the applications is assessed by its intrinsic value. IS research concerned with object-oriented databases, electrical engineering and computer science demonstrates such a life cycle.

Gregor (2006) identified five types of theory in IS research: *Theories for analysing* identify and specify characteristics of events, situations, and personnel and are based on observations. This class of theories is required in

the case when nothing or very little is known about the phenomena and relationships among them. *Theories for explaining* reveal "how things are or why they are as they are", along with alternative insights. *Theories for predicting* predict outcomes but leave part of the system a "black box". *Theories for explaining and predicting (EP Theory)* not only describe the theoretical constructs and underlying relationships to explain causes, but also provide prediction. In other words, EP theories identify what, when, how, why, and what will be. The last class is *theories for design and action* which is concerned with how to do something. Because the design theories identify the methods, principles of form and function, as well as justify theoretical knowledge, they can be found in constructive research, software engineering research, and in prototyping and systems development.

Among these five classes of theories, Gregor (2006) pointed out that analytic theory is the foundation of all of the other types of theory. EP theory can be derived from both theory for explaining and theory for predicting. As design theory is strongly interrelated with the EP theory, it can be informed by all four other types of theories. The work described here, which is centred on the systematic development and evaluation of an artefact, embodies theories for design and action.

## 3.2 Research Frameworks

Nunamaker, Chen & Purdin (1991) reviewed a variety of prior studies and presented a pattern of research relevant to software systems development. They noted that when observing a research domain one can find existing problems and form a hypothesis. The hypothesis can be confirmed and generalised into argument and evidence after analysis. Such a view can accommodate system development as providing "proof-by-demonstration" evidence to support or refute the hypothesis. They presented a multimethodological approach to information systems research that contains four research strategies: *theory building, systems development experimentation,* and *observation*, depicted in Figure 3.1. All of these phases

are essential; to complete the pursued research products, they need to communicate and interact with each other.

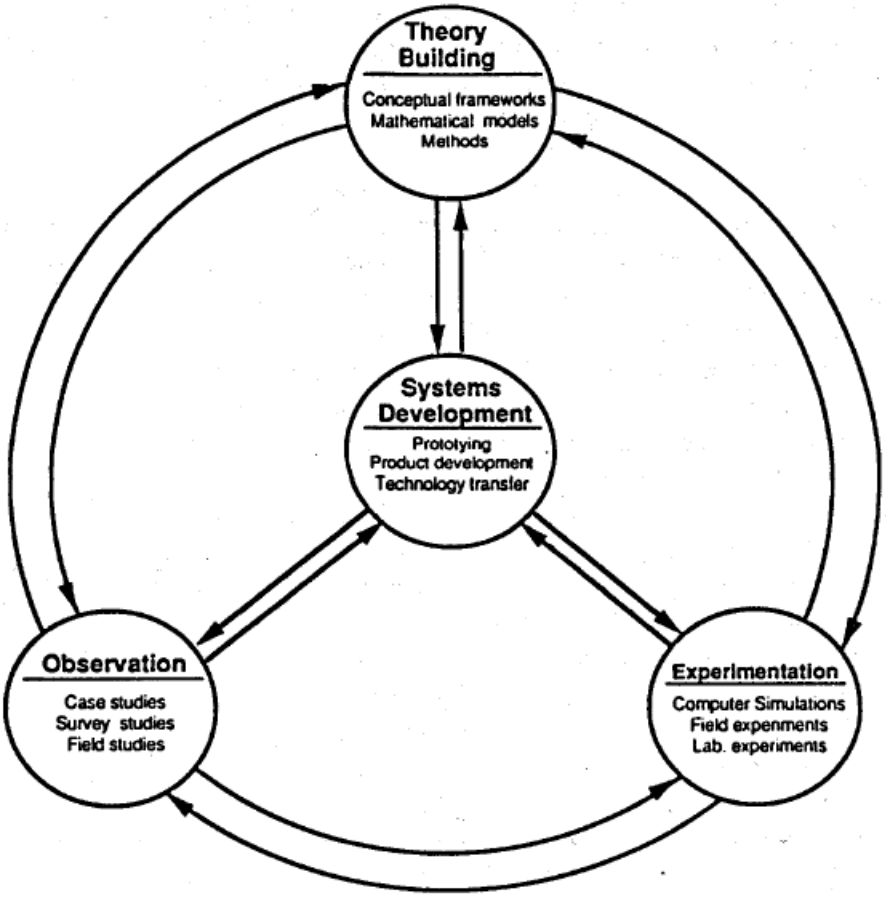


*Figure 3.1*. The Multimethodological Research Approach of Nunamaker et al. (1991)

In Figure 3.1, System Development is shown as the center of research that communicates with other methodologies, and each methodology complements and gives feedback to the others. The authors suggested using Theory Building to formulate hypotheses, design experiments and conduct observation. They also believe that results from Experimentation facilitated by System Development could refine theories and improve systems.

Nunamaker, Chen & Purdin (1991) then provided four examples to demonstrate that System Development could provide basic knowledge of a research domain, help the researcher identify a problem, and modify a current system or build new component(s) and/or a system. This, they claimed, demonstrated that system development is an important part of a multimethodological approach for IS research.

Finally, they proposed a process for system development research that consists of the following steps:

1. Construct a conceptual framework by formulating and justifying a research question that is significant;
2. Develop a system architecture to present guidance to build the system;
3. Analyse and design the system in order to provide a blueprint to implement the system;
4. Build the system to prove the design and the functionalities of the system development research project;
5. Experiment with, observe and evaluate the system.

Drawing on the work of Nunamaker and others, Peffers, Tuunanen, Gengler, Rossi, Hui, Virtanen & Bragge (2006, p. 84) extended the approach:

> We sought to design a design science research process (DSRP) model that would meet three objectives: it would be consistent with prior literature, it would provide a nominal process model for doing DS research, and it would provide a mental model for presenting and appreciating DS research in IS.

By assessing and comparing previous literature, the authors determined six activities in a nominal sequence as representing common process elements in design science research and illustrated them as shown in Figure 3.2. The activities are as follows:

1. *Problem identification and motivation*, which refers to the need to identify a research issue and justify the importance of a solution.
2. *Objectives of a solution*, which refers to the process of deriving the solution objectives from the research question.
3. *Design and development*, which refers to the creation of an artifact.
4. *Demonstration*, which refers to the means through which the artifact is able to be shown to address the issue effectively.
5. *Evaluation*, which refers to the process of observing and measuring whether or not the artifact provides a solution to the issue.

6. *Communication*, which refers to the need to communicate the issue, the artifact and its usefulness.

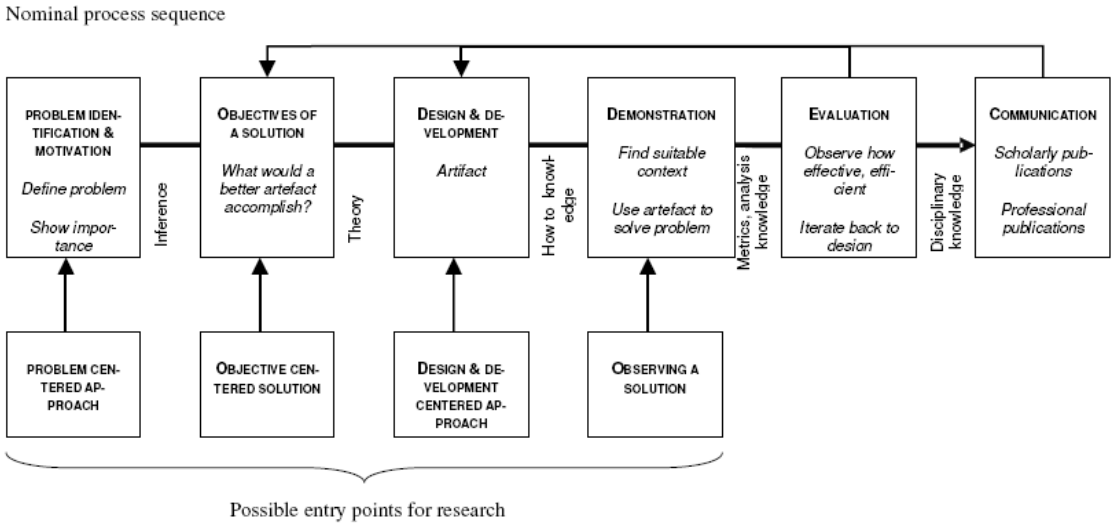


*Figure 3.2.* The Design Science Research Process (DSRP) of Peffers et al. (2006)

Using two case studies, the authors demonstrated that the DSRP model is not only consistent with concepts discussed in previous literature regarding design science in IS, but also provides both a nominal process to perform DS research and a mental model through which to present DS research outputs.

Peffers, Tuunanen, Gengler, Rossi, Hui, Virtanen and Bragge (2006) suggest that a researcher could initiate their work at any of the six steps (depending on the specific nature or the research) and move onwards. In the research reported in this thesis, the intent is to assess the utility of SOM as an instrument for size or effort estimation in software project management. According to Peffers et al. (2006, p. 92) this would be an instance where "the idea for the research resulted from observation of the problem or from suggested future research in a paper from a prior project." As SOMs represent a conceptually different way to provide support for software project management, this research naturally starts with Activity 1: problem identification and motivation, and will follow the steps set out above through to the evaluation of the approach (on existing data sets) and the production of research outputs (both a thesis and associated intellectual materials).

## 3.3 Related Examples from Diverse Domains

With the intention of studying the usage of SOM in ecological communities, Giraudel & Lek (2001) applied SOMs to a set of species abundance data, to examine the ordination of SOM against two linear ordination methods (Principal Components Analysis (PCA), and Correspondence Analysis (CoA)) as well as two nonlinear approaches: Non-metric Multidimensional Scaling (NMDS) and Polar Ordination (PO). Despite some drawbacks of the SOM algorithm, they found the visualization of sample units and species abundance provided by SOMs makes it a suitable exploratory technique that illustrates well the structures in ecological communities.

MacDonell & Gray (2003) presented a fuzzy logic software toolset called FUZZYMANAGER that can effectively incorporate data and knowledge in a single model either with or without historical data. The toolset consists of two modules: CLUESOME (CLUster Extraction for SOftware MEtrics) derives membership functions and rules, while FULSOME (FUzzy Logic for SOftware MEtrics) generates and refines the graphical output of membership functions and rule bases and then supports the prediction process via fuzzy inference. In two case studies of comparing the fuzzy model with a regression model with six measures of accuracy, MacDonell & Gray (2003) demonstrated that in certain circumstances, the fuzzy logic approach not only outperforms linear regression in representing software project management relationships, but is also capable of dealing with uncertainty and vagueness in classification and prediction. This is due to the fact that fuzzy logic methods create models based on the existing management expertise and allow adjustment when new knowledge is gained.

Previous studies of a single-staged Fuzzy Approximate Reasoning (FAR) technique found that it lacked effectiveness when working out complex decision-making problems, so Lee, Cho & Kim (2007) proposed a multi-staged fuzzy approximate reasoning to assure more robust results. The performance of their five step Self-Organizing FAR method (SOFAR) was evaluated against

test data obtained from Takagi and Hayashi (1991) and a real data set acquired from a civil engineering task. According to the rigorous statistical test of comparing actual values and approximations by SOFAR and the benchmarking method proposed by Takagi and Hayashi (1991), Lee, Cho & Kim (2007) illustrated that the proposed SOFAR had the potential of recognising comprehensive fuzzy approximate reasoning and providing accurate and high quality control paths.

Srinivas, Tripathi, Rao & Govindaraju (2008) introduced a two-level SOM-based clustering approach for Regional Flood Frequency Analysis (RFFA). The performance of their approach was measured against canonical correlation analysis and regression analysis. The proposed approach was found to perform better in estimating flood quantiles at ungauged sites. The authors also discovered that four out of five validity indices (including Fuzziness Performance Index (FPI), fuzzy partition coefficient ($V_{PC}$), fuzzy partition entropy ($V_{PE}$) and normalised classification entropy (NCE)) were not directly related to properties of the data, although they had been used previously in hydrology-related research. The relatively new extended Xie–Beni index $V_{XB,m}$, which takes into account the structure of the data and the fuzzy membership degrees, was considered as a convincing alternative for recognizing an optimal number of clusters.

In summary, Giraudel & Lek (2001) showed SOM can be applied to the ecological community ordination with competitive advantages to conventional statistical methods. MacDonell & Gray (2003) demonstrated a novel promising solution for software project management estimation by applying fuzzy logic, fuzzy rules and fuzzy inference in a system with the intent to produce better prediction results. Its contribution to IS design science research is the FUZZYMANAGER toolset itself. While Lee, Cho & Kim (2007) extended the existing FAR approach, Srinivas et al. (2008) also adjusted the knowledge of cluster validity measurement after rigorous statistical examinations.

The next section discusses the role of experimentation within a design science framework, as well as its relevance to the implementation of the research design in this thesis.

## 3.4 Experimentation

To form new methods, theoretical frameworks or models, it is essential that research is founded on rigorous analysis and on the identification of consistent system behaviours. Nunamaker, Chen & Purdin (1991) argued that theories could be exploited to extend hypotheses, construct the basis for the conduct of systematic observations, and guide the design of experiments. Conversely, experimentation validates and helps refine the underlying theories. Experimentation is also concerned with the selection of research strategies and issues of acceptance and technology transfer. Similarly, March and Smith (1995) believed an algorithm with best "worst-case" performance may not be a suitable algorithm for a particular goal; for this reason, metrics themselves ought to be scrutinised by experimental analysis and interpretation.

According to Adams and Courtney (2004), experimentation is related to action research, which is a two stage process. Hypotheses are formulated by collaborative analysis based on the nature of the research domain then experimentation is utilised to introduce collaborative change. Such an approach is especially relevant when the intent is to develop a new tool or system and them deploy that system with an organisational context, with the likelihood of change that typically follows such deployments.

Experimentation is considered as one of the typical methods used to evaluate designed artefacts as shown in Hevner, March, Park and Ram (2004). The experimental design evaluation methods here include: Controlled Experiment that studies the artifact in controlled environment for functional qualities, and simulation that execute the artifact with data.

The Design Science Research Process (DSRP) described in Peffers, Tuunanen, Gengler, Rossi, Hui, Virtanen and Bragge (2006), involves the employment of experimentation, simulation, case study, field studies and other suitable activities to demonstrate the efficacy of the artifact to carry out a task under certain circumstances. Similar to the DSRP of Peffers et al. (2006), which is more or less a modified version of the multimethodological research approach of Nunamaker et al. (1991), Gregor (2006) categorized experiments with case studies, field studies, surveys, and other methods as the approaches for investigating aspects of the type of theory for explaining and predicting.

Another modified version of the Nunamaker et al. (1991) framework is presented in Venable (2006). Experiments along with field studies, action research and simulation are the recommended techniques for Technology Evaluation, which interacts with Problem Diagnosis, Theory Building and Technology Invention/Design. In contrast to the Nunamaker et al. (1991) framework which is centred around System Development, the Venable (2006) approach places it central emphasis on Theory Building.

In this study, as shown in Table 3.1, we first defined the research question based on a literature review and analysis as the application of Activity 1 in the DSRP model. Secondly, we look at the accomplishments and aspects of SOM and Fuzzy SOM (Chapters 4-6) to determine an in-principle answer to our research question. Then we simulate real-world use of the artifact in the Design and Development phase of the DSRP model. As Activity 4, we demonstrate the effectiveness and accuracy of SOM and FSOM in regard to software estimation by creating prediction models for three different data sets (Chapters 7 & 8). Finally, we analyse the outcomes of our empirical analyses and consider limitations as well as future study areas (Chapter 9) in our Evaluation, Activity 5 of the DSRP model.

Table 3.1

*Application of DSRP Model Activity 1-5*

| DSRP | DSRP Activity Description | Chapter | Chapter |
|---|---|---|---|

| Activity No. | | No. | |
|---|---|---|---|
| **1** | Problem identification & motivation | 2 | Literature Review |
| **2** | Objectives of a solution | 4-6 | SOM, FSOM & Fuzzy C-Means |
| **3** | Design and development | 7 | Model Design |
| **4** | Demonstration | 8 | Model Evaluation and Comparison |
| **5** | Evaluation | 9 | Conclusion |

# 4 Self-organizing Maps

This chapter discusses the theory and implementation of Self-Organizing Maps, including some constraints in their application. This discussion is then extended to the implementation and application of Fuzzy Self-Organizing Maps.in chapter 5. These chapters form the basis for the justification of the approach investigated in this thesis using Fuzzy Self-Organizing Maps.

## 4.1 Self-organizing Map Clustering

The Self-Organizing Map (SOM) is a generic name for a group of algorithms concerned with the clustering and visualization of large data sets first introduced by Kohonen (1981). The SOM represents a nonlinear clustering method that projects the distribution of input items from their original multi-dimensional space onto a two-dimensional regular grid in an orderly manner. The mapping tends to preserve the density and the topological relations between input data points.

As Kohonen (1999) pointed out, when the primary data are not relatable metrically, a process of evolutionary learning can generate ordered SOM models and optimize them by their probabilistic variation. The input data set can be either metric vector space which derives the analytical algorithms for the optimal mapping or just the manifold in which the vectorial samples are positioned. Therefore, the SOM performs a form of Vector Quantization (VQ) where the model vectors (i.e., codebook vectors in VQ) can potentially be utilised for deriving nodes of a network that fits the manifold of the samples. Due to the combination of generalised median of a set and the batch computation, the SOM is not limited to metric vector spaces. As long as the similarity or distance measured between the factors can be defined, any set of items can be projected onto a SOM grid (Somervuo & Kohonen, 2000).

The projection is achieved by applying a matching process in contrast to traditional projection methods that represent each original sample separately, that is, the SOM identifies closest model vectors in some metric as a

generalised model for each input item. Such a collection of models is approximated for resembling all original input samples, and each will subsequently be associated with one of the grid units. In the optimised form, mutual similarity automatically determines the order of the model in the array, whereas the degree of difference increases with the distance of grid units in the array. In other words, more similar models are associated with adjacent nodes in the grid, less similar models are positioned farther away from each other.

## 4.2 The Basic SOM Algorithm

By extracting characteristic features or aspects of the data (i.e., finding the clusters of data), SOM can represent the topological relationships of high-dimensional data items. An input item is initially identified with the corresponding best-matching unit. The classification of the input is hence assigned to the best-matching model.

The SOM is composed by *n*-dimensional Euclidean vectors as the input layer $\{\mathbf{x}(t)\}$, and a two-dimensional topologic map (i.e., grid) with specified number of neurons (a.k.a, nodes) as the output layer $\{\mathbf{m}_i(t)\}$. Akin to most artificial neural networks, SOM operates in training and mapping modes. Throughout the training process, the output layer is produced and converged according to Equation (1):

$$\mathbf{m}_i(t+1) = \mathbf{m}_i(t) + h_{ci}(t)[\mathbf{x}(t) - \mathbf{m}_i(t)] \quad (1)$$

Here, *i* represent the spatial index of the grid node with which $\mathbf{m}_i(t)$ is associated, whereas integer *t* defines a step in the sequence. At the same time, the *neighbourhood function* $h_{ci}(t)$ defines the rates of the modifications at different nodes. Equation (2) is "the most applied choice for the neighborhood function" suggested in Kohonen (2008).

$$h_{ci}(t) = \alpha(t)\exp[-\text{sqdist}(c,i)/2\sigma^2(t)] \quad (2)$$

On one hand, $\alpha(t)$ and $\sigma(t)$ are monotonically decreasing function of *t*. on the other hand, *c* refers to the index of a particular neuron in the map.

$$c = \arg\min_i\{||\mathbf{x}(t) - \mathbf{m}_i(t)||\} \quad (3)$$

As the "winner", $\mathbf{m}_c(t)$ has the smallest Euclidean distance from $\{\mathbf{x}(t)\}$.

Equation (1) and (3) delineate recursive steps, when new data is added, Equation (3) determines the best-matching unit in the map. Then the models at this “winner” neuron and its *spatial neighbours* in the map would be modified base on Equation (1). With such a manner, the models are trained to match better with the input (Kohonen, 2008).

In other words, each neuron as well as its neighbouring neurons learn to converge data with similar characteristics. The weight adjustment would not stop unless the map reaches a relatively steady state. Due to the topology preserving property of the map resulting from the training mode, new input data in the mapping mode can be clustered into adjacent regions on the map by its adjacent patterns. At the same time, the spatial relationships of the new input information could provide prediction on missing value of a particular Euclidean vector.

For each mapped unit, the sum of squared distances or maximum distance to other sequences is used to determine the centremost member, which is the item with the smallest sum of generalised distances to other items belonging to the neighbouring nodes (Somervuo & Kohonen, 1999). As pointed out by He (2009), the neighbourhood function, which ensures that the training process does not get trapped into local minima, is one of the unique properties of the SOM algorithm. To minimize the chance of mis-convergence caused by trended data, input data is chosen from the training data pool at random. Not only the prototype vector of the winning neuron but also its close neighbours as calculated by the neighbourhood function are updated with each new input. Such a mechanism increases the total number of clustering iterations as the same data is being reiteratively picked up during the process.

When data items belong to a finite number of predetermined classes, different models can be built to represent these classes with corresponding symbolic labels. Before associating an input item, nodes can also be calibrated according to the classes. Based on the node, the unknown input item is then classified and the most similar model of it is used to construct the map.

Kohonen (2008) mentioned two ways for calibrating models. When the number of input items is very large, the primary issue is to study how to associate each input data item to the various models. Then the identified model would be labelled based on where the majority of matches occur. If the number of input data items is small, the *k* nearest neighbour method is adopted. For each model, k input data items are selected to perform majority voting and labelling. The integer *k* is selected from the range of half a dozen to a hundred with the principle that *k* must be much smaller than the number of input items.

## 4.3 Diverse SOMs

Since SOMs have been applied to diverse applications, numerous versions of SOMs have been constructed. Most of these variations accept metric vectorial data, and their models are also vectors of the same dimensionality. Generally, a recursive algorithm is utilised for optimizing the models (Kohonen, 1999).

Kohonen (2008) emphasised the need of extracting characteristic features from original data. Since natural variations in observations may be very broad, comparing objects directly may not support good identification. Unless the input item is described by statistical indicators, even structural elements like pixels or other pattern components are not appropriate to use as the input vector. On the other hand, by describing the input objects as a finite and rather small set of characteristic features, the dimensionality of the input data and the computing load can be radically reduced. As a result, the first step of constructing a SOM is to extract features for each item then use the vector derived from them as the input vector to the SOM. Generally, feature selection is based on heuristic rules. However, it is worth noting that sometimes mathematical functions or transforms of the input items, such as principal components, spectra, or other orthonormal basis vectors could be regarded as features.

Any generalised distance function derived from the input items can be used in the construction of SOM. There are two main versions of SOM. Both versions initialize the model vectors either as random vectors, or as linear initialization, which is a regular two-dimensional sequence of vectors that allows much faster

convergence. At the same time, the stepwise corrective algorithm version includes the learning rate as a parameter that does not exist in the batch version. With the combination of linear initialization and batch computation, the batch wise training algorithm version produces the most distinctive and stable SOMs.

It was noted in Somervuo & Kohonen (1999) that the averages in the batch training version can be assessed as generalised “medians” over batches of samples when the distance function is defined. In their comparisons of SOMs, Kohonen, Nieminen & Honkela (2009) found that if the set of input vectors in the batch training version was finite and the neighbourhood function was stable, the corrections will equal zero after a certain number of iterations. Such an exact termination of the learning process is very helpful for ensuring that at least a local optimum has been achieved precisely. Once an ordered SOM has been created, it can be used for either clustering the input items directly, or as a gateway or directory in the exploration of data items (Kohonen, 2008).

## 4.4 Drawbacks of SOM

It is evident that SOM is in principle a good candidate for clustering and visualizing multidimensional data sequences onto a direct, straightforward two-dimensional graphic map in a fast computation fashion. However, in the past decade researchers have reported some downsides of the SOM.

For instance, Flexer (2001) noticed that some empirical studies demonstrated that SOM performs equally to or worse than statistical approaches. SOM was further criticised for its lack of density model definition, the absence of objective error function optimization and for convergence not being guaranteed. When comparing to K-means clustering, Flexer found that SOM performed notably worse as the extra neighbourhood had a tendency to skew the obtained cluster centers. Even if the neighbourhood was set to zero at the end of training (which is suggested by theoretical as well as empirical results), SOM still performed

worse in Flexer's study in terms of mean squared error.

Despite the employment of a non-zero neighbourhood at the end of training, SOM also performed significantly worse in terms of topology preservation. In Flexer (2001), the so-called chain-link problem (which consists of two intertwined 3-dimensional rings) was used for comparing Sammon mapping to SOM. The output maps clearly indicated the rigidity and the discretization of SOM's output space. Although the author used more output units than the available input vectors, he observed that the Sammon output map maintained the ring-like structures, whereas the SOM output-space bore the high risk of losing the entire structural information. Nonetheless, Sammon mapping is not only a rather slow and involved technique, but is also a fixed mapping in terms of both input and output. It has to recompute the whole mapping whenever an unknown input point is encountered, a point that in this study would represent a significant disadvantage.

In Giraudel & Lek (2001), the SOM was compared to Polar Ordination (PO), Non-metric multidimensional scaling (NMDS), Principal components analysis (PCA), and effect Correspondence analysis (CoA). A few drawbacks of SOM were identified: 1) it cannot control the direction of the gradients; 2) the training process is computationally intensive and its duration depends on the learning parameters and the size of the map; 3) due to the repeatability of the method (i.e., as stated above, the same sample units could be randomly picked up more than once), the final maps might be different even with fixed learning parameters; and 4) the size and the shape of the map needs to be determined prior to its creation.

When a set of inputs is obtained from the same cluster or category, the learning could get caught into local minima. To overcome such an issue, SOM randomly picks data from all available inputs. However, He (2009) showed that the side effect of this mechanism is an increased number of iterations as it also learns sparse data sets that are little or even negatively used in identifying clusters.

## 4.5 Application and Extension

Kohonen (1999) introduced a new method based on the batch version of SOM. In this faster optimization method, a form of averaging replaced the probabilistic trials. As long as the fitness function between the data and the models is defined, the new method can interpret non-metric data distributions through descriptive models. Such a method can be considered analogous to a fast genetic algorithm that identifies input data under different circumstances by utilising a fitness function to indicate the survival value in different models.

Instead of a single feature vector, Somervuo & Kohonen (1999) treated an entire feature vector sequence as a model to associate with each SOM node. By dynamic time wrapping that captures both input sequence differences and spatial variances of the feature vectors, and Learning Vector Quantization that fine tunes the prototype sequences to optimize class separation, the resulting SOM models can be used for pattern recognition and synthesis.

Similarly, Somervuo & Kohonen (2000) presented an extension of SOM that did not convert data sequences into histogram vectors for clustering. As an alternative, it allowed the user to select similarity measures for the sequences. A collection of sequences that approximate the database contents was then automatically found by the theory of generalised median of symbol strings. This extension was applied for clustering and visualizing large protein sequence databases.

More recently, Kohonen (2008) introduced a new finding where by the least-squares fitting procedure, a linear mixture of a few best-matching models can represent input items more accurately. According to other recent literature, genetic algorithms can increase the convergence speed of conventional SOM. He (2009) thus proposed an efficient approach that uses genetic algorithms to refine training data before learning. The author emphasised that the purpose of the approach is to enable the input vector learning procedure to eliminate the

progression of irrelevant data and the outcome is able to represent the distribution of input data.

For the last few decades, the SOM method has been widely applied in various fields such as statistical analysis, biomedical analysis, finance analyses, industrial analyses, and scientific analysis. Applications are diverse: some researchers implemented SOM with other algorithms as a vehicle to retrieve multimedia from very large databases. Some utilised SOM in the development of criminological computer-aided tracking applications. Li, Kuo & Tsai (2010) presented a framework that integrates fuzzy logic with SOM for crime trend patterns detection and analysis. At the same time, Yang (2009) applied SOM to acquire and reveal the connection between semantic metadata and tags of the Web pages. They also reported that SOM noticeably outperformed the *k*-means algorithm.

The next chapter introduces the notion of combining ideas from fuzzy logic with the SOM to implement a fuzzy SOM. The potential benefits of this mix of techniques are discussed and related to the project management estimation domain. Implementation options are also explored and the motivation for using Fuzzy C-Means for clustering in this thesis is discussed. More detail on the application of Fuzzy C-Means is provided in Chapter 6.

# 5 Fuzzy SOM

Fuzzy Logic and Neural Networks are technologies that complement each other. Since Fuzzy Rule-Based Models make use of linguistic terms and if-then rules, they are relatively easy for human beings to comprehend. In contrast, Neural Networks come with efficient algorithms that can learn from data and feedback but are relatively more difficult to understand and interpret. Therefore, merging these two technologies could be potentially useful in cases where both learning from data *and* human understanding of models are needed.

One of the most popular approaches is to combine the Fuzzy C-Means (FCM) algorithms with unsupervised learning. Bezdek, Tsao & Pal (1992) proposed a Fuzzy Kohonen Clustering Network (FKCN) which automatically adjusts both the learning rate in the competitive layer and the size of update neighbourhood during learning. Their results indicated that, in contrast to FCM, labelling errors in the KFCN were reduced accompanied with improved convergence. This model uses a scheme that decreases fuzziness and the size of the self-organizing map (SOM) without applying the concept of an ordered map. Other indicative examples of fuzzy SOM use are now considered.

## 5.1 FSOM in Image Processing

Due to the properties of visualization which benefit signal transmission, SOM is used frequently in pattern recognition. In particular, the Fuzzy Self-Organizing Map (FSOM) is widely adopted in image processing. Sum & Chan (1994) described an algorithm that merged FCM and SOM for image quantization. It was shown that such an algorithm satisfied the necessary condition of convergence. In an application in data compression, the root mean square error induced by FSOM was found to be smaller than that of SOM. At the same time, Vuorimaa (1994 A) reported that the root mean square error for the validation data set in their FSOM was only faintly worse than the one for the training set. This implied that the FSOM had good generalization capability with high accuracy and fast convergence. Vuorimaa (1994 B) also noted that the accuracy of the simulation results obtained with the FSOM was superior to

those achievable with fuzzy c-means clustering and a standard SOM when evaluated using the well-known IRIS data set.

In their FSOM, Vuorimaa (1994 A&B) replaced the neurons of the SOM with fuzzy rules. The fuzzy sets defined the fire area in the input space for the fuzzy rules. As the firing strengths of the fuzzy rules operate as the weights, a weighted average combines the output singletons of the fuzzy rules together. In other words, the structure of the FSOM is analogous to fuzzy logic controllers. The FSOM has just one default rule which covers all of the input space initially. By adding rules during the learning procedure, the user can control the number of fuzzy rules and consequently adjust the accuracy of the FSOM.

The FSOM proposed in Vuorimaa (1994 A) is a three-step learning approach: 1) establish the centers of the fuzzy sets according to the learning laws of SOM; 2) initialise the fuzzy sets and the outputs of the fuzzy rules; and 3) using an algorithm similar to Learning Vector Quantization 2.1, tune the fuzzy logic controller rather than fuzzifying the learning laws, and also tune the fuzzy sets while not just finding the best fuzzy rules. Because of the use of fuzzy set theory, the neuro-fuzzy systems can represent the learned information in a manner understandable by humans.

Vuorimaa (1994 B) presented a Multiple Input, Multiple Output (MIMO) version of the FSOM for pattern recognition. Membership values in the FSOM not only provided the classification information, but also specified the validity of classification. When the functions were not weakly associated, the MIMO version of the FSOM could model several functions simultaneously. It was asserted that the MIMO version could be exploited as a general purpose function approximator.

In order to overcome the perceived drawbacks of FKCN encountered in an image segmentation application (e.g. long convergence time, randomly initialised network weights, a fixed structure), Wang & Qi (1999) suggested an Adaptive Fuzzy Clustering Network model (AFKCN). The AFKCN is able to

derive an appropriate network structure and rational initial weights by investigating the grey distribution characteristics of an image. By replacing the “fuzzy concentration” operator with the “fuzzy intensification” operator, the convergence process of the network was accelerated and the computation cost per iteration was reduced through sample space conversion.

Kuo, Chi & Teng (2001) proposed an FSOM neural network which incorporates fuzzy inputs, fuzzy weights and fuzzy set theory. Their experimental results demonstrated that the FSOM neural network could properly cluster the image parts based on their captured images. Its accuracy increased along with the size of the output array. Compared to FCM, the proposed approach could support a more precise decision but for slightly longer computational time, which was suitable for applications that favoured accuracy over speed. Another favourable aspect of the FSOM neural network was that no pre-specified cluster number was required.

## 5.2 Assorted FSOM Applications

A two-step method for automatic and adaptive rule extraction with FSOM was introduced by Nomura & Miyoshi (1995). A neural network called the "Fuzzy Inference Network (FIN)" was proposed for learning the trend of data. The learning result was represented as fuzzy rules for performing fuzzy inference with FSOM. The authors claimed their method was more effective in adaptive rule extraction than other methods with feed-forward neural networks like Radial Basis Function and Genetic Algorithm when the centers of input attribute vectors moved steadily while the distances between them remained constant.

Kurd & Kelly (2007) defined a ‘neuro-fuzzy’ model, which is based on the FSOM, called the Safety Critical Artificial Neural Network (SCANN). Their pattern classification case studies indicated the generalization performance of FSOM was radically better than Nearest Neighbour Networks and Learning Vector Quantization. The SCANN was found to perform well in different areas

such as fault diagnosis of a reactor, control of the inverted pendulum, system identification, and handling other non-linear problems.

Another application of FSOM is solving the symmetrical Traveling Salesperson Problem (TSP) by finding a good solution. (The problem is that the salesman needs to start from a given city, visiting n cities only once and back to the origin city using the shortest route.) Chaudhuri, De & Chatterjee (2008) employed the 2opt algorithm (which is an optimisation approach for the TSP) to enhance the solution generated by an FSOM. Fuzzy c-Means was deployed and the difficulty of selecting network parameters was resolved by FSOM. Learning speed and estimation accuracy were enhanced to a great extent by the adoption of single adjustment of the weights policy. According to the numerical simulation, the solution produced by FSOM provided a more satisfactory solution than both the Lin-Kernighan Algorithm and the Evolutionary Algorithm for TSP when the number of cities increased.

## 5.3 FSOM in Decision Support Systems

In the past decade, more and more practitioners have employed clustering analysis methods as important aids in their decision support systems, embracing fuzzy logic due to its capability of modelling vague qualitative knowledge and imprecise data in linguistic terms (e.g. low, normal, high, very high), supporting human type reasoning and conveying uncertainty. Fuzzy Sets Theory is thus considered an appropriate candidate for analysing non-quantifiable problems that rely on semantic judgments in real life. Simultaneously, neural network models, which have the advantages of learning in data-rich environments, are inherently nonlinear, have massive parallelism, robustness and are fault tolerant. Therefore, integrating fuzzy logic with neural networks provides certain advantages when handling uncertainty problems in recognition process and espousing learning function whilst constructing intelligent decision making systems.

A model for the analysis of credit card market segmentation that comprised three main modules was proposed by Chi, Kuo & Teng (2000). The first module utilised FSOM for projecting multi-dimensional fuzzy data onto two-dimensional topological network data. The second module employed FCM for capturing the membership of each possible cluster for all data on the two-dimensional network to provide credit card market information. The third module engaged the Back-Propagation Neural Network (BPN) for learning the relationship between the output pattern of each cluster and the two-dimensional membership functions. With such an approach, the speed of response could be improved as new information is acquired.

To assure robust approximate reasoning results, a multi-staged fuzzy approximate reasoning system named SOFAR (Self-Organizing FAR) was proposed by Lee, Cho & Kim (2007). The proposed SOFAR was able to produce apposite fuzzy rules via SOM for each input-output data pair, as well as consider errors from both learning data and test data through back-propagation driven parameter modifications. The SOFAR comprised five steps of multi-stage FAR mechanism: 1) preparation, 2) determination of fuzzy rule partitions, 3) membership learning for a fuzzy rule, 4) fuzzy rules learning, and 5) decision making.

A two-level SOM-based clustering approach for regional flood frequency analysis was presented in Srinivas, Tripathi, Rao & Govindaraju (2008). A two-dimensional map was produced by using SOM in the first level. Then FCM was used to cluster the output nodes for discovering regions for flood frequency analysis. Prior assumptions regarding cluster number, cluster centers, and fuzzy memberships are necessary for converging to local minimum of the objective function. In order to guarantee optimal partitioning, five fuzzy cluster validation measures (namely fuzzy partition coefficient, fuzzy partition entropy, fuzziness performance index, normalised classification entropy, and extended Xie–Beni index) were computed.

Based on new distance measurement and update rules, Chen & Chen (2008) built a batch SOM algorithm for numeric and categorical data (NCSOM). To handle categorical data, it assigned the input vectors to map units with relative membership degrees by applying fuzzy set theory in SOM training. Hence the algorithm could work well with imprecision, uncertainty, and noisy environments. Considering that SOM can approximate the possible density of data by visualising partitive clustering algorithms with k-means, variants were combined with the SOM algorithms as a hybrid clustering approach to improve computational efficiency, data visualization, and data summarization.

Recently, Li, Kuo & Tsai (2010) developed a framework for the detection and analysis of crime trend patterns from historical data. Such a decision support model was based on FSOM because of its inherent superior learning performance and the ability of handling vague linguistic data. An FCM model was exploited for enhancing the learning rate and weight updating strategy of SOM. The issues of representing fuzzy time series (derived from temporal crime activity data), selecting the best-matching unit, and updating weights when training with crisp data were addressed in this framework. As a result, the proposed FSOM model facilitates the manipulation of fuzzy numbers as inputs, fuzzy similarity measurement, and fuzzy weight updating.

## 5.4 Algorithms Selection

Table 5.1 compares the algorithms employed in the decision support systems discussed in the previous section. It is evident that the Chi, Kuo & Teng (2000) approach and the Li, Kuo & Tsai (2010) approach take into account semantic terms such as “unimportant”, “very unimportant”, “intermediate”, “good”, and the like. Such a feature is essential for an application that is to support software project estimation. On the other hand, the Li, Kuo & Tsai (2010) approach also handles historical data in contrast to the Chi, Kuo & Teng (2000) approach that solely relies on fuzzy questionnaires that are filled out by experts.

Table 5.1

*Comparison of DSS FSOM Algorithms*

| **Literature** | **Chi, Kuo & Teng (2000)** | **Lee, Cho & Kim (2007)** | **Srinivas, Tripathi, Rao & Govindaraju (2008)** | **Chen & Chen (2008)** | **Li, Kuo & Tsai (2010)** |
|---|---|---|---|---|---|
| **Model / Framework** | Market segmentation of credit card system | SOFAR | Regional flood frequency analysis system | FNCSOM (Adapted from Chen & Chen, 2008) | Crime prevention system |
| **Step 1 Preparation** | Collect human judgments by fuzzy questionnaire. Use semantic terms for scale interval. Then pre-process answers into fuzzy data set. | Divide historical data into learning data set and test data set. | --- | Initialize the reference vectors of map units. | Acquire data for investigation. |
| **Step 2 Process** | Fuzzify input vectors and the connection weight vectors. Use FSOM to cluster the customer market. | Use SOM to determine the number of fuzzy rule partitions. | Use SOM to form a two-dimensional map. | Input the samples one at a time. Calculate the membership degrees between input vector and reference vectors. | Use FCM to fuzzify standardised monthly crime volumes, and then convert to semantic term according to the best matching membership degree after defining the fuzzy sets. |
| **Step 3 Process** | Apply FCM to acquire a more precise and reasonable clustering analysis, and to process the membership level of the vague data belonged to | Calculate the neural network-driven membership function for each fuzzy rule that has learning data set. | Apply FCM to cluster the two-dimensional map. | Update the reference vectors on numeric, nominal, ordinal variables separately at the end of each epoch over the training process. Replace old reference | Train FSOM to cluster the fuzzify crime data. |

| | | | | | |
|---|---|---|---|---|---|
| | each cluster. | | | vectors with new ones. | |
| **Step 4 Process** | BPN accept the benefit-seeking variables and the customer's personal data as input variables to produce 2D membership functions as output variables. | Apply BPN for avoiding over learning phenomenon. | Identify optimal number of cluster by five cluster validation measures. | Repeat from Step 2 a few times until the solution can be regarded as steady. | Extract information from time series database. |
| **Step 5 Decision Making** | Use the relation between the input variables and output variable to train a BPN for making decision in marketing promotion. | Calculate the final approximated value for input data. | --- | --- | Analyse crime pattern. |

Table 5.2

*Comparison of FSOM Algorithms*

| **Literature** | **Chi, Kuo & Teng (2000)** | **Lee, Cho & Kim (2007)** | **Srinivas, Tripathi, Rao & Govindaraju (2008)** | **Chen & Chen (2008)** | **Li, Kuo & Tsai (2010)** |
|---|---|---|---|---|---|
| **Model / Framework** | Market segmentation of credit card system | SOFAR | Regional flood frequency analysis system | FNCSOM | Crime prevention system |
| **Create fuzzy rules from data** | No | Yes | Yes | Yes | Yes |
| **Create fuzzy rules from expert knowledge** | Yes | No | No | No | Yes |
| **Create fuzzy sets (e.g. FCM)** | Yes | Yes | Yes | Yes | Yes |
| **Cluster the data (e.g. SOM)** | Yes | Yes | Yes | Yes | Yes |
| **Calculate membership degrees** | Yes | Yes | Yes | Yes | Yes |
| **Extract rule from temporal data** | No | No | No | No | Yes |

We evaluate these five methods as shown in Table 5.2 based on software project estimation scenarios. It is apparent that all five frameworks manage to handle uncertainty and visualize clustering results both of which would aid the software project manager in understanding a model and its meaning in context. However, the SOFAR mechanism in Lee, Cho & Kim (2007) is a fusion of fuzzy logic, SOM and neural network that entails the demand for a large data set, while in the case of software project management, data sets are relatively small. Similarly, the market segmentation model presented in Chi, Kuo & Teng (2000) involves the back-propagation neural network module that contrasts with our requirement. At the same time, the FNCSOM framework from Chen & Chen (2008) employs k-means variant and Learning Vector Quantization while we would prefer to avoid crisp projection.

Considering that a SOM usually generates more units than real clusters, some researchers (Chi, Kuo & Teng (2000), Srinivas, Tripathi, Rao & Govindaraju

(2008) and Li, Kuo & Tsai, 2010) cluster the SOM output by fuzzy c-means algorithm to obtain better insight into the natural structures. As both the regional flood frequency analysis approach in Srinivas, Tripathi, Rao & Govindaraju (2008) and the crime prevention system introduced in Li, Kuo & Tsai (2010) provide high-quality examples of fuzzifying SOM, we have good reason to believe that adaptively adopting algorithms from these two approaches would benefit our software project estimation application.

# 6 Fuzzy C-Means

## 6.1 Clustering

Data clustering analysis is one of the most useful techniques for discovering relevant patterns, groups, and relationships and associations within a large volume of data. It plays an important part not only in pattern recognition, image processing and communication, but also in system modelling, data mining and other decision-making application areas. Generally, cluster analysis is a variety of techniques that segment a set of data into several nonempty subsets (a.k.a. clusters). Each cluster has its weighted average as the center of gravity.

In the iterative clustering process, only cluster centers are moved (i.e. none of the data points are moved) in each step of partitioning the space while finding the better and better centers. The subdividing of the original data set is based on similarity metrics or probability density models; thus after clustering the mathematical similarity of intra-cluster observations is maximised and between data items for inter-cluster is minimised. One of the most commonly employed distance functions is Euclidian distance which measures mathematical similarity by computing the squared difference. When new data becomes available, the distance between the new data point and every cluster center will be calculated before adding the new data point to the cluster with minimum distance to its centre (Raju, Thomas, Kumar & Thinley, 2008).

## 6.2 Crisp Clustering

Clustering can be categorised into two general process types: Crisp clustering and Fuzzy clustering. In crisp clustering, each data point in the data set is assigned to one and only one cluster explicitly. Hence the boundaries of clusters are hard, crisp and have no overlaps (Bezdek, Ehrlich & Full (1984); Kannan, Devi, Ramathilagam & Sathya, 2010).

The diverse variations of *k*-means clustering algorithms are the most well-known and commonly used unsupervised partitioning techniques that are able

to classify crisp and highly structured data without prior information on the data distribution available. The letter ‘k’ stands for the initially provided parameter that indicates the number of clusters in the outcome of partitioning.

However, real world data is often characterised by vagueness and uncertainty and conventional crisp clustering algorithms are inappropriate for handling such challenges. Fuzzy clustering is a robust and flexible approach to dealing with natural data sets that consist of non-strict objects and have poorly defined boundaries that could result in overlapping cluster perimeters. Furthermore, Bezdek, Ehrlich & Full (1984) pointed that since the conventional approach fails to provide a mechanism to absorb deviant or indistinctive data, outliers are treated as noise and fall into the "unclassifiable" category. Partial membership to a fuzzy set can resolve this issue.

## 6.3 Fuzzy Clustering

Raju, Thomas, Kumar & Thinley (2008, p. 882) identify the following six characteristics of natural data:

*1) Not clearly known: Questionable; problematic*

*2) Vague: Not definite of determined*

*3) Doubtful: Not having certain information*

*4) Ambiguous: Many interpretations*

*5) Not steady: Varying*

*6) Liable to change: Not dependable or reliable*

In other words, sharp and precise distinctions are difficult to make and the choice between options is left uncertain. Therefore, by allowing partial membership, the fuzzy sets theory became the ideal candidate for handling uncertainty and modelling imprecise and qualitative information.

The concept of fuzzy set theory was introduced in Zadeh (1965). Membership of an object into a cluster is Boolean in crisp clustering, which means that it either belongs or does not belong to the cluster absolutely. Zadeh’s concept

utilizes the membership function to calculate the distance between object and cluster centers to interpret the memberships.

In contrast to the Boolean value of membership in crisp clustering, each data point in fuzzy sets has an associated degree of membership from 0 to 1 in every cluster. Such non-unique partitioning is fundamental in fuzzy clustering. The higher the value of the membership, the more similarity there is between the data point and that cluster.

### 6.3.1 Fuzzy C-Means

Fuzzy C-Means (FCM) clustering was proposed by Dunn (1973) and subsequently generalised and improved in Bezdek, Ehrlich & Full (1984). Before the name fuzzy c-means was introduced it was even known as fuzzy k-means. This shows that FCM is comparable to k-means clustering in many ways.

Fuzzy c-means clustering involves the computation of cluster centers and measuring the Euclidian distance between an object and the cluster centers. The calculation is repeated until the cluster centers are stable. In contrast to the traditional crisp clustering algorithms, FCM allows each data point to belong to more than one cluster by incorporating partial membership concepts of fuzzy set theory. The membership degree ranges between 0 and 1, and the sum of the memberships for each data point is equal (Bezdek, Ehrlich & Full, 1984).

FCM supports partial membership by forming overlapping clusters using a fuzzification parameter that establishes the degree of fuzziness of the clusters. The higher the parameter value, the more there is overlapping of clusters. When the parameter equals to 1, FCM acts as a crisp clustering algorithm. As a result, the embedment of fuzzy set theory enriches the traditional crisp clustering approach (Bezdek, Ehrlich & Full (1984); Raju, Thomas, Kumar &

Thinley, 2008). Nowadays, the most widely used fuzzy clustering techniques in practice are FCM and its derivatives.

### 6.3.2 Limitations of FCM

Although it has been demonstrated that FCM outperforms crisp and probabilistic clustering algorithms in terms of handling vague and uncertain natural data, it does suffer from some limitations.

As an unsupervised clustering algorithm, the clustering results of FCM need to be validated. Cluster validation examines how well the structure of the data set is reflected in the clustering results. The vital indicator of the structure is the number of clusters which is a user-initialised parameter that may be difficult to determine, especially pre-clustering, in real world practice.

Past literature has proposed several validity indices such as the partition coefficient and classification entropy for FCM. More recent work contends that a validity index ought to consider the density within individual clusters as well as the separation between clusters. Nonetheless, most existing validity indices are inefficient for spotting the number of clusters when the boundaries of different clusters are overlapping. In Sun, Wang & Jiang (2004), a new algorithm is proposed that is able to automatically define the number of clusters with a validating index for overlapping data.

Other stated drawbacks of FCM include: it fails to eliminate noise and outliers; it is poor in handling general crisp data sets due to the Euclidean distance emphasis on appraising dissimilarity; and it has high computational cost for large data sets due to the squared-norm for assessing similarity among data points and cluster centers. Kannan, Devi, Ramathilagam & Sathya (2010) introduce a Bray Curtis distance to reduce the negative impact of Euclidean distance on crisp data set handling and some fuzzy objective functions to reduce running time. Additional terms including a penalty term are also introduced to reduce the effect of noise and outliers in large data sets.

### 6.3.3 Application of FCM

In the interval partitioning process of the Li, Cheng & Lin (2008) forecasting model, the authors deployed FCM clustering for generating unequally-sized intervals as it considers the density of data points, and for taking into account historical data. Similarly to software project estimation, the researchers attempted to use techniques including statistics and artificial neural networks to tackle the forecasting problem based on time series data segmented by fixed time intervals. Likewise, traditional approaches for creating time series forecasting models rely extensively on historical data, which can be sometimes imprecise, ambiguous and even incomplete.

Instead of numeric values for traditional time series, fuzzy time series is represented as linguistic values under fuzzy logic theory and so it is capable of handling incomplete and vague data to take account of the uncertainty of real-world data that hinders the accuracy of forecasting models.

Based on the nature of SOM that was discussed previously and the examples of fuzzifying SOM in the last section, we apply FCM clustering to fuzzify our original data sets before creating SOMs, resulting in the use of FSOMs in the prediction of software project management attributes.

The next chapter describes the implementation of FSOMs using an FCM clustering approach in a series of the experiments undertaken to generate models on three different datasets from the project management domain. The results of these experiments are then presented in chapter 8 and analysed in chapter 9.

# 7 Model Design

## 7.1 Correlation Analysis

In software project estimation, a large number of variables that characterise the system and its development may be available. In order to facilitate the creation of a stable and accurate prediction model, it is important to distinguish between independent and associated or correlated variables. Introducing inappropriately selected variables to the model not only complicates the process of prediction, but also could lead to misestimates being produced.

Pairwise correlation analysis is one of the most commonly used statistical approaches. It is used for the interpretation of strength of association of two variables in case-control studies (O'Gorman & Woolson, 1995) in a wide variety of domains including psychiatric data analysis (Arndt, Turvey & Andreasen, 1999), and health psychology and epidemiology (Kraemer, 2006). In most cases the correlation coefficient is a unit-free measure ranging from −1 to +1. When the correlation coefficient equals one, the two variables are strongly positively associated. The closer the coefficient to zero, the less the two variables are correlated. When the correlation coefficient equals to zero, one can say that the variables are not related (at least in terms of the measure being used).

Among the often used statistical measures of association, Pearson's product moment correlation is commonly employed. However, the Pearson's r ($r_p$) is only suitable for indicating linear relationships, and it can be gravely affected by even just one outlier. Moreover, in previous studies such as that reported by Croux & Dehon (2010), the classical Pearson correlation was reported to be lacking robustness as its influence function is unbounded.

The potential presence of outliers and non-constant variance in software project datasets requires analysis and inferential techniques that can provide stabilised statistics with limited knowledge of the data distribution. Therefore, Spearman's rank correlation coefficient and Kendall's rank correlation

coefficient are the nonparametric procedures that researchers in this domain commonly consider. For instance, Grzegorzewski (2009) generalised the classical Kendall's rank-based nonparametric procedures to handle a fuzzy set in their mathematical model for preference systems with missing information or non-comparable outputs to measure association.

Other researchers have compared the performance of Pearson's $r_{p,}$ Spearman's $r_{s,}$ Kendall's $tau_{b,}$ and other correlations methods. In O'Gorman & Woolson (1995), Kendall's $tau_b$ was reported superior to the other methods with both correlated normally distributed variables and with log-normal variables, while it was just about equal to the other methods in the case of uncorrelated variables. Based on the result of their simulations (comprising fewer than 400 cases and controls), O'Gorman & Woolson report that Kendall's $tau_b$ is a suitable exploratory procedure for variable selection in the early stage of a case-control study with a small to moderate sized data set, unless the candidate variables are independent or follow Bernoulli or normal distributions.

Later, Arndt, Turvey & Andreasen (1999) demonstrated that Pearson's $r_p$ is unstable and performs poorly when outliers and non-constant variance are present. Kendall's correlation and Spearman's $r_s$ exhibit adequate protection against type I errors and are more consistent in these circumstances. However, instead of reflecting the population value, Spearman's $r_s$ is an inherently sample-biased statistic which tends to underestimate the true correlation, and the degree of divergence from the true value increases as the sample size decreases. Such a bias reduces its power during statistical testing. In contrast, Kendall's $tau_b$ is favoured in terms of expressing the strength of associations, especially for small to moderate sample studies as it is unbiased. Arndt et al (1999) also noted that its use led to more stable and therefore more replicable results. Its tendency to produce a narrow confidence interval and to lend itself to straightforward interpretation also make the Kendall's $tau_b$ superior from the statistical perspective.

Croux & Dehon (2010) examine the robustness of Kendall's $tau_b$, Spearman's $r_s$ and Quadrant correlation by a covariance matrix estimator in a simulation experiment. The results indicate that the influence functions of the Spearman's and Kendall's correlations are bounded and smooth, which confirms the general belief that nonparametric correlation methods are robust to outliers. In addition, $tau_b$ and $r_s$ have high statistical efficiency and acceptable levels of gross-error sensitivity, but the Kendall's $tau_b$ is preferable as it outperforms Spearman's $r_s$ from both perspectives.

As pointed out previously, the size of data sets in software project estimation is generally small to moderate. Our purpose of using a correlation coefficient is for the selection of appropriate variable(s) to create models for estimation. Since the Kendall's $tau_b$ is a simple yet efficient correlation and is favoured for revealing dependence of variables in ambiguous data sets it appears to be a good choice for this study. Also, with the advantage of small gross-error sensitivity and lower type I error, we consider Kendall's $tau_b$ as the most appropriate statistical instrument for this activity.

## 7.2 Data Sets

### 7.2.1 The 4GL Systems Data Set

The 4GL systems data set was collected over a period of five years. It contains 70 observations of small- to medium-sized 4GL systems related to transaction processing, data retrieval and reporting, and file maintenance activities. These systems were built by groups of senior students at the University of Otago in New Zealand to meet the real requirements of external clients that are usually small businesses or departments of larger organisations.

The 4GL data set includes variables that reflect the size of the data model, the functional decomposition chart, and the number of source statements. It was used for the demonstration of the viability of fuzzy logic modelling in software project management in MacDonell and Gray (2003), and for the assessment of

the performance of standard SOMs in MacDonell (2005). Table 7.1 illustrates the variables of the 4GL data set.

Table 7.1

*Variables of 4GL Data Set (Adapted from MacDonell, 2005)*

| **Mnemonic** | **Variable** | **Explanation** |
|---|---|---|
| ENT | Entities | Count of entities depicted in the entity-relationship diagram (ERD) |
| RSHIP | Relationships | Count of relationships depicted in the ERD |
| ATTRIB | Attributes | Count of attributes associated with the ERD |
| MENU | Menus | Count of menu screens depicted in the Functional Decomposition Chart (FDC) |
| EDIT | Entry/Edit | Count of data entry/edit screens depicted in the FDC |
| REPORT | Reports | Count of reports depicted in the FDC |
| NONMENU | Non-menu functions | Count of non-menu functions depicted in the FDC |
| FDCSIZE | FDC Size | Count of all functions depicted in the FDC |
| SIZE | System Size | Count of all non-comment source statements in the implemented system |

By calculating the Kendall's $tau_b$ correlation coefficient (Table 7.2), we found that ATTRIB, EDIT, NONMENU and FDCSIZE are significantly associated with system SIZE, which is one of the parameters that software project managers are keen to estimate. We select ATTRIB and NONMENU to construct our prediction models as ATTRIB reflects the feature of the database of the system while NONMENU is indicative of the functional capabilities of the application. In addition, EDIT shows strong correlations with other independent variables, likewise FDCSIZE is highly correlated to NONMENU. Therefore, adding EDIT and FDCSIZE into the creation of models is not likely to increase the accuracy of the prediction but would increase computational cost.

Table 7.2

*Kendall's taub Correlation Coefficient of Variables of 4GL Data Set*

| | ENT | RSHIP | ATTRIB | MENU | EDIT | REPORT | NON MENU | FDC SIZE | Size |
|---|---|---|---|---|---|---|---|---|---|
| **ENT** | 1 | .857** | .525** | .296** | .547** | .145* | .425** | .428** | .359** |
| **RSHIP** | .857** | 1 | .499** | .291** | .496** | 0.127 | .379** | .394** | .335** |
| **ATTRIB** | .525** | .499** | 1 | .258** | .501** | .263** | .476** | .465** | .483** |
| **MENU** | .296** | .291** | .258** | 1 | .375** | .239** | .395** | .542** | .287** |
| **EDIT** | .547** | .496** | .501** | .375** | 1 | .259** | .704** | .680** | .508** |
| **REPORT** | .145* | 0.127 | .263** | .239** | .259** | 1 | .608** | .579** | .379** |
| **NONMENU** | .425** | .379** | .476** | .395** | .704** | .608** | 1 | .897** | .563** |
| **FDCSIZE** | .428** | .394** | .465** | .542** | .680** | .579** | .897** | 1 | .558** |
| **Size** | .359** | .335** | .483** | .287** | .508** | .379** | .563** | .558** | 1 |

### 7.2.2 The Desharnais Data Set

The Desharnais data set was collected in a Canadian software house in the late 1980s by Jean-Marc Desharnais. It comprises data from 81 projects developed using three different programming languages. By respecting (non-)linearity and heteroscedasticity, this data set is considered as representative of data sets of software projects. Table 7.3 presents the properties of the Desharnais data set. As a well-known publicly available data set, the Desharnais data set has been used in many project management studies. With the purpose of evaluating the potential of genetic programming and two other machine-learning approaches for building effort prediction models, Burgess & Lefley (2001) used the Desharnais data set to examine the accuracy and ease of use of the three techniques. In their investigation of a machine learning technique namely C4.5, which yields tolerance missing values, Song, Shepperd, Chen & Liu (2008) assessed the Desharnais data set using the Mann–Whitney test to inspect the accuracy of their cost prediction models. Another instance of use of the Desharnais data set is reported in Keung, Kitchenham & Jeffery (2008). As an alternative to data-intensive methods such as linear regression, analogy-based software cost estimation (a.k.a. Case-Based Reasoning) is popular. The Keung, Kitchenham & Jeffery method employs Mantel's correlation randomization test to produce a method they refer

to as Analogy-X. In their study the Desharnais data set was used to demonstrate the advantages of using Analogy-X.

Table 7.3

*Variables of Desharnais Data Set*

| **Variable** | **Description** | **Data Type** |
|---|---|---|
| ActualEffort | Actual Effort measured in person-hours. A dependent variable. | Discrete |
| Duration | Actual project schedule in months. A dependent variable. | Discrete |
| ExpEquip | Team Experience measured in years. | Ordinal |
| ExpProjMan | Manager Experience measured in years. | Ordinal |
| Transactions | Count of basic logical transactions in the system. | Discrete |
| RawFPs | PointsNonAdjust that equals to Transactions + Entities | Continuous |
| Adj Factor | Function point complexity adjustment factor. | Continuous |
| Adj FPs | Function points adjusted by the Adjustment factor. | Continuous |
| Dev Env | Programming language. | Categorical |
| Year Fin | Year project ended. | Categorical |
| Entities | The number of entities in the systems data model. | Discrete |

Table 7.4 presents the correlation coefficients among the Desharnais variables. It is clear that some of the independent variables have strong associations with ActualEffort, and as a project outcome we use it as the dependent variable in our model. Specifically, RawFPs, Adj FPs and Entities are strongly correlated to ActualEffort. Although RawFPs and Adj FPs show higher correlation coefficients, the count of Entities is easier to understand and more convenient to obtain. Taking into account the significant interrelationship amongst RawFPs, Adj FPs and Entities, we exploit Entities as the sole independent variable in our creation of a prediction model.

Table 7.4

*Kendall's tau$_b$ Correlation Coefficient of Variables of Desharnais Data Set*

| | ActualEffort | Duration | ExpEquip | ExpProjMan | Transactions | RawFPs | AdjFactor | AdjFPs | DevEnv | Year Fin | Entities |
|---|---|---|---|---|---|---|---|---|---|---|---|
| **ActualEffort** | 1.000 | .427** | .177* | .073 | .350** | .518** | .363** | .536** | -.279** | -.030 | .470** |
| **Duration** | .427** | 1.000 | .248** | .189* | .284** | .426** | .163* | .419** | .016 | -.056 | .376** |
| **ExpEquip** | .177* | .248** | 1.000 | .334** | .053 | .185* | .234** | .213** | -.125 | -.135 | .213** |
| **ExpProjMan** | .073 | .189* | .334** | 1.000 | .089 | .147* | -.048 | .131 | .230** | .051 | .143* |
| **Transactions** | .350** | .284** | .053 | .089 | 1.000 | .629** | .306** | .616** | .072 | .048 | .215** |
| **RawFPs** | .518** | .426** | .185* | .147* | .629** | 1.000 | .330** | .911** | .014 | .077 | .589** |
| **Adj Factor** | .363** | .163* | .234** | -.048 | .306** | .330** | 1.000 | .422** | -.178* | -.046 | .223** |
| **Adj FPs** | .536** | .419** | .213** | .131 | .616** | .911** | .422** | 1.000 | -.003 | .077 | .577** |
| **Dev Env** | -.279** | .016 | -.125 | .230** | .072 | .014 | -.178* | -.003 | 1.000 | .322** | -.060 |
| **Year Fin** | -.030 | -.056 | -.135 | .051 | .048 | .077 | -.046 | .077 | .322** | 1.000 | .009 |
| **Entities** | .470** | .376** | .213** | .143* | .215** | .589** | .223** | .577** | -.060 | .009 | 1.000 |

### 7.2.3 The Miyazaki Data Set

Published in Miyazaki, Terakado, Ozaki & Nozaki (1994), the Miyazaki data set is a record of 48 systems in 20 companies managed by the Fujitsu Large Systems Users Group. It was used in the original study to demonstrate that the least squares of balanced relative errors (LBRS) is superior to the ordinary least squares method (given the presence of outliers in the data set). The original data set contains eight variables as shown in Table 7.5.

Table 7.5
*Variables of Miyazaki Data Set*

| **Mnemonic** | **Variable** | **Explanation** |
|---|---|---|
| KLOC | Lines of code in thousands | Count of COBOL source lines, exclude comment lines, screen and form definition codes, and code copied by the COPY statement. |
| MM | Person Months | Count of effort from systems design to systems test. An MM is defined as 160 hours of working time. |
| SCRN | Number of screens | Count of different input or output screen formats. Screen formats are regarded as different only if data elements are different. |
| FORM | Number of forms | Count of different form (report) formats. Form formats are regarded as different only if data elements are different. |
| FILE | Number of files | Count of input, output, update, and storage files. Intermediate files are excluded. |
| ESCRN | Number of data elements in screens | Count of total data elements in all the screens that are included in the number of screens (SCRN). |
| EFORM | Number of data elements in forms | Count of total data elements in all the screens that are included in the number of forms (FORM). |
| EFILE | Number of data elements in files | Count of total data elements in all the screens that are included in the number of forms (FILE). |

It is evident that ESCRN is related to SCRN, likewise EFORM to FORM, as well EFILE to FILE. Both effort and lines of code are considered as dependent variables relevant to software project estimation. Hence, we selected SCRN, FORM, and FILE that are straightforward to count and understand. Taking into consideration that lines of code in fact can only be counted after the

development has been completed, we selected effort in person-months as the dependent variable.

Kendall's $tau_b$ correlation coefficients for the variables are smaller than 0.5 (see Table 7.6), which imply weak associations between them; therefore, we adopted all three independent variables to build the prediction model.

Table 7.6

*Kendall's $tau_b$ Correlation Coefficient of Variables of Miyazaki Data Set*

| | SCRN | FORM | FILE | Person Months |
|---|---|---|---|---|
| **SCRN** | 1 | .264** | .207* | .466** |
| **FORM** | .264** | 1 | .315** | .353** |
| **FILE** | .207* | .315** | 1 | .396** |
| **Person Months** | .466** | .353** | .396** | 1 |

## 7.3 Viscovery4

In our construction of clustering and prediction models, we used Viscovery® SOMine 4 to facilitate the creation of SOM and FSOM. As a tool that aims to fulfill academic research purposes, Viscovery® SOMine supports analysis of non-linear dependencies, parameter-free clustering, data association and recall, pattern recognition, and other tasks such as animated monitoring (Eudaptics, 1999).

Kohonen's Batch-SOM, a robust variant of unsupervised neural networks, is employed to form Self-Organizing Maps with two-dimensional hexagonal grids. Each hexagonal unit, referred to as a "node", represents a part of the numerical, multivariate source data set. The arrangement of the nodes reveals the neighbourhoods within the data set and the intrinsic shape of the data distribution can be represented by the landscape of the grid.

## 7.4 Fuzzifier

We adopt a fuzzifier which employs the basic Fuzzy C-Means algorithm to pre-process our data set in order to obtain a fuzzy 'version' of the original crisp data. Since we were using the same data (in different splits) to train and recall the models, thus we can consider them as fair comparisons.

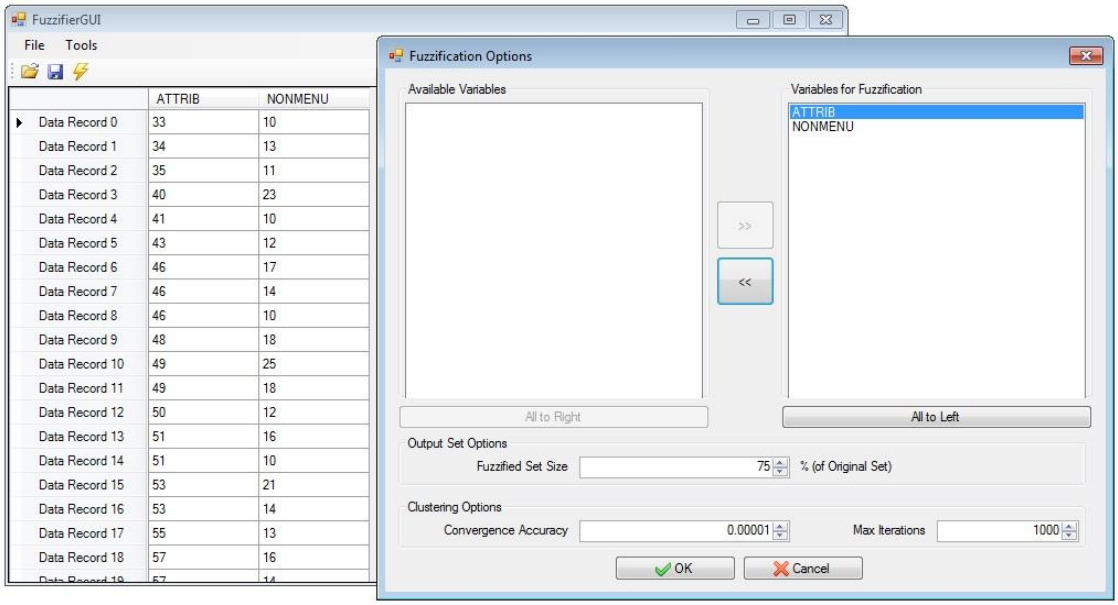


*Figure 7.1.* The User Interface of the Fuzzifier

During our experiment, we found that the clustering options (both convergence accuracy and max iterations, as seen in Figure 7.1) do not have significant impact on our data sets. Therefore, we left them at the default values (which are 0.0001 for convergence accuracy and 10000 for max iterations), and only changed the size of training sets for examining the sensitivity of our models (as the fuzzifier generates centroids based on the size of the fuzzified set). In this respect we used data sets comprising between 50-85% of the original data set size. Since fuzzy c-means clustering utilizes centroids to represent the original data set, the sum of training records and recall records is always smaller than the size of the original data set.

# 8 Model Evaluation and Comparison

Our assessment primarily appraises the accuracy of the predictions of SOM and FSOM by using Linear Regression as a benchmark. To examine the impact of the fuzzy c-means clustering algorithm, we also create and assess a model referred to as Fuzzy Linear Regression.

Each build of the prediction models was executed in the followed steps:

I. Use the fuzzifier to process the original crisp data set for producing training sets;
II. Match the fuzzified data with the original data and label them as the crisp and fuzzy versions of the training set;
III. Pick out unfuzzified data, put them into a new data set and feed it to the fuzzifer;
IV. Label the fuzzified data in the new data set as the recall set;
V. Use the crisp training set to create a SOM model and a Linear Regression model;
VI. Use the fuzzy training set to build an FSOM model and a Fuzzy Linear Regression model;
VII. Use the recall set to test the four different models constructed from the same original data set;
VIII. Compare the predicted size or effort to the actual figure, and calculate the error and absolute error of each software project in every model;
IX. Evaluate the four models based on the sum of absolute error and bias. (Bias = Sum of error / Sum of actual size); and
X. Repeat these nine steps four times to avoid particular sample bias.

Hence, our analysis comprises five tests with very similar parameters (i.e. number of variables, size of training set and size of recall set) in one single build. Table 8.1 shows the parameters of all eight builds in our experiment.

Complete spreadsheets showing training sets and recall sets are provided in Appendices on the enclosed disk. Detailed spreadsheets that assess the estimation errors of the four models for every single data record can also be found on the disk. Clearer views of the data analysis figures in Chapter 8 and

Chapter 9, as well as generated SOM/FSOM maps. are also available on the disk. Appendices in Section 11 indicate the relevant folders.

Table 8.1

*Parameters of Eight Builds*

| Build | Variables | Average Train Set Size | Average Recall Set Size |
|---|---|---|---|
| 4GL Build1 | 2 | 42 | 20 |
| 4GL Build2 | 2 | 28 | 33 |
| 4GL Build3 | 4 | 45 | 20 |
| 4GL Build4 | 6 | 45 | 20 |
| Desharnais Build1 | 1 | 48 | 24 |
| Desharnais Build2 | 1 | 43 | 19 |
| Miyazaki Build1 | 3 | 25 | 16 |
| Miyazaki Build2 | 3 | 32 | 14 |

## 8.1 4GL Build1

As discussed previously, correlation analysis indicated ATTRIB and NONMENU as the two vital variables that together reflect the functional features of the software projects in the 4GL data set. Thus, we used these two variables in our 4GL systems Build1. In this case, we selected 65% of the original set (of 70 records) for the first fuzzified set. After the matching procedure, 42 records were listed in the training set. Then the remaining 28 records were input to the fuzzifier to produce 75% of the data for the recall set. As a result, the recall set contains 20 records after the second matching procedure.

By comparing the actual and predicted size of each project in the recall set (Figure 8.1), we found that the prediction results of SOM and FSOM pair up, likewise Linear Regression and Fuzzy Linear Regression share almost exactly the same result. At the same time, it can be noted that the size estimated by SOM and FSOM share a very close trend with the actual size in this build.

(Note that in this and subsequent analyses we provide indicative results in the chapter, rather than providing all of the outputs. These can be found on the enclosed disk.)

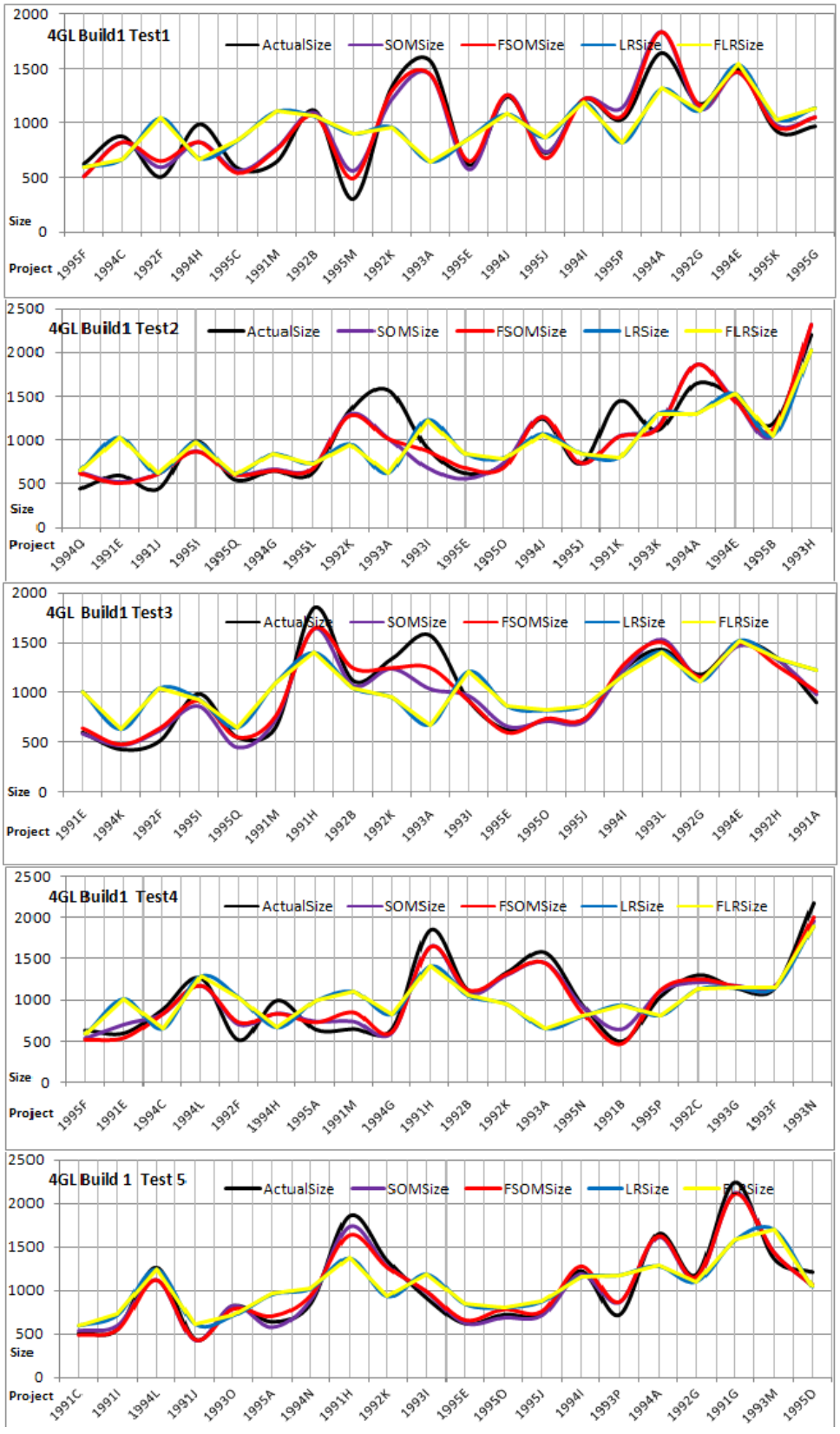


*Figure 8.1.* Comparison of Actual and Predicted Results for 4GL Build1

On the other hand, the sum of absolute errors of each model in five tests (Figure 8.2) illustrate that incorporating fuzzy c-means clustering prior to creating SOM or Linear Regression models do generally improve the accuracy of estimations in this build.

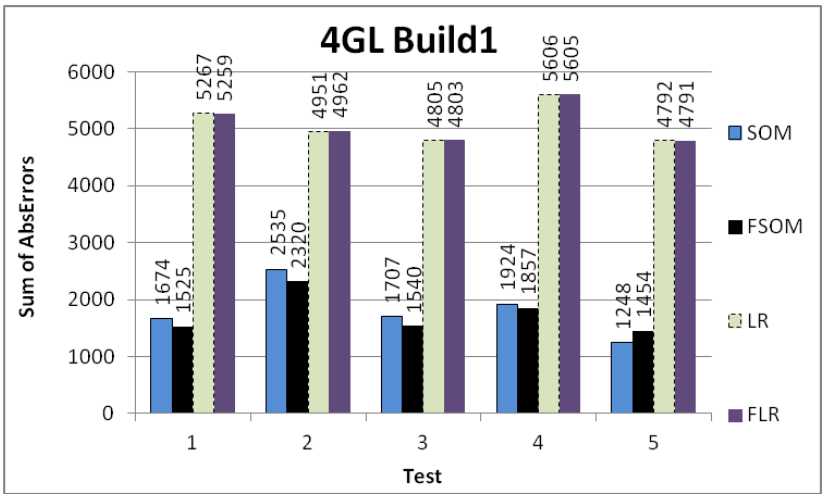


*Figure 8.2.* Sum of Absolute Errors of 4GL Build1

The other parameter we use to evaluate the models is bias. In this build, we found that SOM and FSOM tend to overestimate the size except in Test1, while the bias for Linear Regression and Fuzzy Linear Regression fluctuate more than SOM and FSOM (Table 8.2). From the perspective of bias, we can say that the FSOM model fits the data of this build best, and it also generated the lowest errors overall.

Table 8.2

*Bias of 4GL Build1*

| **Test** | **Bias** | | | |
|---|---|---|---|---|
| | **SOM** | **FSOM** | **LR** | **FLR** |
| **1** | -0.45% | -0.28% | 1.14% | 1.01% |
| **2** | 3.72% | 2.26% | 3.64% | 3.61% |
| **3** | 3.01% | 0.60% | -4.16% | -4.20% |
| **4** | 2.55% | 3.18% | 3.73% | 3.96% |
| **5** | 2.26% | 1.20% | -0.46% | -0.49% |
| **Sum of Absolute Bias** | 11.99% | 7.51% | 13.13% | 13.27% |

## 8.2 4GL Build2

In order to appraise the sensitivity of SOM and FSOM to sampling/split bias, we kept ATTRIB and NONMENU as the two predictor variables and changed the numbers of records in the training and recall sets. Instead of using 60% of the original data in the training sets, we only used 40% in this build, which means more records were left for the recall sets. As a result, we used for each test a recall set comprising approximately 35 records.

From the comparisons of predicted size produced by four different models in each test, we again found that the SOM and FSOM provide more accurate forecasting result than Linear Regression and Fuzzy Linear Regression (Figure 8.3).

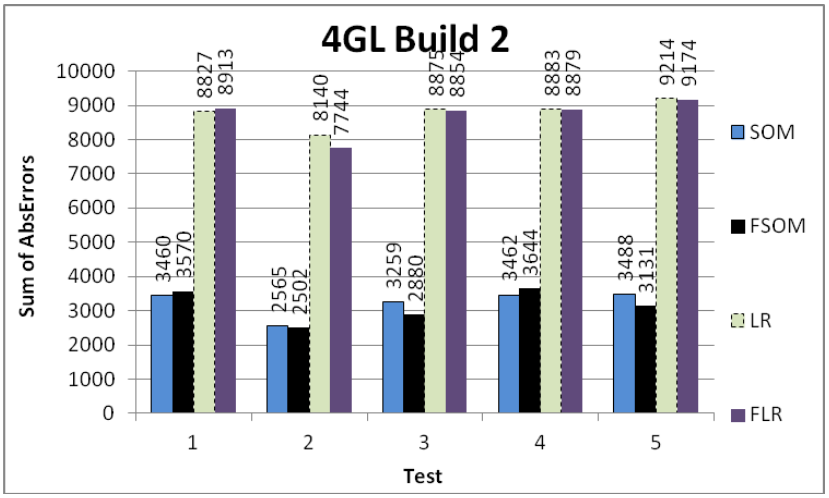


*Figure 8.3.* Sum of Absolute Errors of 4GL Build2

Compared to Build1 with 42 records in the training sets and 28 records in the recall sets, we notice that in Build2 the bias between the actual and estimated size from SOM and FSOM have a tendency to be smaller (Table 8.3). Meanwhile, it is interesting to see that all four models present lager bias in Test2 than other tests, although they all have smaller Sum of Absolute Errors.

We also notice that FSOM was found to be the best in terms of Sum of Absolute Errors, whereas SOM presented the smallest bias in this build.

Table 8.3

*Bias of 4GL Build2*

| Test | Bias | | | |
|---|---|---|---|---|
| | SOM | FSOM | LR | FLR |
| 1 | 0.27% | 1.38% | -1.52% | -1.15% |
| 2 | -4.05% | -2.76% | -19.56% | -18.33% |
| 3 | 1.55% | 0.23% | -1.66% | -0.77% |
| 4 | 0.58% | 1.49% | -3.16% | -2.43% |
| 5 | -0.85% | -1.91% | -8.26% | -7.86% |
| Sum of Absolute Bias | 7.30% | 7.77% | 34.17% | 30.53% |

## 8.3 4GL Build3 & Build4

In order to obtain a fuller insight into the sensitivity of SOM and FSOM, we fed the models with 65% of the original data set as training sets and 29% of the original set as recall sets in both Build 3 and Build 4. In Build 3, however, we kept ATTRIB and NONMENU and added two other variables - ENT and REPORT that have relatively higher correlations with SIZE. In Build 4, we took away NONMENU and FDCSIZE as they are both derived from other variables directly (NONMENU = EDIT + REPORT whereas FDCSIZE = MENU + EDIT + REPORT). In other words, we adopted six elemental variables (ATTRIB, EDIT, ENT, MENU, REPORT, and RSHIP) in Build 4.

The result showed that, the more inadequate variables we provided to the models, the less accurate estimations they produced. We then computed the Mean of Average Absolute Errors in Build 1, 3 and 4 to assess the impact of these variations (Table 8.4). Build 2 was excluded from the table because it is the only one that used 40% of the original data in the training set, against others that used 60-64%. Here,

Average Absolute Errors = Sum of Absolute Errors / Recall set size, and
Mean of Average Absolute Errors = Sum of Average Absolute Errors/5

Table 8.4

*Mean of Absolute Errors of 4GL Builds*

| Build | Variables | Mean of Average Absolute Errors | | | |
|---|---|---|---|---|---|
| | | SOM | FSOM | LR | FLR |
| **1** | 2 | 91 | 87 | 254 | 254 |
| **3** | 4 | 151 | 164 | 288 | 288 |
| **4** | 6 | 211 | 204 | 301 | 311 |

It appears that, as expected, all four models lost their precision when noise was added into the process of creating models. Comparatively, Linear Regression and Fuzzy Linear Regression are in this case steadier than SOM and FSOM when enduring noise, although their results were still worse than those achieved using SOMs.

## 8.4 Desharnais Build1

Considering the correlations among the variables (see Section 4.2.2) in the Desharnais data set, Entities is the only independent variable used in our construction of an effort forecasting model. Each test in this build uses 48 records (i.e. approximately 60% of original data set) in the training set and 24 records (i.e. approximately 40% of the data set) in the recall set.

As in the 4GL builds, the prediction results produced by SOM and FSOM follow the actual effort significantly closely (illustrated by Figure 8.5). In contrast, Linear Regression and Fuzzy Linear Regression only manage to indicate the trend in the overall picture.

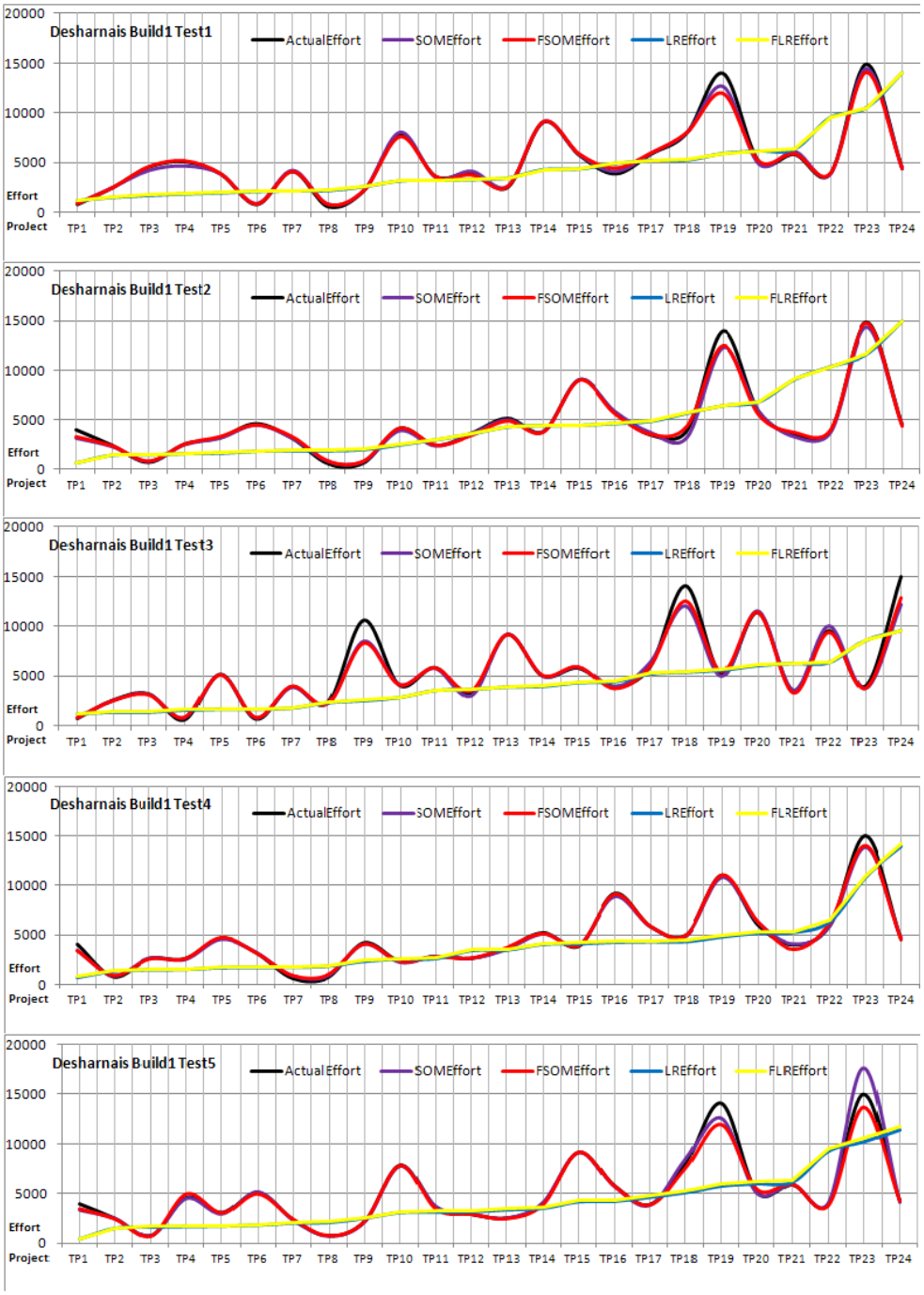


*Figure 8.5.* Comparison of Actual and Predicted Results for Desharnais Build1

We can also argue that SOM and FSOM outperform Linear Regression and Fuzzy Linear Regression in terms of Sum of Absolute Errors (Figure 8.6). However, as the models are used to forecast effort, which is measured in person-hours in the Desharnais data set, whether the prediction result is accurate or not depends on the number of personnel in the particular project. Also, we need to consider the error relative to the number of hours in each project, i.e. the bias.

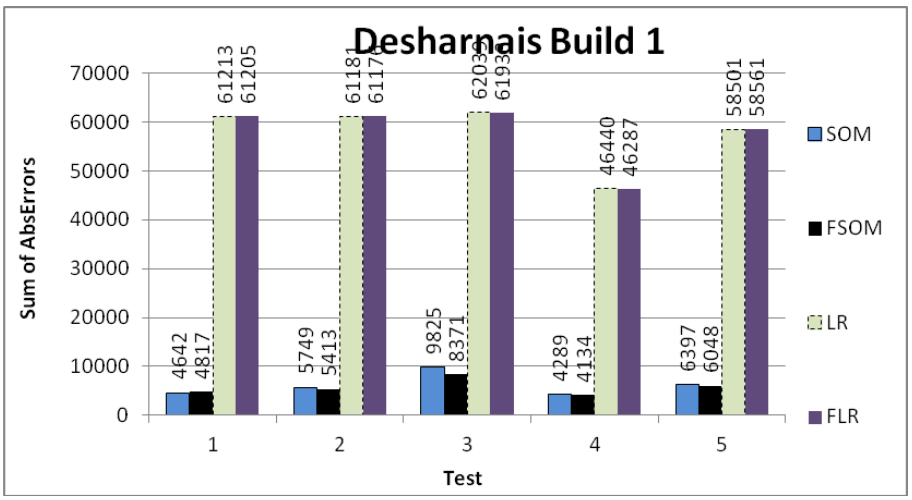


*Figure 8.6.* Sum of Absolute Errors of Desharnais Build1

When appraising the bias of all four models (Table 8.5), it is evident that Linear Regression and Fuzzy Linear Regression are inadequate for a data set that presents features similar to those in the Desharnais data set. Also, the level of acceptable bias depends on the project. In terms of bias SOM performed the best in this build, in contrast to FSOM, and presented the lowest Sum of Absolute Errors.

Table 8.5

*Bias of Desharnais Build1*

| Test | Bias | | | |
|---|---|---|---|---|
| | SOM | FSOM | LR | FLR |
| **1** | 1.98% | 2.35% | 13.76% | 13.43% |
| **2** | 3.31% | 1.85% | -1.28% | -1.26% |
| **3** | 3.81% | 3.20% | 28.66% | 28.35% |
| **4** | 1.01% | 0.79% | 13.10% | 12.08% |
| **5** | -1.07% | 3.73% | 19.16% | 16.44% |
| **Sum of Absolute Bias** | 11.18% | 11.93% | 75.96% | 71.56% |

## 8.5 Desharnais Build2

In the second build of the Desharnais data set the size of training data sets is decreased from 48 to 43 (i.e. approximate 53% of the data set). The size of the recall sets is decreased to 19 (i.e. approximate 24% of the data set). We found

that Linear Regression and Fuzzy Linear Regression predictions match the actual effort much better in Build2 than in Build1. In addition, the precision of SOM and FSOM is improved (see Figure 8.7 and Figure 8.8).

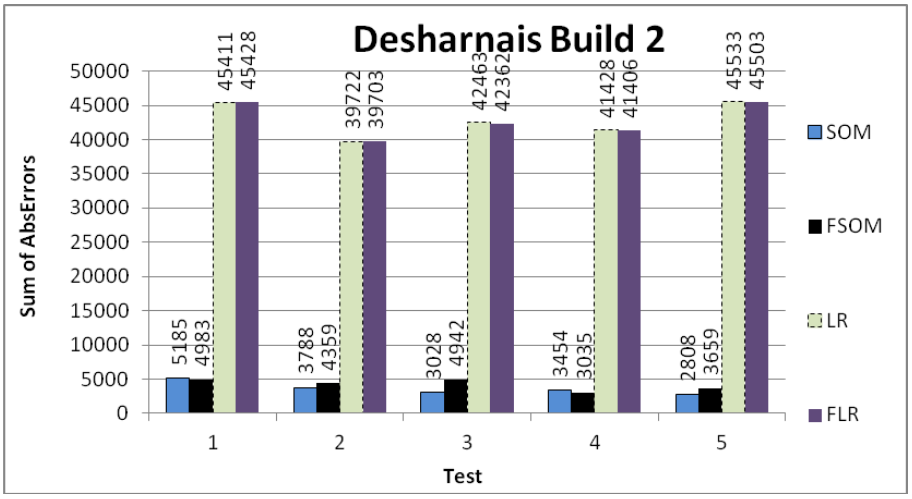


*Figure 8.7*. Sum of Absolute Errors of Desharnais Build2

While examining the bias (Table 8.6), it can be seen that Linear Regression and Fuzzy Linear Regression are overestimating the effort in every test by nearly 19% at the maximum. Whereas the bias of SOM and FSOM prediction results in this build are smaller than in Build1, it can also be seen that FSOM achieved a very low 0.01% bias in Test 1 where SOM has a low bias. Overall, SOM produced the smallest Sum of Absolute Errors in this build while FSOM produced the lowest error from perspective of bias.

Table 8.6

*Bias of Desharnais Build2*

| **Test** | **Bias** | | | |
|---|---|---|---|---|
| | **SOM** | **FSOM** | **LR** | **FLR** |
| **1** | 0.16% | 0.01% | 6.24% | 6.11% |
| **2** | 2.89% | 2.46% | 18.76% | 18.62% |
| **3** | -1.62% | -1.23% | 13.37% | 13.10% |
| **4** | 1.14% | -0.22% | 7.34% | 7.20% |
| **5** | 1.74% | 1.67% | 9.38% | 9.16% |
| **Sum of Absolute Bias** | 7.55% | 5.59% | 55.09% | 54.19% |

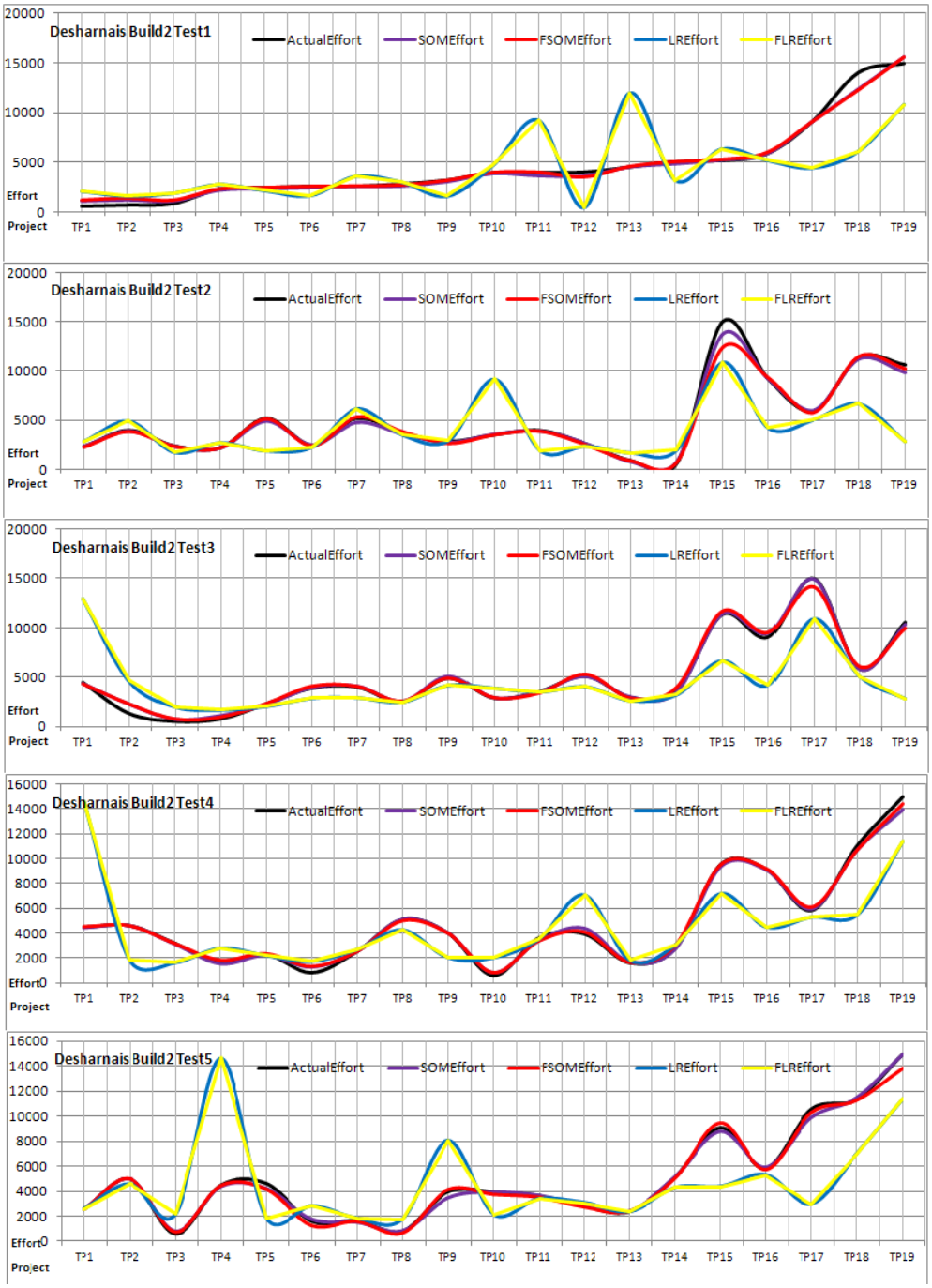


*Figure 8.8.* Comparison of Actual and Predicted Results for Desharnais Build2

## 8.6 Miyazaki Build1

In the first build that utilizes the Miyazaki data set, 25 out of the 48 original records are used as a training set whilst 15 further records are used as a recall set. Three variables - SCRN, FORM, and FILE - are selected in the build as they are easy to understand and can be directly counted in the system.

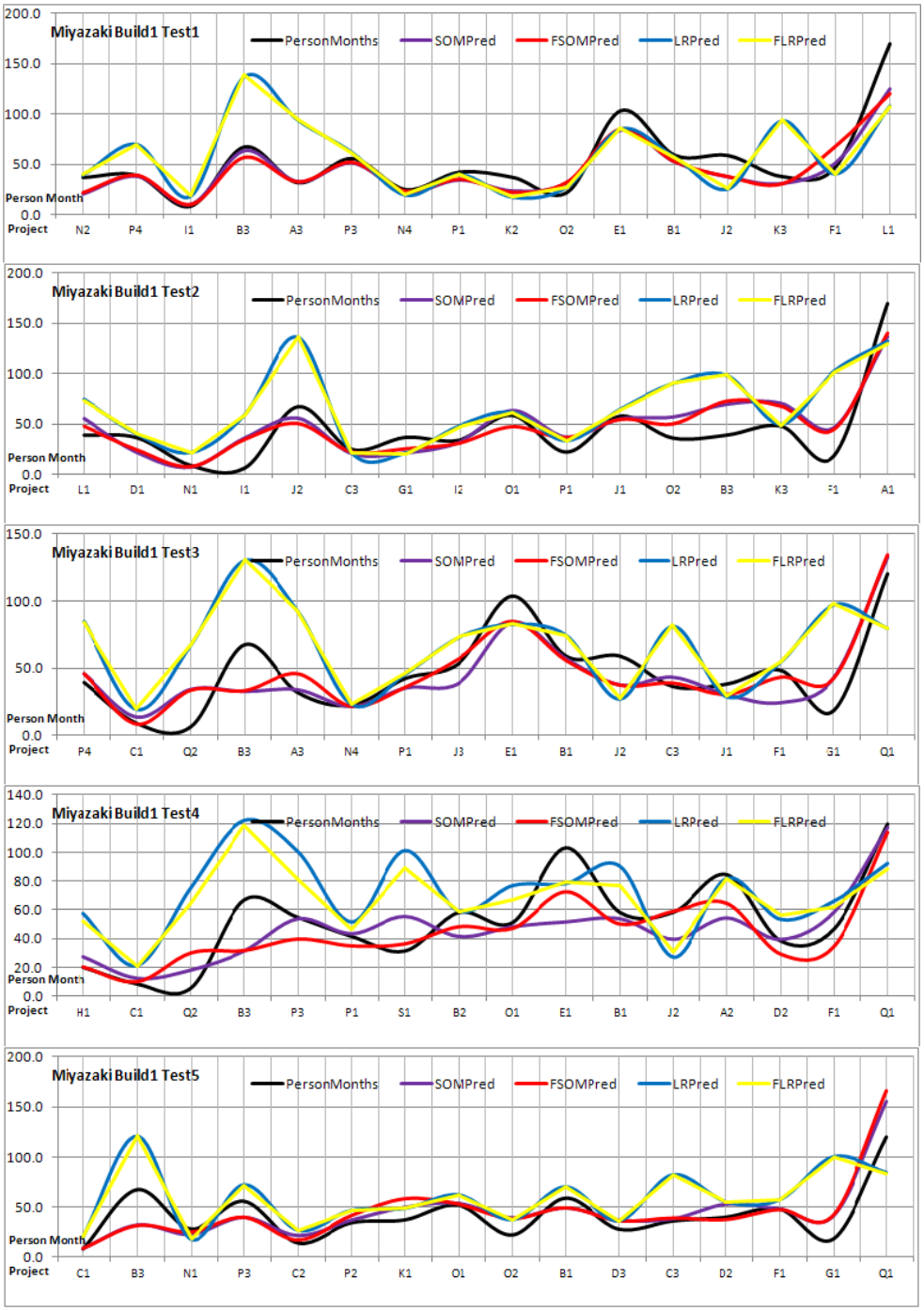


*Figure 8.9*. Comparison of Actual and Predicted Results for Miyazaki Build1

The trend lines in Figure 8.9 indicate that most of the prediction results deviate from the actual results wildly. In this build, when the trend lines of SOM and FSOM split, the gaps between them are more noticeable than they are for the other two data sets, especially in Test3 and Test4. The Sum of Absolute Error (see Figure 8.10) shows that SOM and FSOM are losing their superiority to (but still outperform) Linear Regression and Fuzzy Linear Regression. This is even clearer when appraising the bias.

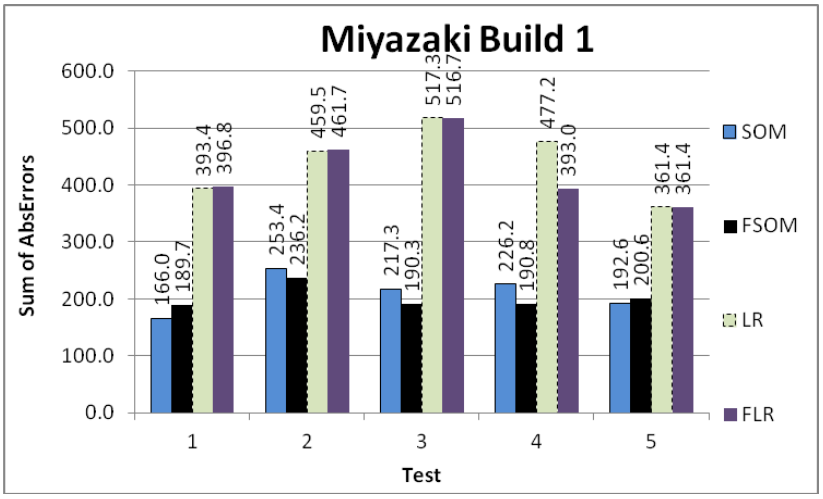


*Figure 8.10.* Sum of Absolute Errors of Miyazaki Build1

On the one hand, the SOM and FSOM forecasts display over 10% bias in most situations. On the other hand, Linear Regression and Fuzzy Linear Regression underestimate the effort in all tests by nearly 50% as shown by the maximum bias (Table 8.7).

Table 8.7

*Bias of Miyazaki Build1*

| Test | Bias | | | |
|---|---|---|---|---|
| | SOM | FSOM | LR | FLR |
| **1** | 15.98% | 14.69% | -11.60% | -11.54% |
| **2** | -11.63% | -8.06% | -48.82% | -48.53% |
| **3** | 5.54% | 0.81% | -42.16% | -42.15% |
| **4** | 12.08% | 15.43% | -35.46% | -25.94% |
| **5** | -9.02% | -10.93% | -40.52% | -40.45% |
| **Sum of Absolute Bias** | 54.25% | 49.93% | 178.54% | 168.61% |

Although strictly speaking SOM and FSOM achieve lower Sums of Absolute Error and bias than Linear Regression and Fuzzy Linear Regression, with Person-Month as the measure, biases that are close to 50% are clearly unacceptable in effort estimation.

## 8.7 Miyazaki Build2

In this build, we keep the same three variables as predictors but we increase the size of the training set to 32 records and reduce the size of the recall set to 14 records.

It is surprising to see in Figure 8.11 that all four models perform much worse for one particular project – J3. By assessing other projects that require similar amounts of Person-Months, we found that J3 has much lower independent variable values except for SCRN. Hence, we have a good reason to believe that project J3 is an outlier.

At the same time, we notice in Figure 8.12 that SOM and FSOM perform less accurately in this build, and their Sums of Absolute Error are closer to those of Linear Regression and Fuzzy Linear Regression. However, FSOM is more accurate than SOM here.

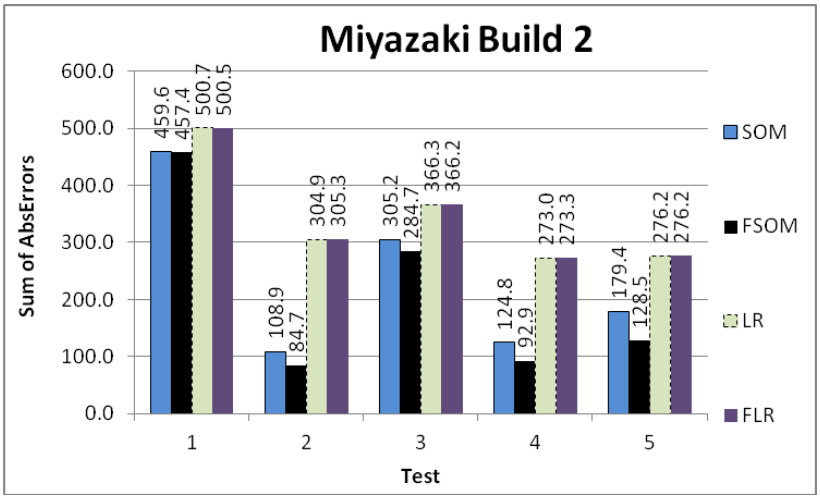


*Figure 8.12.* Sum of Absolute Errors of Miyazaki Build2

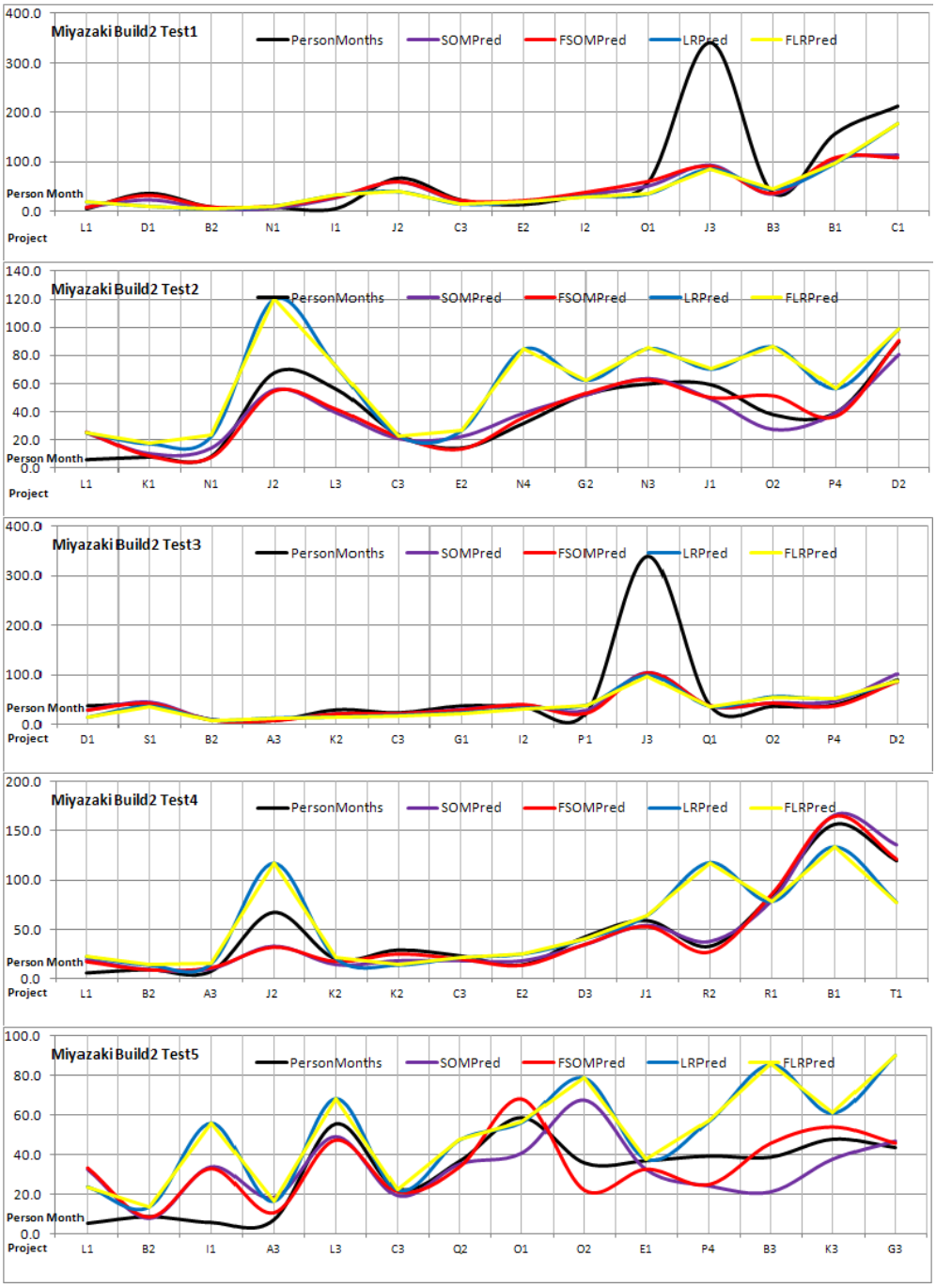


*Figure 8.11.* Comparison of Actual and Predicted Results for Miyazaki Build2

By looking at the bias in Table 8.8, we might say that Linear Regression and its enhanced version, Fuzzy Linear Regression in our case, are highly inappropriate for estimating software projects that are similar to those evident in the Miyazaki data set. Even though SOM and FSOM exhibit better performance, the quality of the predictions is unstable as the biases fluctuate sharply. Furthermore, when Person-Months is used as the unit for counting and predicting the effort, project managers are more sensitive to bias.

Table 8.8

*Bias of Miyazaki Build2*

| Test | Bias | | | |
|---|---|---|---|---|
| | **SOM** | **FSOM** | **LR** | **FLR** |
| **1** | 38.58% | 37.67% | 38.73% | 38.69% |
| **2** | 1.06% | -1.35% | -55.41% | -55.49% |
| **3** | 28.93% | 32.34% | 33.75% | 33.69% |
| **4** | 3.98% | 6.51% | -13.80% | -13.89% |
| **5** | -4.89% | -8.09% | -60.47% | -60.50% |
| **Sum of Absolute Bias** | 77.44% | 85.95% | 202.17% | 202.25% |

# 9 Conclusion

## 9.1 Fuzziness

Since the size of recall sets used in the tests varies significantly, it is inappropriate to evaluate the performance of these four models across builds based on the Sum of Absolute Errors. In order to take into account the influence of absent historical data, we used the Average of Absolute Errors and Mean of Average Absolute Errors as indicators to illustrate the accuracy of different prediction models. Here,

Average of Absolute Errors = Sum of Absolute Errors / Recall set size; and
Mean of Average Absolute Errors = Sum of Average of Absolute Errors / 5

A lower Mean of Average Absolute Errors indicates a more accurate model (Table 9.1). In 4GL Build1 and Build2, all four models perform slightly less accurately when the size of the training set is decreased from 65% of the original data set to 40%. (4GL Build3 and Build4 are not included here as they have different numbers of variables to 4GL Build1 and Build2.)

Table 9.1

*Aspects of Training Data Sets vs. Models' Performance*

| Build | Train Set Size | Total Size | Percentage | Variables | Mean of Average Absolute Errors | | | |
|---|---|---|---|---|---|---|---|---|
| | | | | | SOM | FSOM | LR | FLR |
| 4GL B1 | 42 | 70 | 65% | 2 | 91 | 87 | 254 | 254 |
| 4GL B2 | 28 | 70 | 40% | 2 | 97 | 94 | 262 | 259 |
| Variance | 14 | - | - | 0 | -6 | -7 | -8 | -5 |
| 4GL B3 | 45 | 70 | 64% | 4 | 151 | 164 | 288 | 288 |
| 4GL B4 | 45 | 70 | 64% | 6 | 211 | 204 | 301 | 311 |
| Desharnais B1 | 48 | 81 | 59% | 1 | 258 | 240 | 2411 | 2410 |
| Desharnais B2 | 43 | 81 | 53% | 1 | 192 | 221 | 2258 | 2257 |
| Variance | 5 | - | - | 0 | 65 | 19 | 153 | 153 |
| Miyazaki B1 | 25 | 48 | 52% | 3 | 13.2 | 12.6 | 27.6 | 26.6 |
| Miyazaki B2 | 32 | 48 | 67% | 3 | 16.8 | 15.0 | 24.6 | 24.6 |
| Variance | -7 | - | - | 0 | -3.6 | -2.4 | 3.0 | 2.0 |

When around 6% of the original Desharnais data set was removed from the training set, all four models provided forecasts that were more accurate to

varying degrees. However, when 15% of the original data set was added into the Miyazaki training set, it is interesting to observe that the performance of SOM and FSOM dropped whereas Linear Regression and Fuzzy Linear Regression presented better estimation.

According to the assessment in Table 9.1, we conclude that both SOM and FSOM lose their ability to accommodate ambiguous and vague data when they are over-fed with historical records. They only become reasonably accurate for test data sets that are extremely similar to the training set and are not applicable for wider ranges of data that share the same features. However, when the size of the training set is less than 50% of the original data set (which is 4GL Build2 in our case) all four models lose their accuracy.

It is also important to point out that the degree of fuzziness considered here is not extensive, especially in the 4GL and Miyazaki data sets. In these two data sets, the difference between the original and fuzzified versions of the data are only decimal digits, i.e. within (zero, 1). In contrast, the Desharnais data set has higher degree of fuzziness than the other two data sets. In the Desharnais builds, several original records were represented by one single fuzzified record after fuzzification.

## 9.2 Data Distribution

In order to gain a better insight into the relationship of data distribution and prediction accuracy, we calculated the Absolute Margin between actual size/effort and predicted size/effort for each test. In here,

Absolute Margin = |Sum of actual value in fuzzy training set/Training set size – Sum of actual values in recall set/Recall set size|

Thus, a higher Absolute Margin implies a more uneven data distribution split. At the same time, a lower Average of Absolute Errors means a more accurate model (as defined in Section 9.1). By analyzing the Kendall's $tau_b$ correlation of

the Average of Absolute Errors of the four models against the Absolute Margin in each test, we found that there are associations most of the time (Table 9.2).

Table 9.2

*Kendall's tau$_b$ Correlation between Absolute Margin & Average of Absolute Errors*

| Build | SOM | FSOM | LR | FLR |
|---|---|---|---|---|
| 4GL Build1 | .200 | .200 | .400 | .400 |
| 4GL Build2 | -1.000 | -.800 | -.800 | -.800 |
| 4GL Build3 | .400 | .200 | .200 | .200 |
| 4GL Build4 | .000 | -.200 | .000 | .400 |
| Desharnais Build1 | .200 | .200 | .400 | .400 |
| Desharnais Build2 | 1.000 | .400 | -.400 | -.400 |
| Miyazaki Build1 | .000 | .400 | -.400 | .000 |
| Miyazaki Build2 | -.200 | -.200 | -.200 | -.200 |

A number of these correlations are unexpected association. However, this can be explained by the prediction models being sophisticated. Besides data distribution, there are diverse factors affecting the accuracy of estimation, such as the fuzziness of the data set, the selection of the independent variable(s), the numbers of training cycles (in the case of SOM/FSOM), and so on. Our hypothesis for further work is "*the more even the data distribution the more accurate the prediction model"*. At the current stage, we do not have enough strong evidence to support this assertion.

## 9.3 The Overtraining Issue

In our experiment, the relevance of the overtraining issue with neural networks reported in previous studies is confirmed. In SOM and FSOM, the nodes of the map are generated in the training process. A higher number of nodes requires more training cycles, which implies more system capacity required and longer time consumed.

In the Desharnais builds, apart from constructing the 10000 node maps for comparison with Linear Regression and Fuzzy Linear Regression, we also generated 8000 node maps. It can be seen from Figure 9.1 that in some tests SOM and FSOM produce lower errors with 10000 nodes. Nonetheless, whether

8000 or 10000 nodes are better can only be decided on a case by case basis and is weakly associated to data distribution according to the Kendall's $tau_b$ correlation (see Table 9.1) for the Desharnais data set.

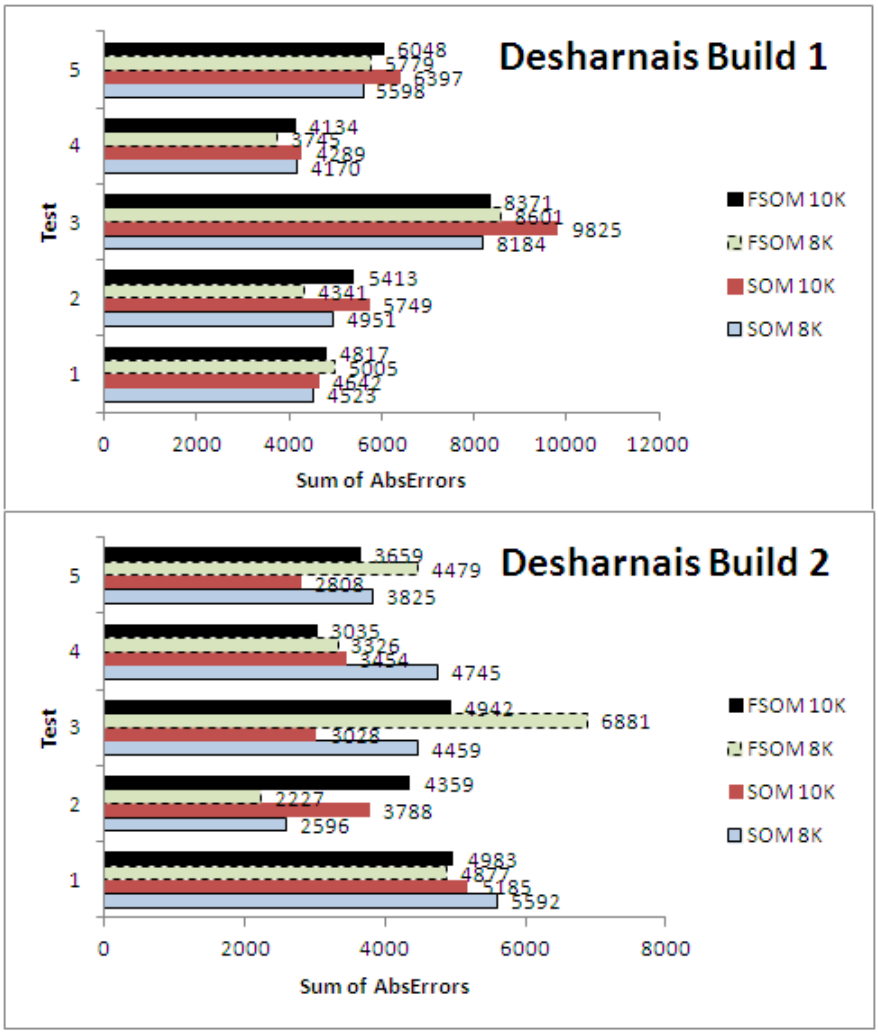


*Figure 9.1.* Sum of Absolute Errors Comparison for the Desharnais Data Set

We also selected Test1 and Test2 in Miyazaki Build1 to observe the influence of the training process. In Test1, we generated 2000, 5000, 8000, 10000 and 20000 node maps, whereas 2000, 8000, and 10000 node maps were used in Test2. A fully trained SOM/FSOM model offers better prediction although an overtrained SOM/FSOM model would be less accurate than a fully trained one. Therefore, we believe a fully trained SOM/FSOM model for Test1 is around 8000 nodes. Meanwhile, a fully trained FSOM model for Test2 is also around

8000 nodes when a fully trained SOM model could have around 10000 nodes or even more, as demonstrated in Figure 9.2.

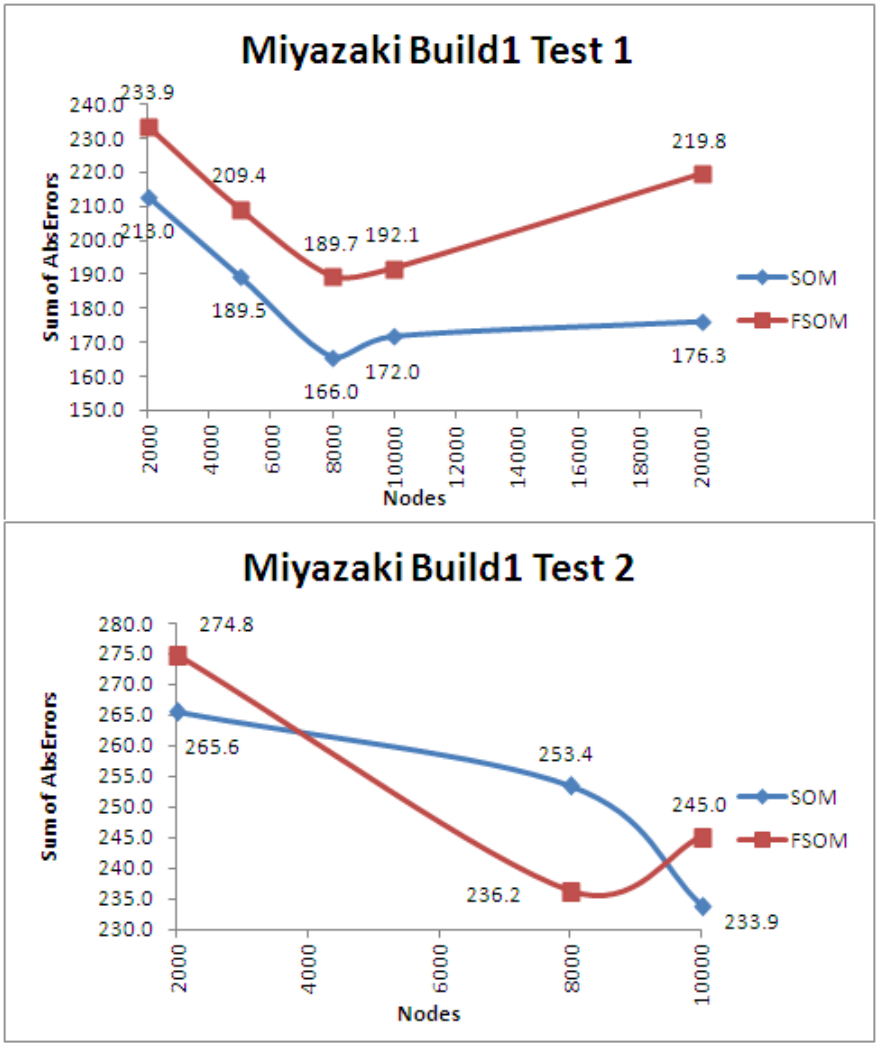


*Figure 9.2.* Sum of Absolute Errors Comparison for Miyazaki Build1 Test 1&2

## 9.4 SOM and FSOM Maps

One of the most novel and significant aspects of SOM is the ability of projecting multi-dimensional data into a two-dimensional map. In software project management, we believe such a characteristic of SOM/FSOM could offer an easy-to-understand straightforward representation of project features from which managers could obtain useful knowledge and understanding of complex problems. To inspect the relationships between the accuracy of estimation and the maps, we utilised the Average of Absolute Errors in each test to calculate the Difference and Absolute Difference of FSOM against SOM. Here,

Difference = SOM Average of Absolute Errors – FSOM Average of Absolute Errors; and
Absolute Difference = |Difference|

The minimum and maximum values of Average of Absolute Errors in Table 9.3 are also highlighted to indicate the best (in green) and worst (in pink) performance of SOM and FSOM within every build.

We notice that FSOM performance is very comparable to that of SOM most of the time, which means when SOM achieves its best (or worst) in a certain test, FSOM also hits its peak (or dip) in that same test. The exceptions are in 4GL Build3, 4GL Build4, and Desharnais Build2, where SOM and FSOM do not perform best at the same time. In our experiment, no evidence was found to show that there is any association among the accuracy of estimation (Average of Absolute Errors), the difference between SOM and FSOM prediction (Difference and Absolute Difference), and the presentation of the maps.

For instance, while the maps in 4GL Build1 Test2 are similar, yet SOM and FSOM are at their worst and the Absolute Difference is the highest for the build. When both achieved the minimum of Average of Absolute Errors with the smallest Absolute Difference in 4GL Build2 Test2, the utility of the maps of SOM are different to that of FSOM. Desharnais Build1 Test3 presents less accuracy and the highest difference between maps. Miyazaki Build2 Test1 is the worst in terms of accuracy, however, Absolute Difference is the minimum for the build and maps are dissimilar.

Table 9.3

*Overall Performance Comparison*

| | | Test1 | Test2 | Test3 | Test4 | Test5 | Min | Max |
|---|---|---|---|---|---|---|---|---|
| **4GL B1** | SOM Avg of AbsErr | 84 | 127 | 85 | 96 | 62 | 62 | 127 |
| | FSOM Avg of AbsErr | 76 | 116 | 77 | 93 | 73 | 73 | 116 |
| | Difference | 7 | 11 | 8 | 3 | -10 | - | - |
| | AbsDifference | 7 | 11 | 8 | 3 | 10 | 3 | 11 |
| **4GL B2** | SOM Avg of AbsErr | 102 | 78 | 102 | 105 | 97 | 78 | 105 |
| | FSOM Avg of AbsErr | 105 | 76 | 90 | 110 | 87 | 76 | 110 |
| | Difference | -3 | 2 | 12 | -6 | 10 | - | - |
| | AbsDifference | 3 | 2 | 12 | 6 | 10 | 2 | 12 |
| **4GL B3** | SOM Avg of AbsErr | 205 | 140 | 141 | 131 | 139 | 131 | 205 |
| | FSOM Avg of AbsErr | 218 | 123 | 144 | 176 | 161 | 123 | 218 |
| | Difference | -13 | 17 | -3 | -44 | -23 | - | - |
| | AbsDifference | 13 | 17 | 3 | 44 | 23 | 3 | 44 |
| **4GL B4** | SOM Avg of AbsErr | 208 | 219 | 196 | 202 | 229 | 196 | 229 |
| | FSOM Avg of AbsErr | 170 | 198 | 175 | 199 | 278 | 170 | 278 |
| | Difference | 39 | 21 | 21 | 3 | -49 | - | - |
| | AbsDifference | 39 | 21 | 21 | 3 | 49 | 3 | 49 |
| **DesharnaisB1** | SOM Avg of AbsErr | 193 | 240 | 409 | 179 | 267 | 179 | 409 |
| | FSOM Avg of AbsErr | 201 | 226 | 349 | 172 | 252 | 172 | 349 |
| | Difference | -7 | 14 | 61 | 6 | 15 | - | - |
| | AbsDifference | 7 | 14 | 61 | 6 | 15 | 6 | 61 |
| **DesharnaisB2** | SOM Avg of AbsErr | 273 | 199 | 159 | 182 | 148 | 148 | 273 |
| | FSOM Avg of AbsErr | 262 | 229 | 260 | 160 | 193 | 160 | 262 |
| | Difference | 11 | -30 | -101 | 22 | -45 | - | - |
| | AbsDifference | 11 | 30 | 101 | 22 | 45 | 11 | 101 |
| **MiyazakiB1** | SOM Avg of AbsErr | 10 | 16 | 13.6 | 14.1 | 12.0 | 10.4 | 15.8 |
| | FSOM Avg of AbsErr | 12 | 15 | 11.9 | 11.9 | 12.5 | 11.9 | 14.8 |
| | Difference | -1.5 | 1.1 | 1.7 | 2.2 | -0.5 | - | - |
| | AbsDifference | 1.5 | 1.1 | 1.7 | 2.2 | 0.5 | 0.5 | 2.2 |
| **MiyazakiB2** | SOM Avg of AbsErr | 33 | 8 | 21.8 | 8.9 | 12.8 | 7.8 | 32.8 |
| | FSOM Avg of AbsErr | 33 | 6 | 20.3 | 6.6 | 9.2 | 6.1 | 32.7 |
| | Difference | 0.2 | 1.7 | 1.5 | 2.3 | 3.6 | - | - |
| | AbsDifference | 0.2 | 1.7 | 1.5 | 2.3 | 3.6 | 0.2 | 3.6 |

Figures 9.3-9.6 depict the maps produced in 4GL Build1 Test3 that are one set of the regular outcomes in our experiment. While the maps of clusters in Figure

9.3 and Figure 9.5 present multi-dimensional data in two-dimensional maps, the maps of variables in Figure 9.4 and Figure 9.6 show three dimensions that are variables in this case. The structures of maps for variables in SOM (Figure 9.4) are exactly the same to the structure of the map for clusters in SOM (Figure 9.3), likewise maps in FSOM (Figure 9.6 & 9.5). In the maps for variables, certain colours are used to represent the different values of each variable. The relationship between colours and values are explained in the scales (Eudaptics, 1999). Blue stands for low and red stands for high.

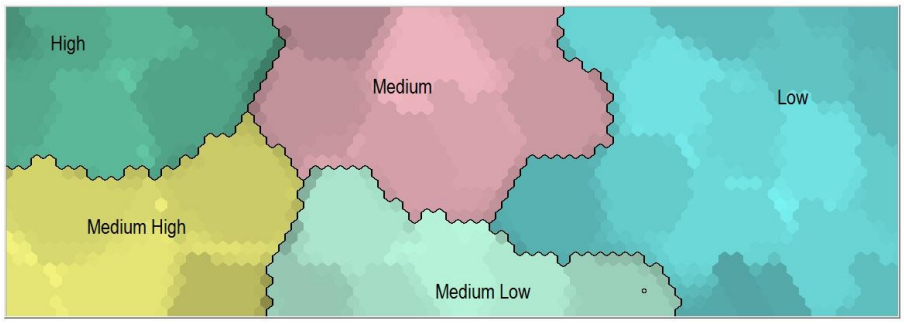


*Figure 9.3.* Map of Clusters in SOM of 4GL Build1 Test3

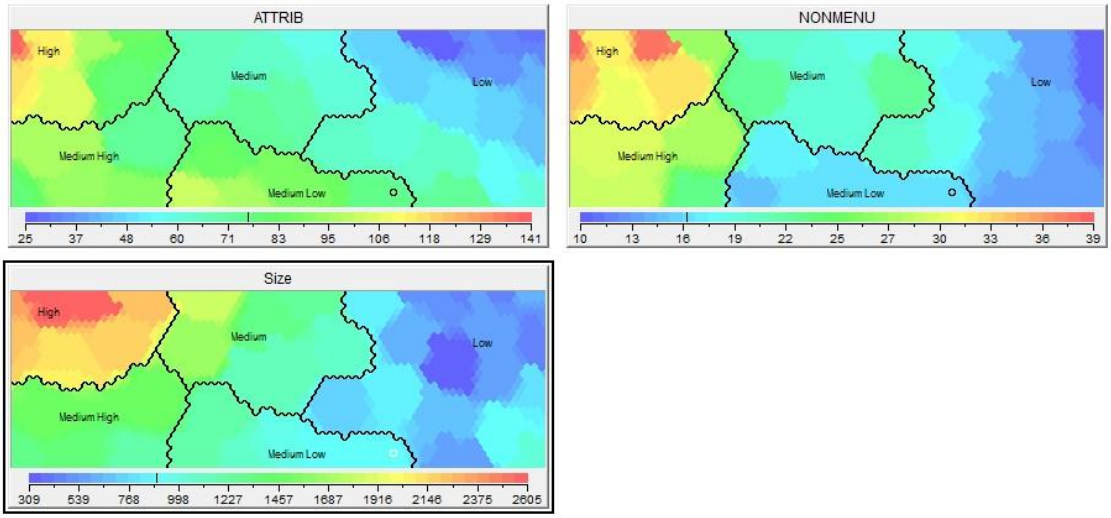


*Figure 9.4.* Maps of Variables in SOM of 4GL Build1 Test3

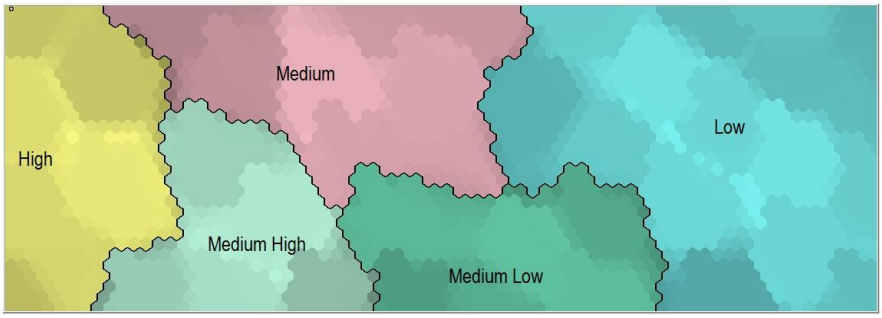


*Figure 9.5.* Map of Clusters in FSOM of 4GL Build1 Test3

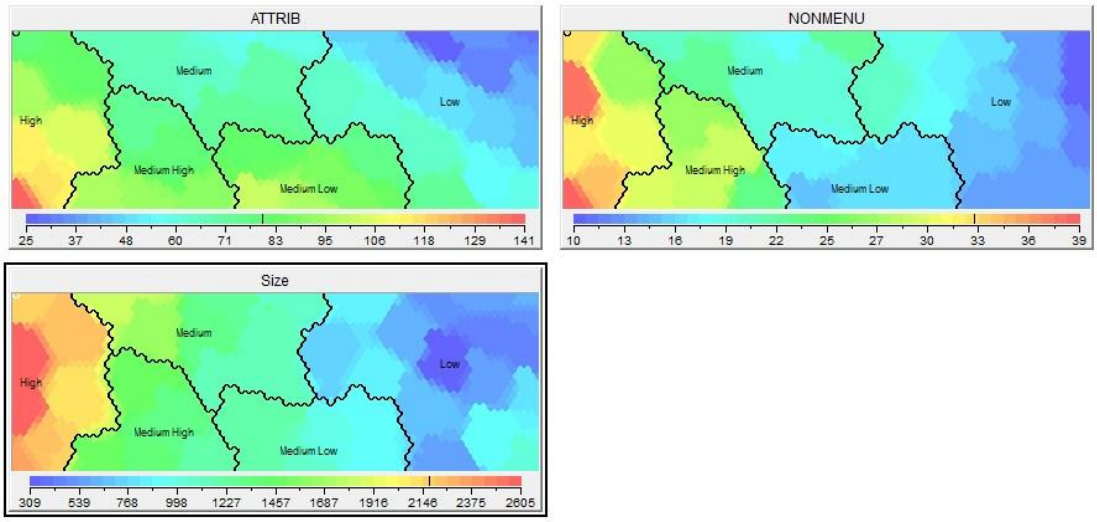


*Figure 9.6.* Maps of Variables in FSOM of 4GL Build1 Test3

It is evident that SOM and FSOM representations cluster the same set of data differently in this test, as well as in most of the other tests, in spite of only minor differences in the original and fuzzified values. Certainly, in some of the tests, such as 4GL Build1 Test2 mentioned above, the structure of maps for variables and clusters in SOM are analogous to maps in FSOM for the same test.

In other words, even though both SOM and FSOM do provide a fair presentation of data distributions for dependent and independent variables, one could not infer whether SOM or FSOM is more accurate by only looking at the maps without consulting the recall outcome.

## 9.5 Summary

Based on the analysis of data presented in Table 9.1, we can assert that for the data sets considered here SOM and FSOM outperform Linear Regression and

Fuzzy Linear Regression in our experiment. This includes cases when noise or overwhelming historical data are introduced into the models, i.e. the SOM and FSOM are overtrained.

According to the differences presented in Table 9.3, FSOM performs better than SOM in 65% (26 out of 40) of the tests. We can then assert that applying the Fuzzy C-Means algorithm prior to creating SOM models improves the accuracy of software project estimation in our experiment.

When predicting software size as in the 4GL data set, SOM and FSOM offer more accurate estimations than Linear Regression. We could also argue the levels of errors and biases are acceptable. When predicting effort, project managers must appraise the estimation results carefully. In the Desharnais data set, Person-Hours is the unit of measure for project effort. Therefore, we could still claim the levels of error and bias of SOM and FSOM are tolerable and recommend SOM/FSOM as a suitable prediction tool. However, when the unit of measure of the project effort is in Person-Months as in the Miyazaki data set, although the errors are statistically small, we could not recommend SOM and FSOM as useful estimation tools. In this case, project managers should cautiously evaluate the performance of SOM/FSOM given their bias.

Meanwhile, we also advocate a thorough correlation analysis to be carried out before constructing prediction models because noise has an impact on accuracy, especially for SOM and FSOM models. In our experiment, SOM and FSOM perform better when the size of the training set is 50-60% (depending on the data set) of the original data set. For that reason, we advise project managers to rebuild each prediction model when significant amounts of new data are acquired in order to achieve more accurate estimations.

## 9.6 Summary of Findings

This study has addressed the research question *"Is the Self-Organising Map an appropriate candidate for estimation in software project management?"*

In our case studies of three different software project data sets, compared to Linear Regression benchmarks, SOM generally provides more accurate estimations of software size and of the personnel effort required in software development. The integration of fuzzy logic techniques, via Fuzzy C-Means in our experiment, is helpful in handling vague and ambiguous real world data. Hence, we regard SOM and FSOM as appropriate candidates for prediction in software project management.

## 9.7 Limitations and Future Study

The foremost limitation of our experiment is that we only used Linear Regression and its transformed version of Fuzzy Linear Regression as the benchmark. Despite the fact that Linear Regression is one of the most commonly used statistical analysis approaches for inference, the focus on the conditional probability distribution restricts its applicability in circumstances that demand the consideration of joint probability distribution. For software project estimation that requires multivariate analysis, when evaluating the performance of SOM and FSOM, project managers should also think about using some alternative modelling methods that give quantitative outputs. For instance:

- Fuzzy inference, which is able to deal with ambiguous data by applying fuzzy logic principles; or
- Support Vector Machine (SVM), which is a set of related supervised machine learning methods for classification and regression analysis; or
- Other forms of Artificial Neural Networks (ANN) which offer non-linear data modelling tools for relationship extraction and pattern recognition.

Another significant limitation is the manner in which each Fuzzy SOM was created. In previous studies, researchers built FSOM using one of two main approaches: either create the two-dimensional map using SOM then apply FCM to cluster the map, or use FCM to fuzzify the original data before forming the two-dimensional map. We utilised the latter method in our experiment, thus future study on the former approach is recommended.

In addition, the fuzzifier that we employed in this experiment implemented a basic Fuzzy C-Means algorithm. As a technique that was developed some time ago, the drawbacks of FCM are well known and researchers have extended the basic Fuzzy C-Means algorithm into diverse enhanced versions. One of the examples is a new version of FCM introduced by Kannan, Devi, Ramathilagam & Sathya (2010). It is able to trim down noise and outliers in large data sets. Bearing in mind the poor performance of SOM and FSOM with the outlier in the Miyazaki data set, it may be that applying this improved version of FCM in creating a Fuzzy SOM would benefit this particular case of software project effort estimation.

Furthermore, we also propose a more thorough investigation of the Miyazaki data set. Even though the SOM and FSOM models statistically outperformed Linear Regression and Fuzzy Linear Regression in our Miyazaki data set builds, the biases of SOM and FSOM are far beyond acceptable limits. In our experiment, two out of three independent variables that were used to form the models displayed low Kendall's $tau_b$ correlation to the dependent variable – effort in Person Months. Therefore, it would be useful to consider the use of other combinations of variables in this model before revisiting the performance of SOM and FSOM with the Miyazaki data set.

Since we believe project managers could benefit from using the two-dimensional maps that offer straightforward representations of data, we suggest a further empirical study of the relationships between the accuracy of estimation and the presentation of SOM and FSOM maps, as perceived by project managers. This would allow project managers to determine under what circumstances FSOM is more or less appropriate than SOM for the estimation of a particular project or set of projects. We also encourage investigation of the hypothesis: "*the more even the data distribution is the more accurate the prediction model would be"*.

Finally, understanding the relationship between the training set and the fully optimized size of SOM/FSOM (i.e. the best number of nodes) is an area that is worth future study. Without doubt, the optimised SOM and FSOM achieve their

best outcomes in terms of resulting in the lowest prediction errors than they do at other sizes.

# 10 References


Adams, L. A., & Courtney, J. F. (2004). *Achieving relevance in IS research via the DAGS framework.* Paper presented at the Proceedings of the 37th Annual Hawaii International Conference on System Sciences, 2004. Retrieved from doi:10.1109/HICSS.2004.1265615

Arndt, S., Turvey, C., & Andreasen, N. C. (1999). Correlating and predicting psychiatric symptom ratings: Spearmans r versus Kendalls tau correlation. *Journal of Psychiatric Research, 33*(2), 97-104. Retrieved from doi:10.1016/S0022-3956(98)90046-2

Aroba, J., Cuadrado-Gallego, J. J., Sicilia, M.-A., Ramos, I., & Garcia-Barriocanal, E. (2008). Segmented software cost estimation models based on fuzzy clustering. *Journal of Systems and Software, 81*(11), 1944-1950. Retrieved from doi:10.1016/j.jss.2008.01.016

Azzeh, M., Neagu, D., & Cowling, P. I. (2010). Fuzzy grey relational analysis for software effort estimation *Empirical Software Engineering, 15*(1), 60-90. Retrieved from doi:10.1007/s10664-009-9113-0

Berlin, S., Raz, T., Glezer, C., & Zviran, M. (2009). Comparison of estimation methods of cost and duration in IT projects. *Information and Software Technology, 51*(4), 738-748. Retrieved from doi:10.1016/j.infsof.2008.09.007

Bezdek, J. C., Ehrlich, R., & Full, W. (1984). FCM: The fuzzy c-means clustering algorithm. *Computers & Geosciences, 10*(2-3), 191-203. Retrieved from doi:10.1016/0098-3004(84)90020-7

Bezdek, J. C., Tsao, E. C. K., & Pal, N. R. (1992). *Fuzzy Kohonen clustering networks.* Paper presented at the IEEE International Conference on Fuzzy Systems, 1992. Retrieved from doi:10.1109/FUZZY.1992.258797

Boehm, B. W., & Valerdi, R. (2008). Achievements and Challenges in Cocomo-Based Software Resource Estimation. *IEEE Software, 25*(5), 74-83. Retrieved from doi:10.1109/MS.2008.133

Burgess, C. J., & Lefley, M. (2001). Can genetic programming improve software effort estimation? A comparative evaluation. *Information and Software Technology, 43*(14), 863-873. Retrieved from doi:10.1016/S0950-5849(01)00192-6

Chaudhuri, A., De, K., & Chatterjee, D. (2008). *A Study of the Traveling Salesman Problem Using Fuzzy Self Organizing Map.* Paper presented at the IEEE Region 10 and the Third international Conference on Industrial and Information Systems, 2008. Retrieved from doi:10.1109/ICIINFS.2008.4798469

Chen, N. (2005). Fuzzy Classification Using Self-Organizing Map and Learning Vector Quantization *Data Mining and Knowledge Management, 3327/2005*, 41-50. Retrieved from doi:10.1109/FSKD.2008.149

Chen, N., & Chen, A. (2008). *A Fuzzy and Hybrid Clustering Framework Using Self-Organizing Map.* Paper presented at the FSKD '08. Fifth International Conference on Fuzzy Systems and Knowledge Discovery, 2008. Retrieved from doi:10.1109/FSKD.2008.149

Chi, S.-C., Kuo, R.-J., & Teng, P.-W. (2000). *A fuzzy self-organizing map neural network for market segmentation of credit card.* Paper presented at the 2000 IEEE International Conference on Systems, Man, and Cybernetics. Retrieved from doi:10.1109/ICSMC.2000.886571

Croux, C., & Dehon, C. (2010). Influence functions of the Spearman and Kendall correlation measures *Statistical Methods & Applications, 19*(4), 497-515. Retrieved from doi:10.1016/S0950-5849(01)00192-6

Dick, S., Meeks, A., Last, M., Bunke, H., & Kandel, A. (2004). Data mining in software metrics databases. *Fuzzy Sets and Systems, 145*(1), 81-110. Retrieved from doi:10.1016/j.fss.2003.10.006

Dunn, J. C. (1973). A Fuzzy Relative of the ISODATA Process and Its Use in Detecting Compact Well-Separated Clusters. *Journal of Cybernetics, 3*(3), 32-57. Retrieved from doi:10.1080/01969727308546046

Eudaptics. (1999). *Viscovery® SOMine Version 4.0 – User's Manual*: Eudaptics software gmbh.

Flexer, A. (2001). On the use of self-organizing maps for clustering and visualization. *Intelligent Data Analysis, 5*(5), 373.

Giraudel, J. L., & Lek, S. (2001). A comparison of self-organizing map algorithm and some conventional statistical methods for ecological community ordination. *Ecological Modelling, 146*(1-3), 329-339. Retrieved from doi:10.1016/j.fss.2003.10.006

Gray, A. R., & MacDonell, S. G. (1997). A comparison of techniques for developing predictive models of software metrics. *Information and Software Technology, 39*(6), 425-437. Retrieved from doi:10.1016/S0950-5849(96)00006-7

Gregor, S. (2006). The nature of theory in information systems. *Management Information Systems Quarterly, 30*(3), 611-642.

Grzegorzewski, P. (2009). Kendall's correlation coefficient for vague preferences. *Soft Computing - A Fusion of Foundations, Methodologies and Applications, 13*(11), 1055-1061. Retrieved from doi:10.1007/s00500-008-0378-9

He, N. (2009). *A Fast Self-Organizing Map Algorithm by Using Genetic Selection.* Paper presented at the Third International Symposium on Intelligent Information Technology Application, 2009. IITA 2009. . Retrieved from doi:10.1109/IITA.2009.291

Hevner, A. R., March, S. T., Park, J., & Ram, S. (2004). Design Science in Information Systems Research. *Management Information Systems Quarterly, 28*(1), 75-106 Retrieved from http://www.jstor.org/pss/25148625

Holz, H. J., Applin, A., Haberman, B., Joyce, D., Purchase, H., & Reed, C. (2006). *Research methods in computing: what are they, and how should we teach them?* Paper presented at the Working group reports on ITiCSE on Innovation and technology in computer science education, Bologna, Italy. Retrieved from doi: 10.1145/1189215.1189180

Hsiao, W.-F., Lin, H.-H., & Chang, T.-M. (2008). Fuzzy consensus measure on verbal opinions. *Expert Systems with Applications, 35*(3), 836-842. Retrieved from doi:10.1016/j.eswa.2007.07.040

Huang, S.-J., Chiu, N.-H., & Liu, Y.-J. (2008). A comparative evaluation on the accuracies of software effort estimates from clustered data. *Information and Software Technology, 50*(9-10), 879-888. Retrieved from doi:10.1016/j.infsof.2008.02.005

Jain, A. K., Murty, M. N., & Flynn, P. J. (1999). Data Clustering: A Review. *ACM Computing Surveys, 31*(3), 264. Retrieved from doi:10.1145/331499.331504

Jorgensen, M., & Shepperd, M. (2007). A Systematic Review of Software Development Cost Estimation Studies. *IEEE Transactions on Software Engineering, 33(1),* 33-53. Retrieved from doi:10.1109/tse.2007.256943

Kannan, S. R., Devi, R., Ramathilagam, S., & Sathya, A. (2010). Some robust objectives of FCM for data analyzing. *Applied Mathematical Modelling, 35*(5), 2571-2583. Retrieved from doi:10.1016/j.apm.2010.11.020

Keung, J. W., Kitchenham, B. A., & Jeffery, D. R. (2008). Analogy-X: Providing Statistical Inference to Analogy-Based Software Cost Estimation. *IEEE Transactions on Software Engineering, 34*(4), 471-484. Retrieved from doi:10.1109/TSE.2008.34

Kocaguneli, E., Menzies, T., & Keung, J. (2011). On the Value of Ensemble Effort Estimation. *IEEE Transactions on Software Engineering, PP(99),* 1-14. Retrieved from doi:10.1109/tse.2011.111

Kohonen, T. (1981). *Automatic Formation of Topological Maps of Patterns in a Self-Organizing System.* Paper presented at the Proceedings of the 2nd Scandinavian Conference on Image Analysis. Retrieved from citeulike-article-id:793951

Kohonen, T. (1999). Fast Evolutionary Learning with Batch-Type Self-Organizing Maps *Neural Processing Letters*, 9(2), 153-162. Retrieved from doi:10.1023/A:1018681526204

Kohonen, T. (2008). Data Management by Self-Organizing Maps In *Computational Intelligence: Research Frontiers*, 5050/2008, 309-332: Springer Berlin / Heidelberg. Retrieved from doi:10.1007/978-3-540-68860-0_15

Kohonen, T., Nieminen, I. T., & Honkela, T. (2009). On the Quantization Error in SOM vs. VQ: A Critical and Systematic Study In *Advances in Self-Organizing Maps,* 5629/2009, 133-144: Springer Berlin / Heidelberg. Retrieved from doi:10.1007/978-3-642-02397-2_16

Kraemer, H. C. (2006). Correlation coefficients in medical research: from product moment correlation to the odds ratio. *Statistical Methods in Medical Research, 15*(6), 525-545. Retrieved from doi:10.1177/0962280206070650

Kuo, R. J., Chi, S. C., & Teng, P. W. (2001). Generalized part family formation through fuzzy self-organizing feature map neural network. *Computers & Industrial Engineering, 40*(1-2), 79-100. Retrieved from doi:10.1016/S0360-8352(00)00073-5

Kurbel, K. E. (2008). Software Project Management In *The Making of Information Systems,* 473-531: Springer Berlin Heidelberg. Retrieved from doi:10.1007/978-3-540-79261-1_8

Kurd, Z., & Kelly, T. P. (2007). Using fuzzy self-organising maps for safety critical systems. *Reliability Engineering & System Safety, 92*(11), 1563-1583. Retrieved from doi:10.1016/j.ress.2006.10.005

Lee, K. C., Cho, H. R., & Kim, J. S. (2007). A self-organizing feature map-driven approach to fuzzy approximate reasoning. *Expert Systems with Applications, 33*(2), 509-521. Retrieved from doi:10.1016/j.eswa.2006.05.031

Li, S.-T., Cheng, Y.-C., & Lin, S.-Y. (2008). A FCM-based deterministic forecasting model for fuzzy time series. *Computers & Mathematics with Applications, 56*(12), 3052-3063. Retrieved from doi:10.1016/j.camwa.2008.07.033

Li, S.-T., Kuo, S.-C., & Tsai, F.-C. (2010). An intelligent decision-support model using FSOM and rule extraction for crime prevention. *Expert Systems with Applications, 37*(10), 7108-7119. Retrieved from doi:10.1016/j.eswa.2010.03.004

MacDonell, S. G. (2003). Software source code sizing using fuzzy logic modeling. *Information and Software Technology, 45*(7), 389-404. Retrieved from doi:10.1016/S0950-5849(03)00011-9

MacDonell, S. G. (2005). *Visualization and analysis of software engineering data using self-organizing maps.* Paper presented at the 2005 International Symposium on Empirical Software Engineering, 2005. Retrieved from doi:10.1109/ISESE.2005.1541820

MacDonell, S. G., & Gray, A. R. (2003). Applying Fuzzy Logic Modeling to Software Project Management. In T. M. Khoshgoftaar (Ed.), *In Software Engineering with Computational Intelligence* (pp. 17-43). Boston MA, USA.

March, S. T., & Smith, G. F. (1995). Design and natural science research on information technology. *Decision Support Systems, 15*(4), 251-266. Retrieved from doi:10.1016/0167-9236(94)00041-2

Miyazaki, Y., Terakado, M., Ozaki, K., & Nozaki, H. (1994). Robust regression for developing software estimation models. *Journal of Systems and Software, 27*(1), 3-16. Retrieved from doi:10.1016/0164-1212(94)90110-4

Moreno, M.N., Ramos, I., García, F.J. & Toro, M. (2008). An association rule mining method for estimating the impact of project management policies on software quality, development time and effort. *Expert Systems with Applications, 34*(1), 522-529. Retrieved from doi:10.1016/j.eswa.2006.09.022

Muzaffar, Z., & Ahmed, M. A. (2010). Software development effort prediction: A study on the factors impacting the accuracy of fuzzy logic systems. *Information and Software Technology, 52*(1), 92-109. Retrieved from doi:10.1016/j.infsof.2009.08.001

Nomura, T., & Miyoshi, T. (1995). *An adaptive rule extraction with the fuzzy self-organizing map and a comparison with other methods.* Paper presented at the Proceedings of ISUMA - NAFIPS '95 The Third International Symposium on Uncertainty Modeling and Analysis and Annual Conference of the North American Fuzzy Information Processing Society. Retrieved from doi:10.1109/ISUMA.1995.527713

Nunamaker, J. F., Chen, M., & Purdin, T. D. M. (1990). System Development in Information Research. *Journal of Management of Information System, 7*(3), 89-106.

O'Gorman, T. W., & Woolson, R. F. (1995). Using Kendall's tau b Correlations to Improve Variable Selection Methods in Case-Control Studies. *Biometrics, 51*(4), 1451-1460. Retrieved from http://www.jstor.org/stable/2533275

Pedrycz, W., & Succi, G. (2005). Genetic granular classifiers in modeling software quality. *Journal of Systems and Software, 76*(3), 277-285. Retrieved from doi:10.1016/j.jss.2004.06.018

Peffers, K., Tuunanen, T., Gengler, C. E., Rossi, M., Hui, W., Virtanen, V., et al. (2006). *The Design Science Research Process: A Model for Producing and Presenting Information Systems Research (DESRIST 2006)* Paper presented at the First International Conference on Design Science Research in Information Systems and Technology (DESRIST 2006), Claremont, California. Retrieved from http://www.mendeley.com/research/the-design-science-research-process-a-model-for-producing-and-presenting-information-systems-research/

Pfleeger, S. L. (2001). Planning and Managing the Project. In *Software engineering: theory and practice* (2nd Edition) (pp. 77-134). Upper Saddle River, NJ: Prentice Hall.

Pressman, R. S. (2001). Managing Software Projects. In *Software engineering: A practitioner's approach* (5th Edition) (pp. 53-241). New York, NY: McGraw-Hill.

Raju, G., Thomas, B., Kumar, T., & Thinley, S. (2008). Integration of Fuzzy Logic in Data Mining to Handle Vagueness and Uncertainty. In *Advanced Intelligent Computing Theories and Applications. With Aspects of Artificial Intelligence,* 5227, 880-887: Springer Berlin / Heidelberg. Retrieved from doi:10.1007/978-3-540-85984-0_106

Seliya, N., & Khoshgoftaar, T. M. (2007). Software Quality Analysis of Unlabeled Program Modules With Semisupervised Clustering. *Systems, Man and Cybernetics, Part A: Systems and Humans, IEEE Transactions on, 37*(2), 201-211. Retrieved from doi:10.1109/TSMCA.2006.889473

Seliya, N., Khoshgoftaar, T. M., & Zhong, S. (2005). *Analyzing software quality with limited fault-proneness defect data.* Paper presented at the Ninth IEEE International Symposium on High-Assurance Systems Engineering, 2005. HASE 2005. Retrieved from doi:10.1109/HASE.2005.4

Somervuo, P., & Kohonen, T. (1999). Self-Organizing Maps and Learning Vector Quantization for Feature Sequences *Neural Processing Letters, Volume 10*(Number 2 / October, 1999), 151-159. Retrieved from doi:10.1023/A:1018741720065

Somervuo, P., & Kohonen, T. (2000). Clustering and Visualization of Large Protein Sequence Databases by Means of an Extension of the Self-Organizing Map In *Discovery Science,* 1967/2000, 76-85: Springer Berlin / Heidelberg. Retrieved from doi:10.1007/3-540-44418-1_7

Sommerville, I. (2007 A). Project management. In *Software engineering* (8th Edition) (pp. 92-113). Edinburgh: Pearson Education Limited.

Sommerville, I. (2007 B). Software cost estimation. In *Software engineering* (8th Edition) (pp. 92-113). Edinburgh: Pearson Education Limited.

Srinivas, V. V., Tripathi, S., Rao, A. R., & Govindaraju, R. S. (2008). Regional flood frequency analysis by combining self-organizing feature map and fuzzy clustering. *Journal of Hydrology, 348*(1-2), 148-166. Retrieved from doi:10.1016/j.jhydrol.2007.09.046

Sum, J., & Chan, L.-W. (1994). *Fuzzy self-organizing map: mechanism and convergence.* Paper presented at the IEEE World Congress on Computational Intelligence. IEEE International Conference on Neural Networks, 1994. Retrieved from doi:10.1109/ICNN.1994.374408

Sun, H., Wang, S., & Jiang, Q. (2004). FCM-Based Model Selection Algorithms for Determining the Number of Clusters. *Pattern Recognition, 37*(10), 2027-2037. Venable, J. R. (2006). *The Role of Theory and Theorising in Design Science Research.* Paper presented at the First International Conference on Design Science Research in Information Systems and Technology (DESRIST 2006) Claremont, California. Retrieved from http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.110.2475&rep=rep1&type=pdf

Vesanto, J., & Alhoniemi, E. (2000). Clustering of the self-organizing map. *IEEE Transactions on Neural Networks, 11*(3), 586-600. Retrieved from doi:10.1109/72.846731

Vuorimaa, P. (1994 A). Fuzzy self-organizing map. *Fuzzy Sets and Systems, 66*(2), 223-231. Retrieved from doi:10.1016/0165-0114(94)90312-3

Vuorimaa, P. (1994 B). *Use of the Fuzzy Self-Organizing Map in pattern recognition.* Paper presented at the Proceedings of the Third IEEE Conference on Fuzzy Systems, 1994. IEEE World Congress on Computational Intelligence. Retrieved from doi:10.1109/FUZZY.1994.343837

Wang, L., & Qi, F. (1999). *Adaptive fuzzy Kohonen clustering network for image segmentation.* Paper presented at the IJCNN '99. International Joint Conference on Neural Networks, 1999. Retrieved from 10.1109/IJCNN.1999.833498

Yang, H.-C. (2009). Automatic generation of semantically enriched web pages by a text mining approach. *Expert Systems with Applications, 36*(6), 9709-9718. Retrieved from doi:10.1016/j.eswa.2009.02.022

Yuan, X., Khoshgoftaar, T. M., Allen, E. B., & Ganesan, K. (2000). *An application of fuzzy clustering to software quality prediction.* Paper presented at the 3rd IEEE Symposium on Application-Specific Systems and Software Engineering Technology, 2000. Retrieved from doi:10.1109/ASSET.2000.888052

Zadeh, L. A. (1965). Fuzzy sets. *Information and Control, 8*(3), 338-353. Retrieved from doi:10.1016/S0019-9958(65)90241-X

Zhong, S., Khoshgoftaar, T. M., & Seliya, N. (2004). *Unsupervised learning for expert-based software quality estimation.* Paper presented at the Eighth IEEE International Symposium on High Assurance Systems Engineering, 2004. Retrieved from doi:10.1109/HASE.2004.1281739

# 11 Appendices

## Appendix A. Training Sets and Recall Sets

Eight excel files of training sets and recall sets are provided in this appendix. One file includes five tests for one build. Please refer to the enclosed disk for the image files.

## Appendix B. SOM and FSOM Maps of Variables and Clusters

This appendix includes SOM and FSOM maps generated in the eight builds. Please refer to the enclosed disk for the image files.

## Appendix C. Evaluation of Effort Estimation Spreadsheets

This appendix is comprised by eight files that appraise the estimation errors of the four models in every single data records. Please refer to the enclosed disk for the Excel files.

## Appendix D. Comparison of Actual and Predicted Results

This appendix contains eight line charts from the eight builds that compare actual and predicted results. Please refer to the enclosed disk for the image files.

## Appendix E. Data Analysis Tables and Figures

The file in this appendix provides overall data analysis in detail. Please refer to the enclosed disk for the Excel file.